



# **Data-Centric Routing Simulation in Cellular Network**

## **Dissertation**

Submitted to

Central Department of Computer Science and Information Technology

**Tribhuvan University**

in partial fulfillment of the requirements for the degree of

**Masters in Computer Science and Information  
Technology**

By

Rashmi Shrestha

Central Department of Computer Science and Information Technology

Tribhuvan University

Kirtipur, Kathmandu

Nepal

**August, 2008**



# **Data-Centric Routing Simulation in Cellular Network**

## **Dissertation**

Submitted to  
Central Department of Computer Science and Information Technology  
**Tribhuvan University**  
in partial fulfillment of the requirements for the degree of  
**Masters in Computer Science and Information  
Technology**

By  
Rashmi Shrestha

Central Department of Computer Science and Information Technology  
Tribhuvan University  
Kirtipur, Kathmandu  
Nepal

**August, 2008**

## **CERTIFICATION**

This is to certify that the dissertation work entitled “**Data-Centric Routing Simulation in Cellular Network**”, submitted by **Rashmi Shrestha** has been carried out under my supervision and guidance. This work is done independently for the fulfillment of Masters Degree in Computer Science. I, therefore recommend for further evaluation.

---

**Prof. Dr. Srinath Srinivasa**  
International Institute of Information Technology  
Bangalore-560100, INDIA  
(Supervisor)



**Tribhuvan University  
Institute of Science and Technology  
Central Department of Computer Science and Information  
Technology  
Kirtipur, Kathmandu  
Nepal**

## **LETTER OF APPROVAL**

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Masters Degree in Computer Science and Information Technology.

### **Evaluation Committee**

---

**Head, Central Department of  
Computer  
Science and Information Technology  
Tribhuvan University**

---

**Prof. Dr. Srinath Srinivasa  
International Institute of Information  
Technology, Bangalore (IIT-B)  
(Supervisor)**

---

**(External Examiner)**

---

**(Internal Examiner)**

**Date:** \_\_\_\_\_

## Acknowledgement

Firstly I would like to bestow my sincere gratitude to **Prof. Dr. Srinath Srinivasa** (IIIT-B) for his supervision and encouragements and I am very grateful to **Mr Sanket Patil** (IIIT-B) for his invaluable guidance and co-operations.

I would also like to express my gratitude to **Prof. Dr. Devi Dutta Paudyal** (Former Head, Central Department of Computer Science and Information Technology) for his inspiration during our two years study of Master Degree.

My heartily thanks to **Prof. Dr. Shashidhar Ram Joshi, Asst. Prof. Arun Timilsina, Dr. Tanka Dhamala** (Head, CDCSIT), **Prof. Dr. Laxmi P. Gewali** (University of Nevada, Las Vegas, USA), **Asst. Prof. Min B. Khati** and all other teachers of my Master Degree for their guidance.

I thanks **Lalita, Bhaskar, Achyut, Dinesh, Rajiv, Anil** for supporting me and always being there for me to help me out and my family for their never-ending support and co-operation.

Thanks

**Rashmi Shrestha**

## Abstract

Mobile data communication has become a very important and rapidly evolving technology as it allows users to transmit data from remote locations to other remote or fixed locations.

Cellular network consists of the network nodes distributed into number of cells. Each cell has an independent fixed base station controlling all the nodes within that cell and all these base stations are controlled by Mobile Switching Station.

Each cell has a fixed frequency allocated. Cell can use its frequency only within its boundary. Adjacent cells cannot use the same frequency but cells that are sufficiently far from each other can use the same frequency and thus the frequency is reused in cellular network. This frequency reuse enables cellular network to handle number of connections with limited number channels.

Mobile nodes can be anywhere at an instance of time. This mobility of the mobile nodes from one cell to another is observed by its respective base station and control of the moving nodes is handed over to the base station of the new cell. This feature of cellular network is termed as handover.

Since the structure of mobile network is very different from the static network, routing in mobile network is under different approach than that of static network.

Host centric routing approach focuses on finding short routes between pair of static nodes whereas the data centric routing approach focuses on finding the single destination of required data with no redundancy, from any node in the network.

Data centric abstraction middleware is one of the distributed data centric application that abstract a network of mobile nodes as a database where an instance gets some data from another application instance simply by querying the network without knowing where that data is located.

In mobile network, the nodes get disconnected frequently due to their mobility; hence they may not be available all the time. So, routing protocols in mobile network are significantly different from traditional routing protocols. To obtain the data from a

node in mobile network, no matter where the node is situated, data centric approach is used rather than host centric approach. In data centric approach routing is performed based on data element and not on the nodes in the network. For this a distributed index is maintained, Distributed Hash Table (DHT), and simulation routing is performed over it. DHT is the overlay network protocol.

However, the simulation of maintaining the DHT, adding and retrieving the records and other DHT aspects are covered in another dissertation work **“Design and Simulation of Distributed Hash Table in Cellular Mobile Network”**.

To improve the way of routing overlay network is used. The overlay network has no control over how data are routed in the underlying network but it can control the sequence of overlay nodes data traverse before reaching its destination.

Computer simulations are invaluable and often unavoidable tool for studying the dynamic behavior and performances of network systems.

Networks usually involve a large number of entities (up to millions) each of which having a potentially complex and time-independent (asynchronous) behavior. Discrete-event simulation techniques are well suited for modeling the individual behavior of entities in such networks. Though the network simulator involves assumptions and simplified models that may not reflect in real network system, the majority of routing protocol research has been done in simulation only.

# Table of Contents

Certification .....	i
Letter of Approval .....	ii
Acknowledgement .....	iii
Abstract .....	iv
Table of Contents .....	vi
Figures List .....	viii
<b>Chapter 1 : Introduction .....</b>	<b>1 – 5</b>
1.1. Cellular Network.....	1
1.2. Mobile Switching Center .....	2
1.3. Base Station .....	3
1.4. Mobile Station.....	4
1.5. Handover.....	4
1.6. Frequency Reuse.....	4
1.7. Cellular Network Operations .....	5
<b>Chapter 2 : Background and Problem Formulation .....</b>	<b>8 – 11</b>
2.1 Background .....	8
2.2. Problem Definition.....	9
2.3. Objective .....	10
2.4. Literature Review.....	11
<b>Chapter 3 : Data Centric Approach Overview .....</b>	<b>16 – 18</b>
3.1. Data Centric over Host Centric.....	16
3.3. Overlay Network.....	18
<b>Chapter 4 : Routing .....</b>	<b>19 – 31</b>
4.1 Routing overview.....	19
4.2. Routing with data-centric approach .....	20
4.3. Routing Scheme .....	23
4.4. Routing process Illustration .....	26
4.5. Routing performance .....	31



<b>Chapter 5 : Simulator Design .....</b>	<b>32 – 35</b>
5.1 Network Simulator.....	32
5.2. Discrete-event Simulation.....	33
5.3. Simulator Description .....	33
5.4. Add record event.....	34
5.5. Retrieve record event .....	34
5.6. Routing.....	35
<b>Chapter 6 : Implementation.....</b>	<b>36 – 45</b>
6.1. Simulator Class Descriptions.....	36
6.1.1. Simulator.....	36
6.1.2. Configurator .....	37
6.1.3. Space .....	37
6.1.4. Chunk.....	38
6.1.5. Node.....	39
6.1.6. Event .....	39
6.1.6. NodeCreation .....	40
6.1.7. NodeMove.....	40
6.1.8. AddRecordEvent.....	41
6.1.9. RetrieveRecordEvent .....	42
6.1.10. Bucket Class.....	44
6.1.11. DHT Class.....	44
6.1.12. The Base Station(BS) Class .....	45
6.1.13. Route Data Class.....	45
<b>Chapter 7 : Simulator Testing .....</b>	<b>47</b>
<b>Chapter 8 : Conclusions and Future works.....</b>	<b>50</b>
<b>Chapter 9 : References .....</b>	<b>51</b>
<b>Chapter 10 : Bibliography.....</b>	<b>54</b>

## Figures List

Figures used in the documentation are as follows:

Fig 1 :	Cell divisions in cellular network .....	2
Fig 2:	Mobile Switching Center .....	3
Fig 3:	Base Station controlling mobile nodes .....	3
Fig 4:	Frequency reuse .....	5
Fig 6:	Cellular network architecture.....	21
Fig 7:	Bucket and Node files illustration.....	22
Fig 8:	Routing Scheme Design.....	24
Fig 9:	Sequence diagram of direction discovery .....	26
Fig 10:	Sequence diagram for checking the neighbor node .....	27
Fig 11:	Sequence diagram of data broadcasting to n-1 nodes.....	27
Fig 12:	Sequence diagram of routing table maintaining .....	28
Fig 13:	Complete Routing Process .....	30
Fig 14:	Implementation Class Diagram.....	36

# **Chapter 1 : Introduction**

## **1.1. Cellular Network**

Mobile devices are small or medium sized devices with mobility, such as mobile phones, PDAs and laptops. Mobile network is the network of such devices. They have very little or no infrastructural support.

Two classes of mobile networks are

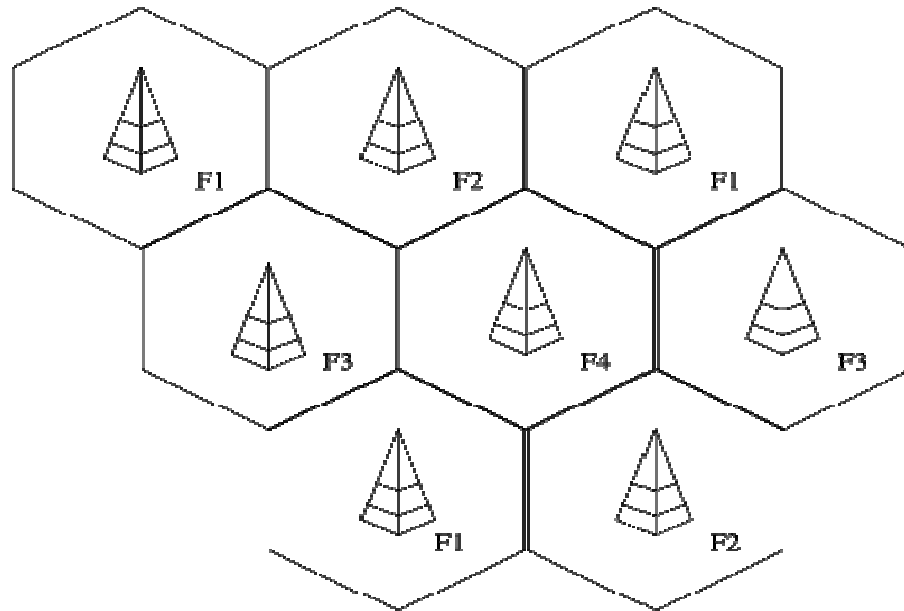
1. Cellular Network
2. Adhoc Network

If the mobile devices can discover and communicate with each other within their ranges without involving central access points, it is peer-to peer or adhoc network. In cellular network mobile units are linked to an infrastructure of switching equipment interconnecting different parts of the system.

The cellular network is divided into number of hexagonal cells. These cells are used to cover different areas in order to provide the coverage to the wider area than the area of one cell.

Each cell has a fixed base station that communicates with all the mobile units in that particular cell simultaneously.

Each cell is allocated a band of frequencies. Adjacent cells are assigned different frequencies to avoid interference. Cells that are sufficiently distance from each other can use same frequency band.



**Fig 1 : Cell divisions in cellular network [28]**

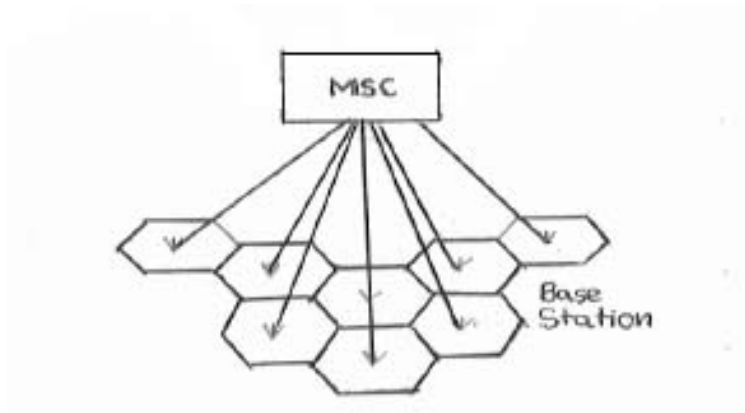
The general system architecture of a cellular network consists of

1. Mobile Station
2. Base Station
3. Switching Centre

The base station is connected to mobile devices via radio interface. Mobile switching centre controls number of cells and, arranges base stations and channels for the mobile and handle the connections.

## **1.2. Mobile Switching Center**

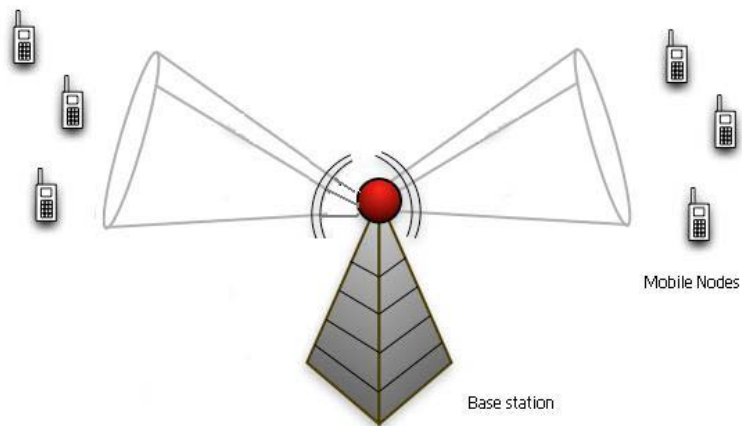
Mobile switching center holds the primary part in the cellular network. It handles all the base stations in network. It plays important role in inter base station processes such as hand over.



**Fig 2: Mobile Switching Center [27]**

### 1.3. Base Station

Base station in cellular network is responsible for handling traffic and signaling between mobile nodes and switching center. A base station serves several different mobile nodes from the same location. It is possible because of the directional antennas on the base station pointing in different directions. Base stations also handle the handover of mobile node control from one base station to another.



**Fig 3: Base Station controlling mobile nodes [29]**

## **1.4. Mobile Station**

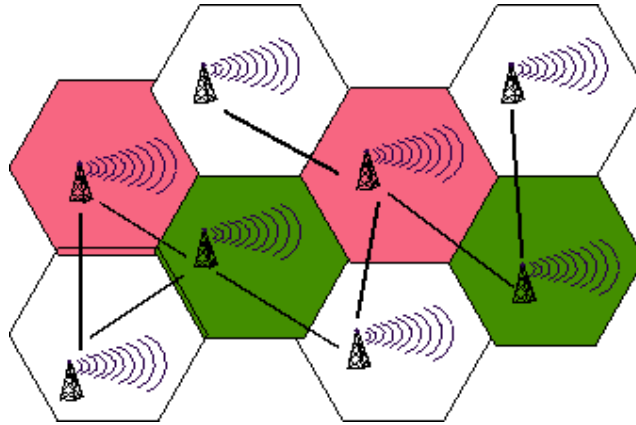
Mobile stations are the network hosts with the mobility. These mobile hosts comprise the cellular network with addition to base station in every cell and mobile switching center for numbers of base stations. They can be in any of the cells in the cellular network at any instance of time. They can move freely in the network and can get out of range. The stations are tracked by the base station. The data are transferred from one mobile host to another which is within the range of the network. A mobile host is under the control of the base station depending upon its location.

## **1.5. Handover**

Since the mobile nodes have mobility, mobile nodes can move from one cell to another in the cellular network. The base station will detect this from the signal power and informs its respective Mobile Switching Center. Then the Mobile Switching Center switches the control of the call to Base Station of the new cell of the mobile node. This behavior is called **handover**.

## **1.6. Frequency Reuse**

In cellular system, **frequency reuse** is achieved. The subset of total number of channels available is assigned to each base station. This increases the capacity by increasing number of channels available to users. And the adjacent cells are not allowed to operate at same frequency since this causes interference between the cells. Whereas in earlier nodes of communications, areas (groups) are allocated dedicated radio frequencies and to ensure that those channels are not affected by transmissions from other users operating at same frequency, sufficient separation between the transmitters must be allowed while allocation of frequencies.



**Fig 4: Frequency reuse [30]**

### **1.7. Cellular Network Operations**

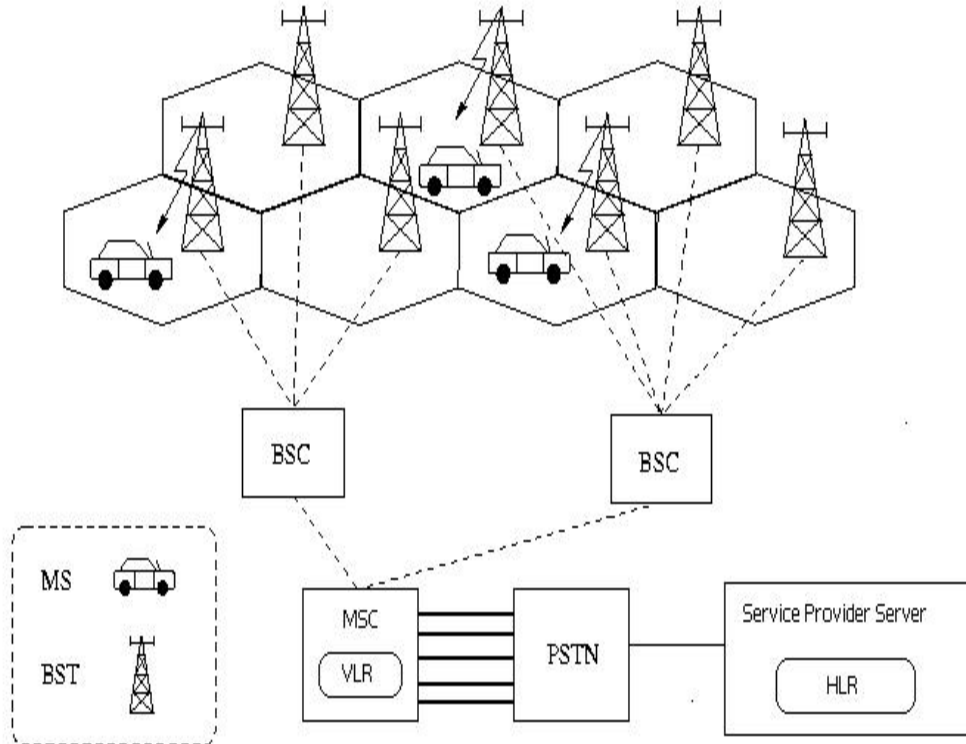
Each cell in the cellular network contains a base station at an appropriate center. At any time, a number of mobile user units may be active and moving within a cell, communicating with the Base Station. Each Base Station is connected to a Mobile Switching Center, with one MSC serving multiple Base Stations.

When the mobile unit is turned on, it scans and selects the strongest setup channel to establish the relationship between mobile unit and nearest Base Station. The mobile unit is identified and its location is registered to MSC through Base Station in the current cell. As long as the mobile unit is on, this procedure is repeated periodically to account for the motion of the mobile unit. If the unit enters a new cell, then new Base Station is selected.

If the mobile unit moves out of the range of one cell into the range of another during the connection, the connection of Base Station to the new cell is to be changed. The system makes this change without either interrupting the call or alerting the user.

It is necessary for the network to monitor the location of every registered mobile station in order to connect the mobile nodes to the network upon request. The management of mobile station location information is handled by the mobility management scheme. The scheme operated mobile stations registering themselves with the BSC of the cell where the mobile station is currently located. A centralized

database stores a list of all the mobile stations in the network, and the BSCs they are currently registered with. A distributed database system in MSC is used to synchronize the database at the BSC and the centralized database at the server provider's premise.



**Fig 5: Cellular Network Operations [29]**

In fig, a mobile station (little car) arrives in a cell served by some BSC. It sends a message identifying itself to the BSC. The BSC sends this message to the MSC, which enters the identity of the mobile station in its visiting location register (VLR). The MSC then notifies the server on the service providers' premise that it must update the home location register (HLR) with the new information about the mobile station's location. The mobility management scheme therefore consists of a distributed database (the VLR and HLR) to maintain location information of all the mobile stations on a network.



Using this scheme, a query to the centralized HLR is all that is necessary to find the current location of a mobile station. In order to keep the HLR current, a considerable amount of information is transmitted across the network. The HLR is updated every time a mobile station moves from one BSCs area into another. As the number of subscribers on the network increases, the information flow of updates to the HLR increases exponentially. Mobility management is also responsible for the authentication of mobile stations to the network. Mobile stations entering MSCs area must be authenticated before they acquire the network's resources. This ensures that only valid customers make use of the network. [24]

## **Chapter 2 : Background and Problem Formulation**

### **2.1 Background**

Small distributed application instances run over mobile nodes and these distributed applications instances need to communicate and exchange data. To get the required data, the application needs to know the location of the data and how to get that data.

Conventionally, if an application instance wants any data from another instance then it has to know the address of the node on which the latter application instance is running. Then the two application instances communicate in a point to point manner. Thus the communication is based on host addresses and the conventional communication in network is host centric. However, the application instances only need to know the data and not the location of the data. As well as, the host centric networking approach works well in case of static networks only. Because, the static networks are managed explicitly and routing of data is based on host centric routing tables.

The mobile network topology is dynamic in nature and there cannot be any special node in the network that will know the entire topology. In Data centric network the network is abstracted as a database and if any node needs any kind of data, it simply queries the network like querying database and gets the data regardless of where the host is situated. Whereas in Host centric networking communication is based on host addresses. It is used in static network topology where data are routed based on static routing table. Host centric approach in mobile network is not feasible. Since availability of data matters much than the host containing the data, data centric approach is used. Routing in Data centric network is based on data and queries rather than the host centric routing tables.

Hence the routing algorithm used in static networks would not be suitable in cellular network.

Since routing in cellular network is based on data centric approach rather than host centric, overlay network concept is followed for routing. Overlay network concept creates a virtual topology on top of physical topology.

## **2.2. Problem Definition**

Data centric approach in the mobile network makes mobile network different from wired network with fixed topologies. In mobile network, the topology is changing all the time. Moreover, in cellular network there is mixed kind of topologies. The mobile nodes in the cellular network are under dynamic topologies whereas the base stations controlling those mobile nodes are under fixed topology. Due to these circumstances, routing in cellular network is quite different from routing in other type of networks.

Every mobile node in the cellular network is under control of a base station. And numbers of base stations are under control of mobile switching center. In this dissertation, existence of only one mobile switching center is being considered which will be controlling a fixed numbers of base stations. Each of those base stations will be controlling number of mobile nodes that are within its range. Since mobile nodes will be moving handing over its control from one base station to another, getting data residing in one mobile node from another mobile node seems impossible under host centric approach. Hence, following the data centric approach, all the data are stored in a distributed way. As the result, if a mobile node needs any kind of information then it is not necessary to connect to every other mobile node one by one until the required data is obtained rather the required data is obtained from the common place of data storage in the network. This makes easy to retrieve required data from any mobile node in the network.

To store the data commonly a metadata is being maintained. The distributed index, Distributed Hash Table [2], is maintained in Data centric networking which is a set of <key,value> pairs that is accessible by every node in the network. A common hash function is used in the network.

DHT is the decentralized distributed system providing lookup service similar to a hash table with (key, value) pair and any node involved in DHT can easily retrieve the value associated with the given key. Node itself is responsible for maintaining the

mapping from name to values resulting to minimal disruption. This feature allows DHT to handle large numbers of nodes and can handle continual arriving, departing and failings of nodes

The data from every mobile node in the network are hashed using a common hash function. The hash value obtained from every data determines where that data is to be stored. For data storage a single bucket in every base station is defined. This bucket contains the information about in which mobile node the particular data resides in key value pair. Collection of these buckets is DHT.

In key value pair of DHT, key is the name in string and value is the hashed value of the name. The bucket where the name is located is determined by comparing the value of key-value pair with the bucket id.

For hashing, SHA-1 hash function is being used.

**SHA-1 hash function** [3] is one of the popular one-way hash function used to create 160 bit digital signature known as message digest which is, to a high degree of probability, unique for a given input sequence. In this hash function, it is computationally impossible to find a message that corresponds to a given message digest as well as to find two different messages that produce same message digest

### **2.3. Objective**

The main objective of this dissertation is to get the data as requested by the nodes in the cellular network with reference to DHT, and route the data to the destination host from the source efficiently.

In cellular network, though IP can be used for host identification, it is not feasible to for routing as in traditional routing in static network. Overlay networks can eliminate the binding between IP and host [9]. IP can be acquired in overlay networks but these IP can no longer be used to define the identity and location of host. As the result, routing can be done using aliases of host instead of IP.

Traditionally, IP addresses are used by routers to transfer packets to their destination using certain routing algorithm. The packet is transferred from the host and routers forward the packet using their routing tables until the receiver host is found. Overlay networks affect this traditional routing function.

Overlay network can undertake the routing decision. With the usage of overlay networks, aiming better routing performance, routers can be functionless, because these overlay networks propose alternative routing paths to the ones determined by routers. The servers contributing the overlay network communicate with the help of routers, but the routing paths between two hosts are not totally determined by or dependent on the current deployed routers. However, totally discarding routers is at the extreme point and can be impossible.

## **2.4. Literature Review**

The topic of routing has been covered in computer science literature for more than two decades, but routing achieved commercial popularity as late as the mid-1980s. The primary reason for this time lag is that networks in the 1970s were simple, homogeneous environments.

In mobile network, numbers of routing protocols are being designed over overlay networks. Some of the routing protocols designed over the DHT are as followings:

Chord [6] is a distributed look up protocol that locates the node storing particular data item given. It efficiently adapts as nodes leave and join the network and answers the queries even if the system is continuously changing. Chord nodes need the routing information about only few other nodes and maintains the routing information as node leaves and joins the network. A chord node requires information about  $O(\log N)$  other nodes for efficient routing, but performance degrades when that information is out of date. Chord uses 160 bit circular id space and forwards the messages in clockwise direction in the circular id space [7]. Chord nodes maintain a routing table called finger table consisting of up to 160 pointers to other live nodes. The  $i$ th entry in the finger table of node  $n$  is the node with the smallest node id clockwise from  $n + 2^{i-1}$ . The first entry in finger table of node  $n$  points to  $n$ 's successor and subsequent entries refer to nodes at repeatedly doubling distance from  $n$ . Each node in chord also

maintains pointers to its predecessor and to its  $n$  successors in the node Id space. The expected routing hops in chord are  $1/2\log_2 N$ .

Pastry [8, 10] is a scalable, distributed object location and routing substrate. Pastry is an overlay and routing network for the implementation of DHT and is similar to chord. The DHT of pastry is similar to other DHT but its routing overlay network built on top of DHT sets it apart. The routing metric can be supplied by an outside program such as ping or trace route to determine the best route to store in its routing table. It performs application level routing and object location in large overlay network on nodes. Pastry is completely decentralized, scalable and self organizing and automatically adapts to the arrival, departure and failure of nodes. Each node in pastry has a unique numeric identifier (nodeId).

For given message and key, the pastry node efficiently routes the message to the node with the nodeId that is numerically closest to the key. The expected number of routing steps is  $O(\log N)$  where  $N$  is the number of nodes in the pastry. At each node in pastry within the route, the application is notified and may perform application-specific computation related to the message. In pastry, the packet is routed on the circular ring and the node whose nodeId is closest to the desired destination will receive the packet. Whenever the node receives the packet and want to forward it, it first examines its leaf set and routes directly to the correct node if found.

Pastry nodeIds are assigned randomly with uniform distribution from a circular 128 bit id space [7]. With 128 bit key, Pastry routes the message the node whose nodeId is numerically closed to the key. Each pastry node keeps track of its neighbor set and notifies applications of the changes in the set. For the purpose of routing, nodeIds and keys are thought of as a sequence of digit of base  $2^b$  ( $b$  is configuration parameter with typical value of 4). Routing table in the node is organized into  $128/2^b$  rows and  $2^b$  columns. The  $2^b$  rows of the routing table contains the IP addresses of the nodes whose nodeId share the first  $r$  digit with the present node's nodeId and  $r+1$  digit of nodeId of the node in column  $c$  of row  $r$  equals  $c$ . The column of row  $r$  that corresponds to value of  $r+1$  digit of the local node's nodeId remains empty. A routing table entry is left empty if no node with the appropriate nodeId prefix is known.

Each pasty node also maintains a neighbor set called as leaf set. It contains  $l$  nodeIds of nodes that are numerically closest to the nodeId of current node with  $l/2$  larger and  $l/2$  smaller nodeIds than current nodeId. The value of  $l$  is approximately  $\lceil 8 \cdot \log_2^b N \rceil$  where  $N$  is the number of nodes in the overlay. The message to be routed is forwarded to the node in the routing table whose nodeId shares with the prefix that is at least one or  $b$  digits longer than the prefix that the key shares with the present node's nodeId. If no such node is found then the message is forwarded to the node whose nodeId shares prefix with the key as long as the current node, but is numerically close to the key than the present node's id. Again if no such node is found then routing table or in neighbor then the current node or its immediate neighbor is the destination.

Tapestry [11] is a DHT which provides a decentralized object location, routing and multicasting infrastructure for distributed applications. It is overlay network offering efficient, scalable, self repairing location aware routing. Tapestry, CAN, Chord, Pastry, these overlays implements a basic key based routing mechanism and allow deterministic routing of messages and adaptation of node failures in the overlay network. Tapestry is an extensible infrastructure providing decentralized object location and routing focusing on efficiency and minimize message latency. The routing tables are constructed locally from initialization and maintained in order to reduce routing stretch. It also allow multicasting in overlay network. Each tapestry node is assigned a unique node id and SHA-1 hash function is used to produce 160 bit space represented by 40 digit hex key. Application specific GUID's are also assigned a unique identifier. Tapestry efficiency is said to increase with increase in network size so multiple application sharing same overlay network is said to increase the efficiency.

[7] Tapestry is similar to chord but differs in mapping keys to nodes and managing replication. The neighboring nodes are not aware of each other in tapestry. When a node's routing table doesn't have an entry for the node that matches the key's  $n$ th digit, the message is forwarded to the node with next higher value in the  $n$ th digit, modulo  $2^b$  found in routing table. This process is called surrogate routing and maps keys to unique node if node routing tables are consistent. The expected number of routing hops in tapestry is  $\log_2^b N$ .

CAN [12] uses  $d$ -dimensional Cartesian space and space is partitioned into rectangular zones where each zone is maintained by a host. Each object is hashed to the point and assigned to the host. Similar to routing in other overlay network, the routing in CAN is greedy. The message is passed to neighbor closest to destination. Routing in CAN is said to be inefficient when  $d$  is small. CAN routing is  $O(N^{1/d})$  with  $O(d)$  routing state per node. [7] Each node in CAN maintains the routing table with  $O(d)$  entries and any node can be reached in  $(d/4) (N^{1/d})$  routing hops on average. Unlike Chord, Pastry and Tapestry, CAN's routing table doesn't grows with network size but the number of routing hops grow faster than  $\log N$

[13] One of the key features provided by systems such as CAN, Chord, Pastry, and Tapestry are scalable routing performance while maintaining a scalable amount of routing state at each node. Scalable routing paths mean the expected number of forwarding hops between any two communicating nodes is small with respect to the total number of nodes in the system. Chord, Pastry, and Tapestry scale with  $\log N$ , where  $N$  is the system size, while maintaining  $\log N$  routing state at each overlay node. CAN scales with  $D \cdot N^{1/D}$ , where  $D$  is a parameter with a typical value of 6, while maintaining an amount of per-node routing state proportional to  $D$ . A second key feature of these systems is that they are able to route to destination addresses that do not equal the address of any existing node. Each message is routed to the node whose address is "closest" to that specified in the destination field of a message. This feature enables implementation of a distributed hash table (DHT), in which content is stored at an overlay node whose node ID is closest to the result of applying a collision-resistant hash function to that content's name (i.e. consistent hashing).

Another routing protocol designed for wireless network is Ad Hoc On Demand Distance Vector (AODV) [14]. It is an on-demand routing protocol that uses traditional routing tables to store routing information. It is capable of both uni-cast and multicast routing. AODV uses timers at each node and the routing table entry expires after the route is not used for a certain time. AODV [15, 16, 17] is the reactive routing protocol and routes are determined only when needed. When a source has data to transmit to an unknown destination, it broadcasts a Route Request (RREQ) for that destination. At each intermediate node, when a RREQ is received a route to the source is created. Nodes receiving this packet update their information for the source node



and set up backwards pointers to the source node in the route tables. If the receiving node is not the destination and does not have a current route to the destination, it rebroadcasts the RREQ. If the receiving node is the destination or has a current route to the destination, it generates a Route Reply (RREP). The RREP is unicast in a hop-by-hop fashion to the source. As the RREP propagates, each intermediate node creates a route to the destination. When the source receives the RREP, it records the route to the destination and can begin sending data. If multiple RREPs are received by the source, the route with the shortest hop count is chosen. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it. As data flows from the source to the destination, each node along the route updates the timers associated with the routes to the source and destination, maintaining the routes in the routing table. If a route is not used for some period of time, a node cannot be sure whether the route is still valid; consequently, the node removes the route from its routing table.

DSR [18] (Dynamic Source Routing) is another routing protocol designed for multi-hop wireless adhoc network. This protocol is composed of two routing mechanisms: Route discovery and route maintenance. DSR is also an on-demand algorithm. Route Discovery is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D. Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D. Route Maintenance is the mechanism by which node S is able to detect, while using a source route to D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works. When Route Maintenance indicates a source route is broken, S can attempt to use any other route it happens to know to D, or can invoke Route Discovery again to find a new route. Route Maintenance is used only when S is actually sending packets to D.

## **Chapter 3 : Data Centric Approach Overview**

### **3.1. Data Centric over Host Centric**

In network, data are routed from one host to another. There are numbers of routing algorithms which can be used depending upon various networking factors, to achieve maximum performance in routing. While getting any kind of data in network one host needs to request data to another host. The location of the data is found and data transferring is started.

Location of the host with the required data is considered important than the data itself. This is host centric network. In static network host centric approach is used, whereas in dynamic network with mobiles node host centric approach would not be effective to use and data centric approach must be implemented.

In static network, if an application instance wants to get data located on some other node then firstly the location of the destination node is obtained and required data is routed to the source node using the convenient type of routing table. Since the destination node address is prior known, this approach is host centric approach. Host centric approach works well in static network because in static network, network topology is explicitly managed and data are routed based on routing table.

But, the application instance running in the node is interested in only the data and not the location of the destination node where that data resides. Moreover, the mobile network topology changes constantly due to the mobility in the nodes resulting to frequent disconnection and failures. In mobile network, maintaining routing tables is not possible so host centric approach is not possible in mobile networks.

In host centric, host to host communication takes place and hosts are assumed to know which host to contact for required data. However, an increasing number of applications involve accessing particular data objects whose location can't easily be determined within the host centric architecture [25]. In such type of architectures data centric approach is more feasible.

In the overall data transferring process within the network, data is of much importance rather than the host where the data resides. No matter where the host is, more important is the data in that host machine. This highlighting of data rather than the data residence is defined in data centric approach.

Since applications are interested in data and not the location of data, they should be able to get the required data regardless of the data location. Data centric approach abstracts the network as database and applications query the network for the data required resulting to the query being routed to the node holding that data. Instead of static routing tables, routing is done on basis of the query and data.

The database abstracted from the network is queried using simple database queries like select, update, delete etc. These queries returns the data requested.

Any data can be searched using select query in the database of network as if the whole network data is locally available to the user. The user need not know where the data is located. However, one query can requests for the data located in more than one node, in that case, query is decomposed and processed separately.

The mobile nodes can be unavailable at any instance and users may query for the data located in the unavailable node. Then the middleware maintains the data availability in such node failures by replicating the data across the network in different nodes. However, consistence replication of data is to be maintained and the garbage replications are to be removed [20].

### **3.2. Distributed Applications on Data-Centric Approach**

Mobile application instance can communicate with other instances and form a collaborative group for any reason. If the user wants to share any kind of information in his group he can add data in the network abstracted database by insert query and other users in the group can access same data using select query without knowing the exact location of where the data is stored [26].

Followings are the potential distributed applications over cellular network.

1. Distributed Address Book

Distributed address book is a mobile collaborative application where the data centric abstraction middleware allows the user to search for contact number of other users in collaboration. If we need to get contact number of one of the user, then instead of guessing who might have that contact information and querying each of the guesses, the data centric abstraction middleware allows us to query the whole database for the contact number and returns the search result as if the whole data base is our own local address book.

2. Distributed Social Book marking

Distributed book marking application allows book marking and organizing them. The data centric abstraction middleware allows sharing the book marking among each other and searching much larger space than that the own bookmarks.

### **3.3. Overlay Network**

In mobile network routing researches, overlay networks are being used. Overlay network is constructed in order to permit the routing to destination not specified by IP address. DHT is an example of overlay network where routing is performed by specific logical address, whose IP address is not known in advance. Overlay network is built on top of an existing network. Formally, overlay network is defined as a virtual network of nodes and logical links that is built on top of an existing network with the purpose to implement a network service that is not available in the existing network [5]. Internet is also considered as an overlay network built on top of local area networks to connect local area networks and adds an internet protocol headers in all the packets transferred.

For routing in mobile nodes, overlay network can be built on top of existing network where the nodes in overlay network define the neighbor nodes by content stored, and they can change the search process from standard graph traversal problem into a localized iterative process. In this process, each hop brings the query closer to its target set of hops, which can be calculated according to the mathematical function. This reduces the overall network load and makes the query process deterministic. Thus overlay network works like DHT and allows key insertion, querying and removal.

## Chapter 4 : Routing

### 4.1 Routing overview

**Routing** (or **routeing**) is simply defined as the process of selecting paths in computer networking to send data. Routing is choosing a specific route among the multiple paths between source and destination. In an efficient routing algorithm, the best path is chosen.

Routing directs forwarding, the passing of logically addressed packets, from their source toward their ultimate destination through intermediary nodes.

There are various routing algorithms designed for static network which gives the best performance depending upon the numbers of networking factors. However, routing in static network and mobile network is completely different from one another besides transferring of data from source to destination host.

The routing algorithm designed should have following properties

- Correctness
- Simplicity
- Robustness
- Stability
- Fairness and
- Optimality
- Convergence

The routing algorithm should be designed as simple as possible and efficient with minimum numbers of software and utilization overheads. The routing should be done in a simple manner so that the overhead is as low as possible. With increasing complexity of the routing algorithms the overhead also increases.

Optimality refers to the capability of routing algorithm to select the best route which depends upon the routing algorithm metrics.

Routing algorithm must be robust and stable. They should perform correctly in the face of certain numbers of unusual and unforeseen circumstances as well, such as hardware failure, high load conditions and incorrect implementation. The routing algorithms that proves to be working under variety of network conditions is said to be the best routing algorithm.

Routing algorithm must converge rapidly. Convergence is the process of agreement for the optimal path. When one of the routers goes down or become unavailable resulting to the recalculation of the optimal path, the routing algorithm that converges slowly may cause the routing loops.

## **4.2. Routing with data-centric approach**

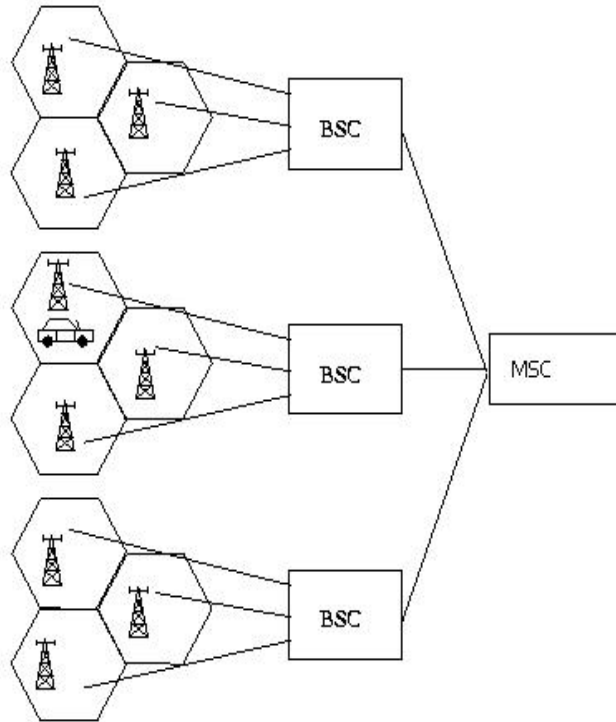
Numbers of routing schemes are being designed for ad hoc network. But, again within mobile network as well, the routing between cellular network and ad hoc network differs. This is because cellular network is under mixed topology and ad hoc network is totally under dynamic topology.

In cellular network, the nodes remain in mobility whereas base stations controlling those nodes remain stationary. Mobile node may be under one base station at a time and at another instance of time may be under different base station. The base station controlling the mobile host can be determined by the position of the mobile node, where the mobile is currently located.

Each cell in the cellular network contains a base station in an appropriate center. At any time, a number of mobile user units may be active and moving within a cell, communicating with the Base Station. Each Base Station is connected to a Mobile Switching Center, with one MSC serving multiple Base Stations.

If the mobile unit moves out of the range of one cell into the range of another during the connection, the connection of Base Station to the new cell is to be changed.

In our design, the whole network is divided into number of chunks or cells and in every chunk one base station is located, those base stations are further controlled by a Mobile Switching Center.

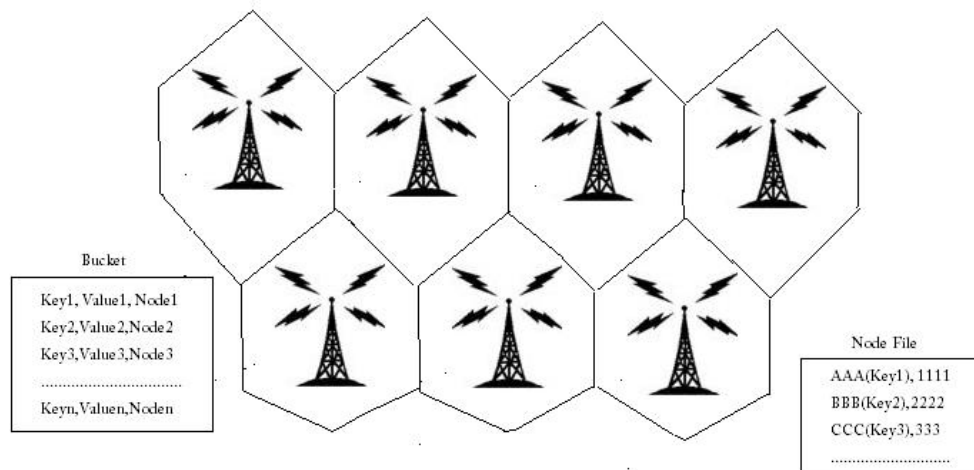


**Fig 6: Cellular network architecture [29]**

For routing data in data-centric cellular network, DHT is used and the following processes are followed.

- Every Base Station contains a bucket where key (name), value (hash value) and node id are stored.
- If any node needs the contact number for a particular name say AAA, then the bucket where the contact number for AAA is stored is obtained.
- The bucket is searched for AAA and node id containing the record of AAA is obtained.

- If node is not in the same Base Station, then the new Base Station of the node is searched first.
- After retrieving the record being searched, that record is routed from the base station which contains the bucket where the record is located to the node requesting the record.
- Firstly, the base station of the mobile node requesting for the record is obtained. And request name AAA with its contact number is routed to destination base station to further to the destination mobile node.
- Before the record being searched reaches the destination mobile node, if the destination mobile node move under some other base station then again the new base station controlling that mobile node is to be obtained.
- This process is continued until the name and contact number being searched is stored in the file of mobile node initiating the search.



**Fig 7: Bucket and Node files illustration**



### 4.3. Routing Scheme

Routing the data packets from source to destination is under the network layer function. The general routing process holds two processes within. First process handles each data packets as it arrives, working up the outgoing path to use for it in routing table. This process is called forwarding. And the other process is responsible for filling in and updating the routing tables. The routing algorithm is specifically used in second process.

In this routing process, all stationary nodes in the network are arranged in the circular ring interconnecting each other and are ordered in clockwise direction starting from the least base station id to higher. If  $i$  is the identification number of a node, then the node with identification number  $i-1$  is prior to node  $i$  and node with identification number  $i+1$  is behind the node  $i$ .

Data being searched is retrieved from DHT routed to the destination node in the network. The routing process designed is composed of numbers of steps as follows:

- Direction Discovery
- Checking the first neighbor
- Data Broadcasting
- Maintaining routing table

First step in our routing process is **Direction Discovery**.

In this step, the direction of the routing process is determined. Following scheme designed is implemented to decide whether the routing process should proceed in clockwise or anti clockwise direction and is implemented only once at the starting of each routing process. **S** is the source base station Id and **R** is base station of the destination node. **T** is the total numbers of base stations in the network.

S: SenderId

R: ReceiverId

T: Total number of nodes

When  $(S-R) < 0$ , (-) ve

$|S-R| < T/2$ , move towards right direction

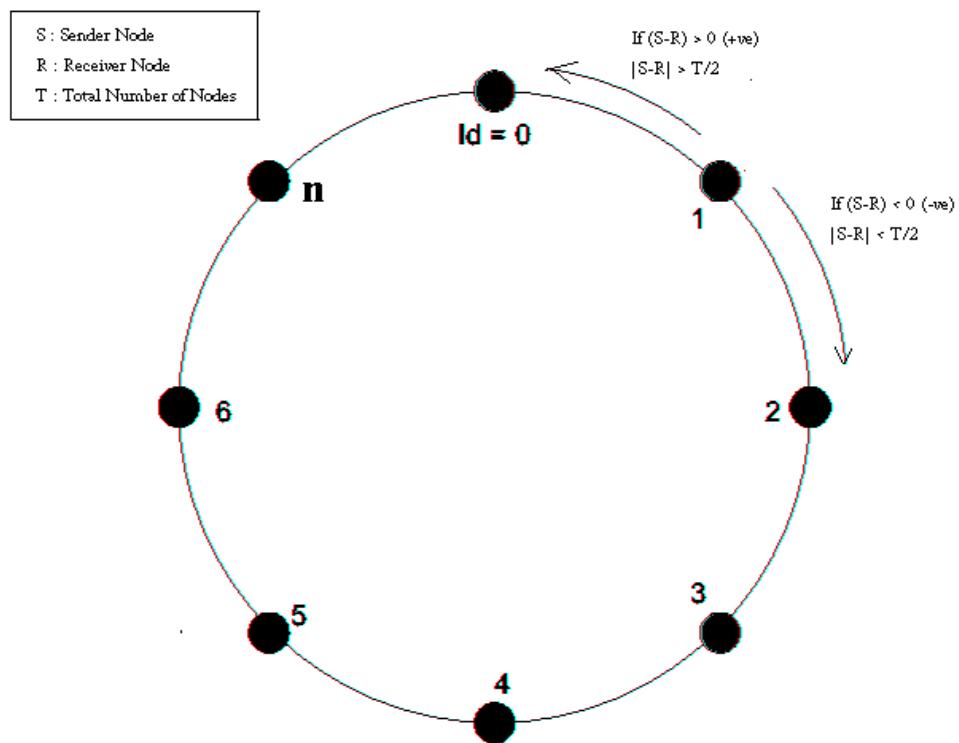
Else move towards left

When  $(S-R) > 0$ , (+) ve

$|S-R| > T/2$ , move towards right

Else move towards left.

NOTE: If T is odd, take floor of  $\lfloor T/2 \rfloor$



**Fig 8: Routing Scheme Design**

After determining the direction, second step is **checking the first neighbor** towards the direction obtained. The first node in the obtained direction is checked whether the

destination node is under its range or no. If yes, the destination node is reached. If no, then third step is carried out.

The third step is **data broadcasting**. The source node broadcasts the message containing the destination base station id and node id to all the other T-1 number of nodes, except itself. Every node receiving the broadcasted message checks base station id in the message and replies back, only if the message contains its own base station id and the node id in the message is within its own range. If the message doesn't contain its own id, then that node ignores the broadcast.

The reply to the broadcast message contains the destination base station id and node id if the destination node is still under the range of same base station, but if the destination node is not within the range of same base station then instead of the destination node id some negative number, say -1, is sent in the reply.

The source base station after receiving the reply checks the node id in the reply, if the reply contains the destination node id then the Name being searched and the contact number retrieved is sent to that base station replying the broadcast and further to the destination node id. But if the reply doesn't contain the destination node id but the negative number -1, then it assumes that the destination node id is not within the network and is out of range.

After receiving the reply, fourth step of **maintaining routing table** is implemented. The information received in the reply is stored in routing table of the source node in sequence of destination node Id, destination base station Id and direction such that later if same node id again requests the data then there is no need to implement the whole routing process again. The routing table maintained can be directly referred to get the base station id of requesting node id. Since the base stations are assumed to be in sequential ordered forming a circle, number of hops to reach the destination node is obtained from the destination node and the source node itself in the direction stored in the routing table.

Since the network topology keeps changing, there is no guarantee that the routing table maintained contains the updated information. So, before beginning the routing process, source base station first searches the route to the base station id of the destination node in its own routing table, if base station id of the destination node is

already in the routing table then source base station sends the message to that base station. If the destination node is under the same base station as recorded in the routing table, then base station replies back with the node id otherwise negative number -1 is replied back and in that case only, the routing process is started. Likewise, if no entry is made in routing table for that node id then the whole routing process is to be implemented.

Hence, every base station node maintains a routing table which contains the data that are searched till now and the bucket id.

The source node waits for reply for some period of time, say 1 minute, in this design, and if no reply is received then it assumes that the node requesting the data is not within the entire network.

#### 4.4. Routing process Illustration

##### Step 1: Direction Discovery

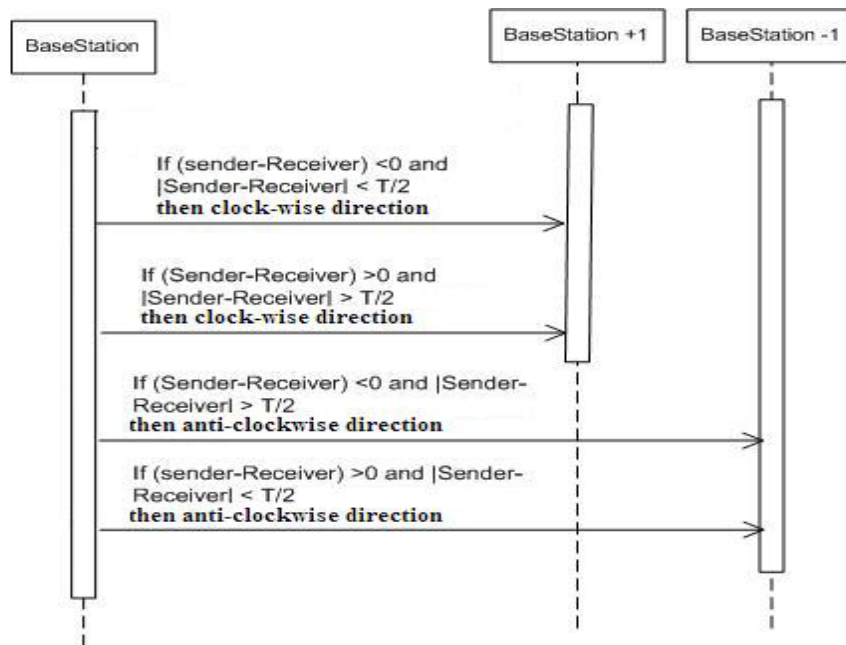
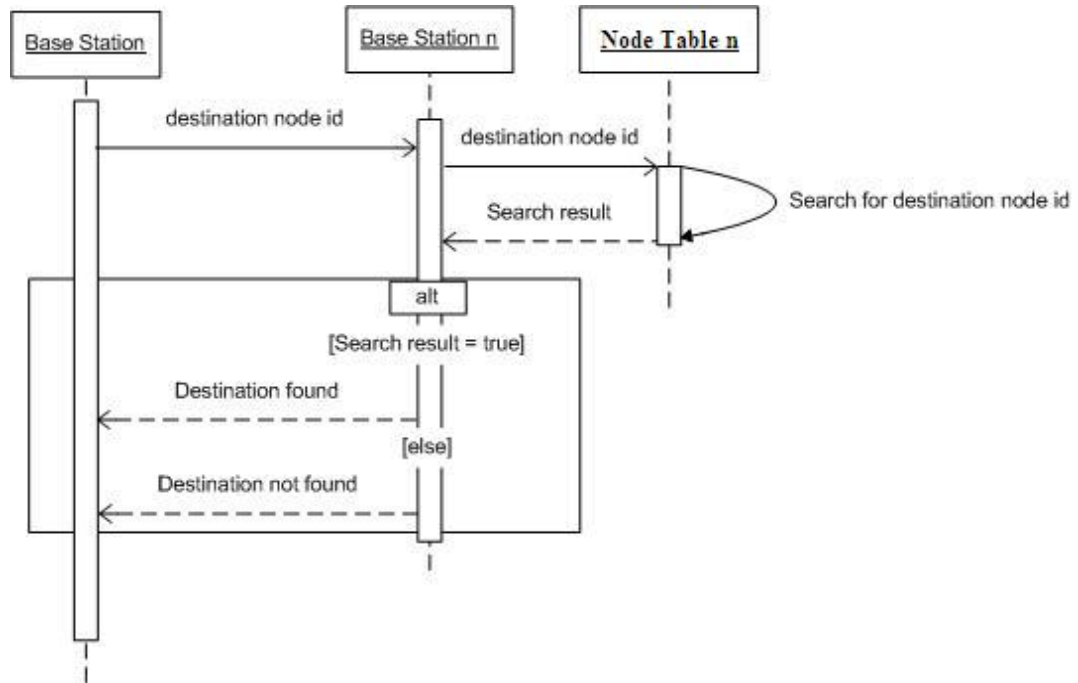


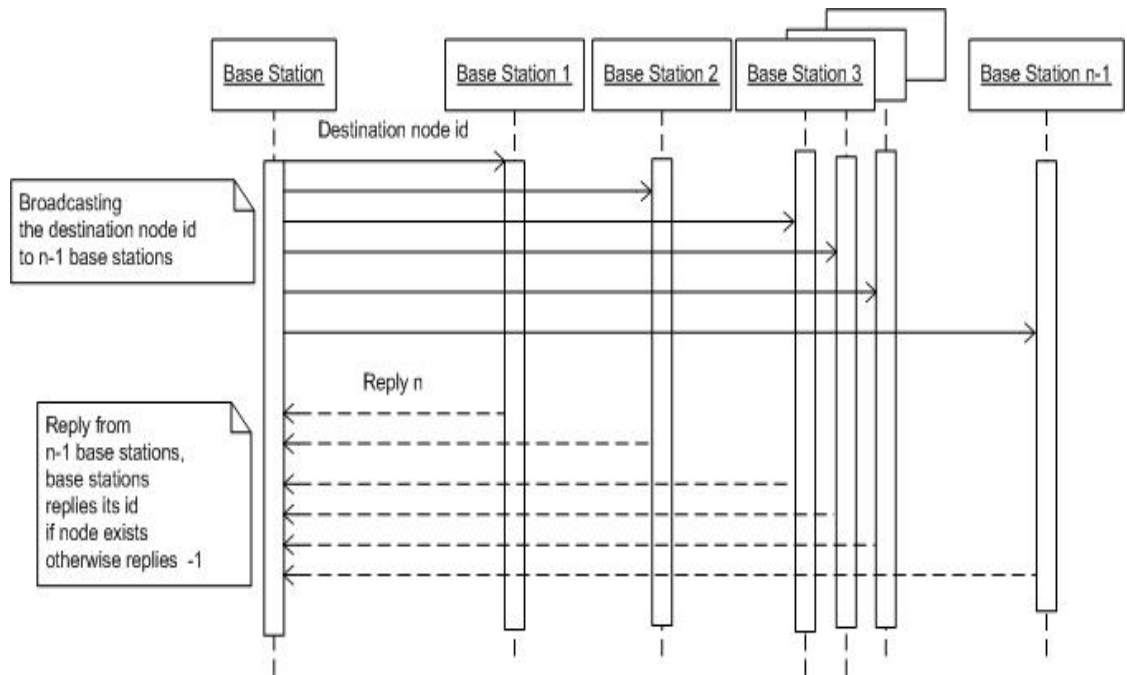
Fig 9: Sequence diagram of direction discovery

**Step 2: Check neighboring node**



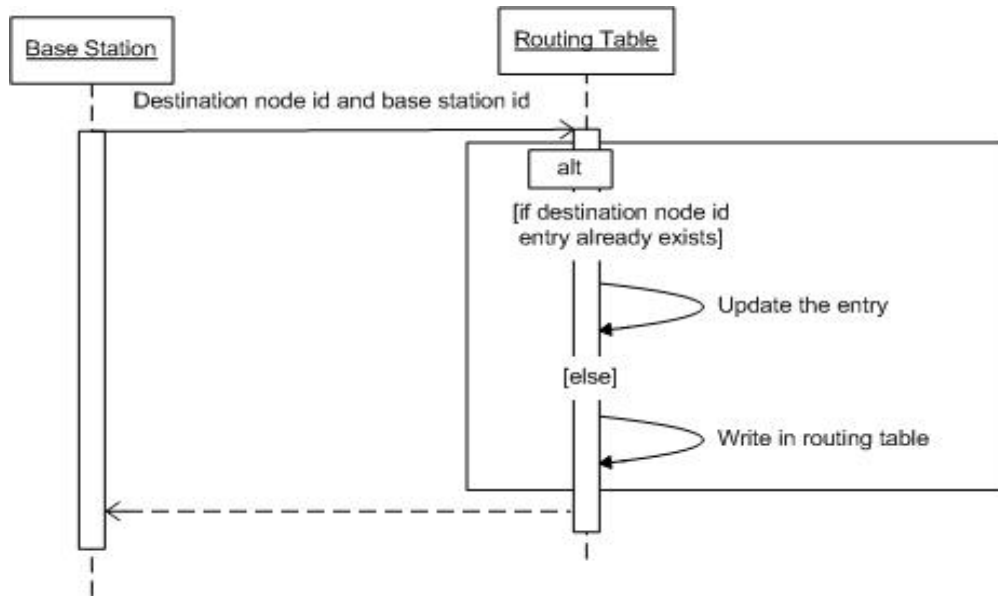
**Fig 10: Sequence diagram for checking the neighbor node**

**Step 3: Broadcasting**



**Fig 11: Sequence diagram of data broadcasting to n-1 nodes**

#### Step 4: Maintaining Routing Table



**Fig 12: Sequence diagram of routing table maintaining**

Two cases that may arise in the above designed routing process are as follows:

#### **Case I:**

#### **When Sender id is less than Receiver id (S<R)**

If Sender id is base station 2, receiver id is base station 10 and total numbers of base stations are 15.

Then,

#### **Step 1:**

$$S-R = 2-10 = -8 \text{ (-ve)}$$

$$|S-R|=8 \text{ and } T/2=15/2=7.5$$

Since T is odd floor of T/2 is taken

Since,  $|S-R| < T/2$  then move towards right direction else move towards left direction.

And,  $8 > T/2$ , the routing is towards left in anti-clockwise direction, that is, towards base station 1.

**Step 2:**

The first neighbor in anti-clockwise direction, base station 1, is checked if the destination node is under base station 1. If yes, the destination node is reached and if no, the further steps of the routing process are implemented.

**Step 3:**

The source node broadcasts the message with searched data to all the T-1 nodes in the network in anti-clockwise direction. Base station 2 broadcasts the destination base station id and node id to 15 nodes. In this case, AAA is broadcasted to base stations 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0 and 1.

**Step 4:**

The broadcasting node, base station 2, waits for reply for 1 minute and if no reply received then it assumes destination node is not within the entire network. If any reply is received, base station 2, maintains the destination node id and its base station id in its routing table along with the direction calculated.

**Case II:****When Sender id is greater than Receiver id (S>R)**

If Sender id is base station 10 and receiver id is base station 2 and total numbers of base stations are 15

Then,

**Step 1:**

$$S-R = 10 - 2 = 8 \text{ (+ve)}$$

$$|S-R|=8 \text{ and } T/2=15/2=7.5$$

Since T is odd floor of T/2 is taken

So,  $|S-R| > T/2$  and move towards right direction

Now the routing is towards right in clockwise direction, that is, towards base station 11.

**Step 2:**

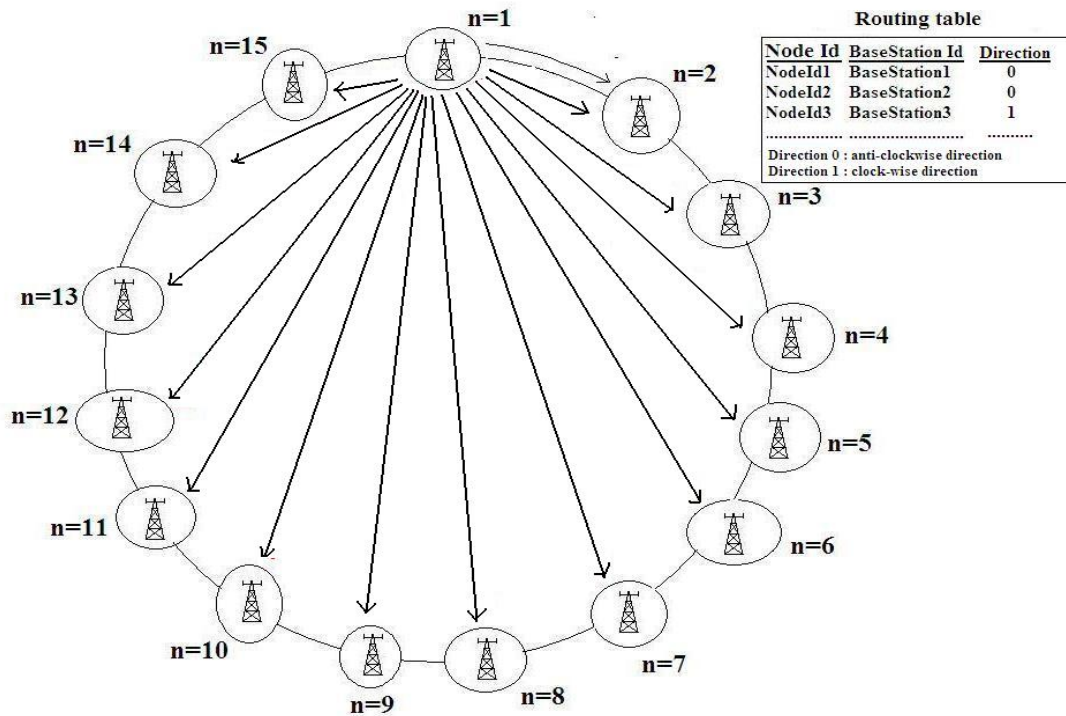
The first neighbor in clockwise direction, base station 11, is checked if the destination node is under the range of base station 11. If yes, the destination node is reached and if no, the further steps of the routing process are implemented.

**Step 3:**

The source node broadcasts the message with searched data to all the T-1 nodes in the network in anti-clockwise direction. Base station 10 broadcasts the destination base station id and node id to 15 nodes, 11, 12, 13, 14, 15, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

**Step 4:**

The broadcasting node, base station 10, waits for reply for 1 minute and if no reply received then it assumes that the destination node is not within the entire network. If any reply is received, base station 10 maintains destination node id and its base station id in its routing table and the direction obtained.



**Fig 13: Complete Routing Process**



## **4.5. Routing performance**

Routing process defined above gives scalable routing performance since every overlay node maintains the scalable amount of routing state at each node. Each node maintains  $\log N$  routing state where  $N$  is the total number of nodes in the system. Routing path between two nodes is small with respect to the total number of nodes in the system.

The routing process defined above determines the routes only when needed and uses traditional routing tables to store routing information.

This routing process gives better performance with higher number routes determined. Once a route is obtained it is stored in routing table which is used when same data is requested again and the whole routing process implementation is not required.

So in the network with  $n$  stationary nodes, if a node has  $n$  entries in routing table, its efficiency is  $\log n$  since the node needs to look up only  $n$  entries in routing table and no routing process is to be implemented.

## Chapter 5 : Simulator Design

### 5.1 Network Simulator

The system is modeled using number of tools and techniques and simulation is one of them. Simulation is defined as:

"The technique of imitating the behavior of some situation or system by means of an analogous model, situation or apparatus either to gain information more conveniently or to train personnel".

So, the simulation is technique to build model of real or proposed system in order to study to behavior of the system under specific conditions following the time progression.

Network Simulation is the technique where a program simulates the behaviors of a network. The program performs the simulation by calculating the interaction between the different network entities and hence the behaviors of the network can be observed. The various attributes of the environment can also be modified in a controlled manner to assess these behaviors under different conditions. The users can also customize the simulator to fulfill their specific analysis needs. Some network simulators require input scripts or commands and produce trace-files. The network parameters describe the state of network (node placement, existing links) and events (data transmissions, link failures etc). Trace files can document every event that occurred in simulation and are used for analysis. Some simulators incorporate or provide visualization tools for trace files.

The simulation approaches are:

1. continuous
2. discrete

In continuous simulation, the time is controlled by continuous variables expressed as differential equations and during the simulation the software will integrate the equations.

And within discrete simulation, there are number of logic expressions that are evaluated at discrete points in time [4].

## **5.2. Discrete-event Simulation**

Discrete event is the event that occurs at an instant of time. The discrete events do not have time duration. However, the starting and ending of such activities, with the time duration, can be considered as two separate discrete events.

Discrete event simulation is to study complex system by computing the times that would be associated with real events in a real life situation. The idea of discrete event simulation is to compute physical times that would occur in real time in physical system, but without waiting for the delays between two events to occur in real time. The discrete event simulation is a sequential program accomplished using a queue of events that are ready to be performed. The event queue is kept sorted in increasing time order and the program processes the events in increasing time order. The program is started with one or more events initially in the queue and other events are added to either event queue sequentially. These events are caused by the initial events in the queue and the program has built in knowledge about how each kind of event causes other sequential events.

When the simulation grows larger and lots of concurrency occurs as result of the events that are to be processed simultaneously. The work of processing events is distributed across many simulators. Each simulator is responsible for processing certain kinds of events, maintaining its own event queue and processing the events in time order. The new event occurred are placed in local event queue or may need to be sent elsewhere for processing.

## **5.3. Simulator Description**

The cellular network simulator is named as cAsset. Asset is Adhoc System Simulator and Enhancement Test-Bed. It is the discrete event simulator. cAsset is designed on the top of Asset which is an adhoc network simulator. In cAsset all events are fired one after another. An event takes place only after completion of previous event. The events are as follows:

- a) Node Creation Event
- b) Node Reposition Event
- c) AddRecord Event
- d) Retrieve Record Event

Among these events, some are inherited from Asset.

#### **5.4. Add record event**

- AddRecordEvent is invoked from the event list. The node invoking the event selects a name randomly from a common text file. In the text file, numbers of names are placed along with the contact number. This name and contact number is also written in the node file of this node (Every nodes has a separate text file ).
- In this event, the record is selected randomly from the external file and hashed using SHA-1 hashing algorithm. The hashed value produces 16 bit number which is converted to an integer value.
- The value obtained after hashing is compared with hash value of the every base station and the base station id where the selected name should reside is obtained.
- Every base station contains a bucket. For simplicity, each bucket has the same identification number as of its base station. The bucket whose id is equal or nearest greater integer value of that record is obtained
- And in the bucket of the base station the selected name, the hashed value and the node id invoking the event is added

#### **5.5. Retrieve record event**

- RetrieveRecordEvent is also invoked from the event list. The node invoking this event selects the name randomly from the common text file, where only the names of the contacts are stored.

- The name selected is hashed using SHA-1 hash function. Since SHA-1 hash function returns 160-bit of value and value obtained from every key is to be compared to the base station to find the bucket where that record can reside, every base station identifiers are also hashed using SHA-1 hash function. The obtained hash value of the key is compared with the hash value of base station one by one. And, the base station where the selected name resides is obtained. The hash value of the selected base station should be equal to the hashed value of key or most nearly greater to it.
- In the base station, the name requested is searched in the bucket one by one sequentially starting from the top of the bucket. The first found record in the bucket with the matching name is to be routed to the requesting node.
- Along with name in the bucket, the node id where the name is located is also stored. The node id is checked. If the node is within the base station, the name and contact number is obtained and routed to the node invoking this event.
- If the destination node is not within the base station, the node is located in other base stations. After locating the node, the name and contact number is received and routed to the destination nodes.

## **5.6. Routing**

All the base stations in the overlay network are considered to be arranged in the circular order according to the base station identification number. After the record of the requested name is retrieved from RetrieveRecordEvent, the record is routed from the base station which contains the bucket where the record is located to the requesting node.

Firstly, the base station id of the requesting node is obtained. Now, the record is routed from its source base station to the requesting base station. The routing scheme is repeated in every base station during routing until the requesting node is found.

In simulator implementation, the routing of records is done in RetrieveRecordEvent class.

## Chapter 6 : Implementation

The implementation of simulator is done in JAVA. The class diagram of the implementation is as follows

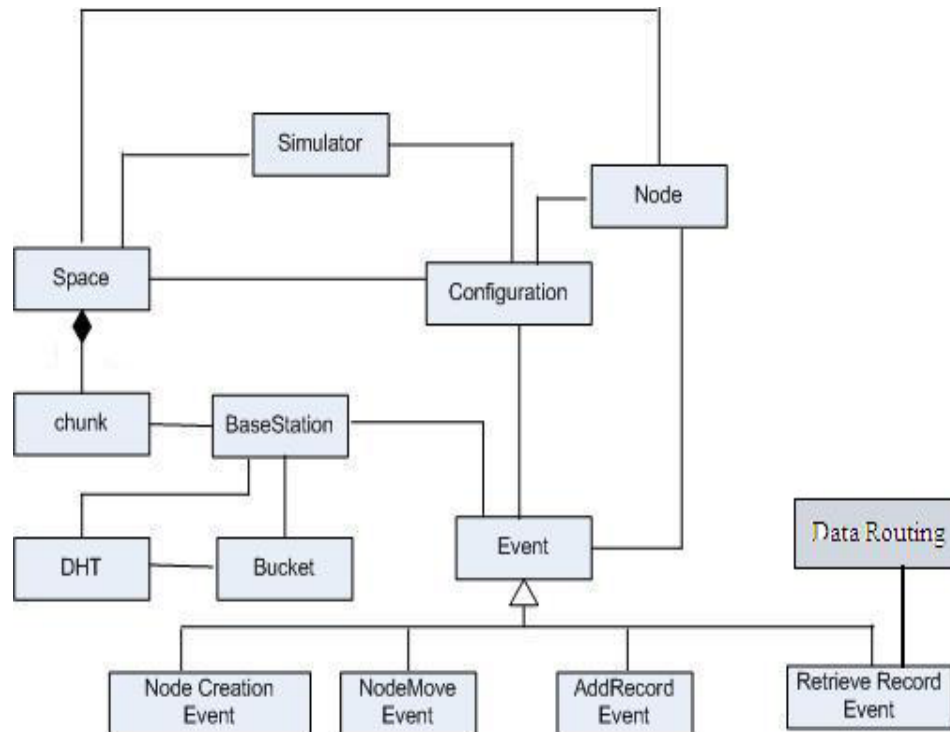


Fig 14: Implementation Class Diagram

### 6.1. Simulator Class Descriptions

#### 6.1.1. Simulator

The simulation starts with this class. It consists of an object of Configurator class. It initiates the enqueueing of events in the event queue.

**Objects used:**

1. nodespace : object of type Space
2. ConfObj : object of type Configurator
3. EventQ : a linked list of the events occurring in the cellular network
4. Localtime: the timestamp

**Methods used:**

1. Simulator (Space s, Configurator co): It sets nodespace as s and ConfObj as co
2. Run ( ) : It sets the localtime as timestamp for the next event in the Event queue and then executes that event by calling the happen( ) method
3. addEvent(Event e): Adds an event to EventQ in its place if and only if its time stamp is below timelimit
4. main(String args[]) does the following: creates objects for configurator , space , simulator . The simulation starts with the command 'start'. The node creation event is added to the event queue. If number of nodes is specified then the node creation events will be enqueued to the event queue .The number of events depends on the number of nodes specified or the udensity factor. The simulation stops at the time as timelimit (parameter set in the Configurator class). To stop the simulation before this time issue the command 'quit' may be used.

### **6.1.2. Configurator**

It configures various parameters (such as number of nodes, time limit, radius, udensity, space size etcetera) for the simulation. The configurations decide the behavior of the system. The parameters are fed in the form of a file - The Configurator file where the values for each parameter are stored beforehand.

### **6.1.3. Space**

This is the cellular network space. This is the entire bounded space where the cellular network system is simulated. The space has collection of mobile devices (represented

by **Node** class). The space is divided into cells or **Chunks**. Each chunk has a fixed **Base-Station**.

**Parameters used:**

1. xmin , ymin , xmax , ymax : The coordinates which bound the space under consideration.
2. Nodelist : a linked list of all the nodes in the space
3. Chunklist : a linked list of the chunks in the space
4. ConfObj : an object of type Configurator

**Methods used**

1. Space (Configurator co) : It makes a linked list of chunks. Their ids vary from 0 till n-1 , where n depends on the following parameters. The boundaries of the space (xmin,ymin) and (xmax,ymax) and the chunksize. All these parameters are obtained with the help of the the object co.
2. int determinechunk(int x, int y) : For the node located at position (x,y) , the method returns the id of the chunk to which the node belongs to. This calculation depends on the node's position, the chunk-size and the boundary of the space.

### **6.1.4. Chunk**

These are the partitions made in the Space. They are square regions of size = chunksize \* chunksize, chunksize = radius. Each chunk consists of a Base Station. To ease the implementation, the base station id and the chunk id have been made same.

**Parameters used:**

1. num: Gives the chunk number.
2. LinkedList Nodelist : It is a linked list of all the nodes included in the chunk.

**Methods used:**

1. int getnum( ) : returns the chunk number



### **6.1.5. Node**

The node class represents mobile device. Node has position, unique id, network id. Each Node knows about its immediate neighbors and even knows their location. Each Node maintains a list of its neighbor nodes.

#### **Methods used:**

1. `int getX()` : It returns the x coordinate of the node
2. `int getY()` : It returns the y coordinate of the node
3. `int getId()` : It returns the Node id.
4. `addConfigurator(Configurator co)` : It sets the `ConfObj = co`
5. `int getchunk()` : It returns the chunk
6. `int getnetid()` : It returns the Node's network id 'nid'

### **6.1.6. Event**

It stands for any event happening in the space. Node creation, Node movement, Record addition, Record retrieval are all events.

#### **Parameters used:**

1. `timestamp`: the time at which an event occurs since the starting of the simulator.

#### **Method used:**

1. `Event()` : It instantiates the Event class
2. `Event(int ts)` : sets the `timestamp = ts`
3. `int gettimestamp()` : It returns the timestamp of the event
4. `settimestamp(int ts)` : It sets the `timestamp = ts`
5. `abstract void happen()` : an abstract method. It will be overridden in all the other classes that inherit the Event class.

### **6.1.6. NodeCreation**

This is an event that creates a new node in the space.

#### **Parameters used:**

1. SimObj : an object of type Simulator
2. n : an object of type node
3. ConfObj : an object of type Configurator
4. x , y : coordinates of the node which has to be created

#### **Methods used:**

1. happen( ) : It will create a node by calling an appropriate constructor of the node class. It then finds out the neighbour list of the node. It then adds node movement event at timestamp = timestamp + mf. Similarly, the AddRecord event and the RetrieveRecord event are also added at timestamp= timestamp + constant. All of these events are added in the event queue of the Simulator by calling the method SimObj.addEvent(event).
2. createFile( ) : A file is created which stores the records of interest belonging to that particular node.

### **6.1.7. NodeMove**

This is an event that relocates a node from one position to another.

#### **Parameters used:**

1. SimObj : an object of type Simulator
2. nodespace : an object of type space
3. n : an object of type node
4. ConfObj : an object of type Configurator

**Method used:**

1. `happen()` : It relocates the node by calling the method `relocate()` with the help of the object `nodespace`. It recomputes node's neighbours and displays them. It then creates a new object of the type `NodeMoveEvent` and adds it to the event queue using the object `SimObj`.

**6.1.8. AddRecordEvent**

This event adds the record in the distributed hash table. This event is fired from the event queue before the `RetrieveRecord` event. The node firing this event is said to select a record from the common text file “`CommonFile.txt`” randomly and store in itself. A record containing name and contact number is selected randomly from the list of records in common text file which is accessible to all the nodes in the space. Name in the record is hashed using SHA-1 hash function. DHT class is used to obtain the hash value. The hashed value of the name returns an integer. Along with the name base station id is also hashed using SHA-1 hash function such that the hashed value of Name and base stations are comparable. The hashed value of Name is compared with the each hashed value of base station. A base station is selected with its hash value equal to or most nearly greater than the hashed value of Name and the record will be stored in the bucket of thus obtained base station as the tuple of `<hash_value of record key, record_key, node_id>`

**Parameters used:**

1. `node` : object of Node Class
2. `buck` : object of Bucket Class
3. `dht` : object of DHT Class

**Methods used:**

1. **`addRecordEvent (Space s, Node n)`**: Constructore, initializes the objects `nodespace` to `s` and `node` to `n`.

2. **void happen()** : Within this function a record is selected from the common text file and hashed. Separate text file is created for every node firing this event where the record obtained from common text file “CommonFile.txt” is stored. Then Name in the record is stored in a bucket obtained with the hashed value of the name. The bucket is obtained using `getBSBucket(String name, String contactNo)`
3. **getBSBucket(String name, String contactNo)**: It takes a name of a record as an input, hashes it calling the function of DHT class, finds the corresponding base station by comparing the hashed value of Name with the hashed value of every base station and stores the tuple `<value, key, node id>` in the bucket of that base station.
4. **int getRecordCount(String filename)**: This function gets the number of records in the filename passed as the parameter. File name of common text file is passed as the parameter
5. **int getRecord(int recdCount)** : This function is used to obtain the random number among the number of records stored in the common text file. The total number of records in common text file is passed as the parameter `recdCount`.

### 6.1.9. RetrieveRecordEvent

It retrieves the existing record from a node (mobile device) and returns it to the mobile device requesting it. The routing of the record after retrieving is also simulated in this class. This event is fired after `AddRecordEvent` in event queue. The node firing this event firstly selects any of the names from the common text file “RetrieveFile.txt”. In this file only the name of the records that are in “CommonFile.txt” are stored. To get the name from the “RetrieveFile.txt”, **int getRecordCount(String filename)** and **int getRecord(int recdCount)** of `AddRecordEvent` is used. Thus retrieved name is hashed using the hash function from DHT class and compared with the hashed value of base station to find the bucket where the record is stored. Then, the record is retrieved from the bucket file and routed to the node firing this event. The record routed is then stored in the text file of the node.

**Parameter used:**

1. node: object of Node class
2. space : object of Space class

**Methods used:**

1. RetrieveRecordEvent(Space nodespace, Node n) : Constructore of the class and initializes s to nodespace and node n
2. routeString(String strToRoute, int destBS): It takes destination basestation and string to be routed (i.e. the requested record) as an argument, then routes the string to the destination base station using the routing algorithm designed.
3. writeInDestNode(String strToRoute) : It takes the string to be routed (requested record) as an argument and writes it in the text file of the destination mobile node (the requesting mobile device).
4. int getNodeSource(int bs,String recdName): It reads the tuple <name,hashed value , node id > from the bucket file and parses the node id where the requested record is stored. The node id parsed is returned. The bucket id and the name of the record is passed as parameters.
5. String readString(int src, String recd): After obtaining the node id where the record requested is stored, this function searches the record in the text file of the node and returns the record of name and contact number. The node id and the name in the record is passed as arguments.
6. int getBSBucket(String recd): This function hashes the name to be searched using the functions in DHT class and compares the hashed value of the name with the hashed value of the base stations. The base station with its hashed value equal or nearest greater than the hashed value of name contains the bucket where record is stored. The obtained bucket id is returned.
7. void happen() : This function reads “RetrieveFile.txt” and get a name from the file randomly. The record containing this name is searched in DHT using the above functions. Then the record obtained is routed to the node firing this event and is stored in its text file.

### **6.1.10. Bucket Class**

A bucket class keeps the list of tuples <value, key, node\_id> where value is the hash value of a record key, key is the record key and node\_id is the identifier of the mobile device which consists of the actual record of the key. The size of the bucket is not fixed. The newly registered record-tuple is appended at the end of the tuple-list. Each bucket has a file to store this list of tuples.

#### **Parameters used:**

1. bucket\_id : an integer that uniquely identifies a bucket. This value equals to the base station to which the bucket belongs.

#### **Methods used:**

1. void createFile(int bsid): This function takes basestation identifier as an argument. Then it creates a file for the bucket belonging to that base station.
2. void addtoBucket(String recordStr, int buckid): It takes record string to be added in bucket and bucket id as arguments and appends the record string at the end of the file belonging to that bucket.

### **6.1.11. DHT Class**

It hashes a key and produces its corresponding value. For hashing SHA-1 hash function is being used. Functions in DHT are being used while adding as well as retrieving records. SHA-1 hash function being used produces 160-bit string as value for every key. Since the hashed value of the name and the base station is be compared to obtain the bucket for every record, the hashed value of name is converted into integer and every base station identifier is also hashed and hashed value converted to an integer.

#### **Parameters used**

1. name : a string to be hashed
2. bucketed : an integer type data to store bucket identifier

**Methods used:**

1. `int hashKey(String name)` : It takes a string as an argument and hashes it to produce a 16-bit string hash value. This value is returned after being converted to an integer.
2. `int hashBS(int bsID)`: It takes an integer (base station identifier) as an argument and hashes it to produce a 16-bit string hash value. This value is returned after converted to an integer.

**6.1.12. The Base Station(BS) Class**

Base stations are created at the starting of the simulator in Space class. For every chunk one base station is created and chunk size is defined as base station range.

**Parameters used:**

1. `bs_id`: identifier which is equal to its chunk id
2. `bs_range`: its range which is equal to the chunksize
3. `buck`: Object of type bucket
4. `ConfObj`: Objects of Configurator class

**Methods used**

1. `BaseStation(int n, Configurator ConfObj)`: This is the constructor of the class and is called in Space class for creating base station. The base station id and the configurator object is passed as parameters. Within the constructor, `bs_range` is defined to the chunksize and file is created for the bucket of the particular basestation.

**6.1.13. Route Data Class**

Route data class routes the records retrieved from the dht to the destination base station. Firstly, the routing table is traced, if no record found then only the routing direction is obtained then neighbor node is checked and lastly the broadcast message is sent to T-1 node where T is the total number of base stations in the network.

**Parameters used:**

1. dir: routing direction which can be 0 or 1, 0 for anti clockwise direction and 1 for clock wise direction.
2. totalBS : total number of base stations in the network
3. receiverId: The receiver id where the retrieved record is to be routed.
4. senderId: The base station id from where DHT retrieves the record.
5. strToroute: The record retrieved to be routed to the destination.
6. strName : String whose record is being searched.

**Methods used**

1. GetDestination()

This function gets the destination id where the data is to routed which is the same base station searching for data.

2. GetDirection (int senderId)

This function gets the routing direction which is anticlockwise if it returns 0 and clock wise if it returns 1.

3. CheckRoutingTable(int nodeId, int bsId)

This function checks the node id and the base station id in the Routing Table.

4. RouteString(int senderId)

The record being searched is routed to the destination node in this function.



## Chapter 7 : Simulator Testing

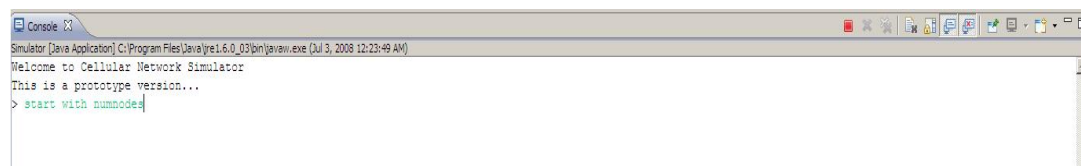
This simulator is discrete event simulation and simulates the functions of Address Book Application.

Simulator starts with creation of space and nodes and base stations on it. Space is divided into number of chunks with specified radius and there is a fixed base station in chunk. Nodes created can move around the space and can go out of range as well. User can define the number of nodes to be created in start of the simulator. The number of chunks in the space is defined and so is the base station numbers. Separate storage in text file is created for every base stations and nodes as bucket file and node file in the space. This file contains the number of Address Book Records of name and address.

Events are added in the simulator in FIFO manner and get executed one after another after specified time stamp. The first event is added in the simulation queue in starting of simulation is Node Creation Event. When this event is executed other events Node Move Event, Adding Record Event, Retrieve Record Event are added simultaneously and simulation of the Routing part is covered in Retrieve Record Event.

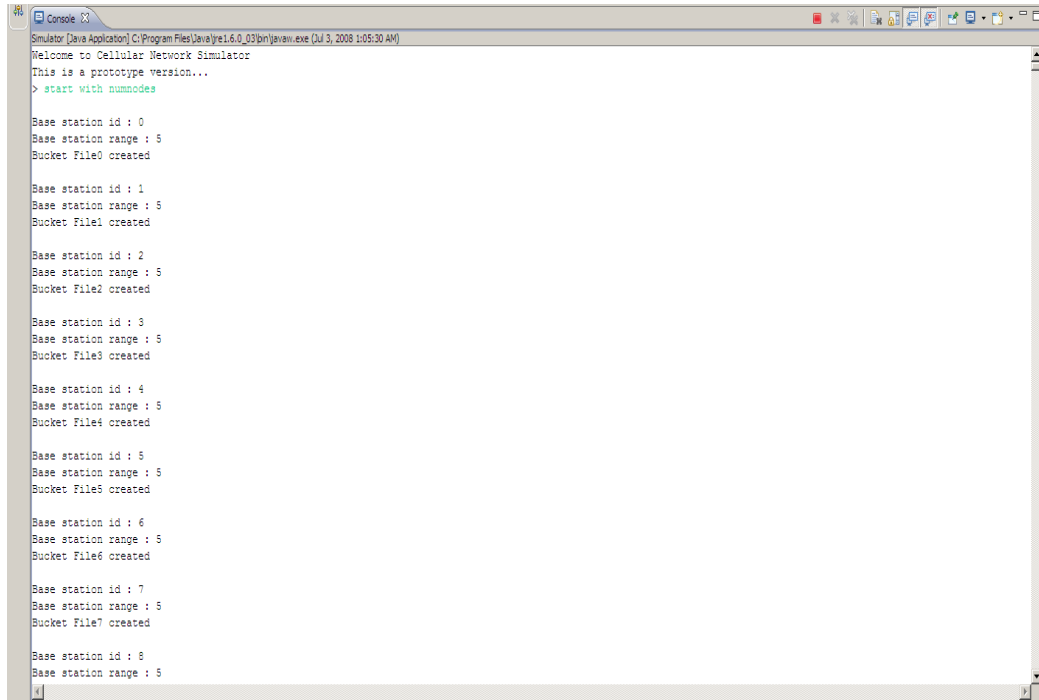
Some of the snapshots of testing of this simulator are as follows:

### 1. Simulation Starting



```
Console
Simulator [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 3, 2008 12:23:49 AM)
Welcome to Cellular Network Simulator
This is a prototype version...
> start with numnodes
```

## 2. Base Station Creation



```
Simulator [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 3, 2008 1:05:30 AM)
Welcome to Cellular Network Simulator
This is a prototype version...
> start with numnodes

Base station id : 0
Base station range : 5
Bucket File0 created

Base station id : 1
Base station range : 5
Bucket File1 created

Base station id : 2
Base station range : 5
Bucket File2 created

Base station id : 3
Base station range : 5
Bucket File3 created

Base station id : 4
Base station range : 5
Bucket File4 created

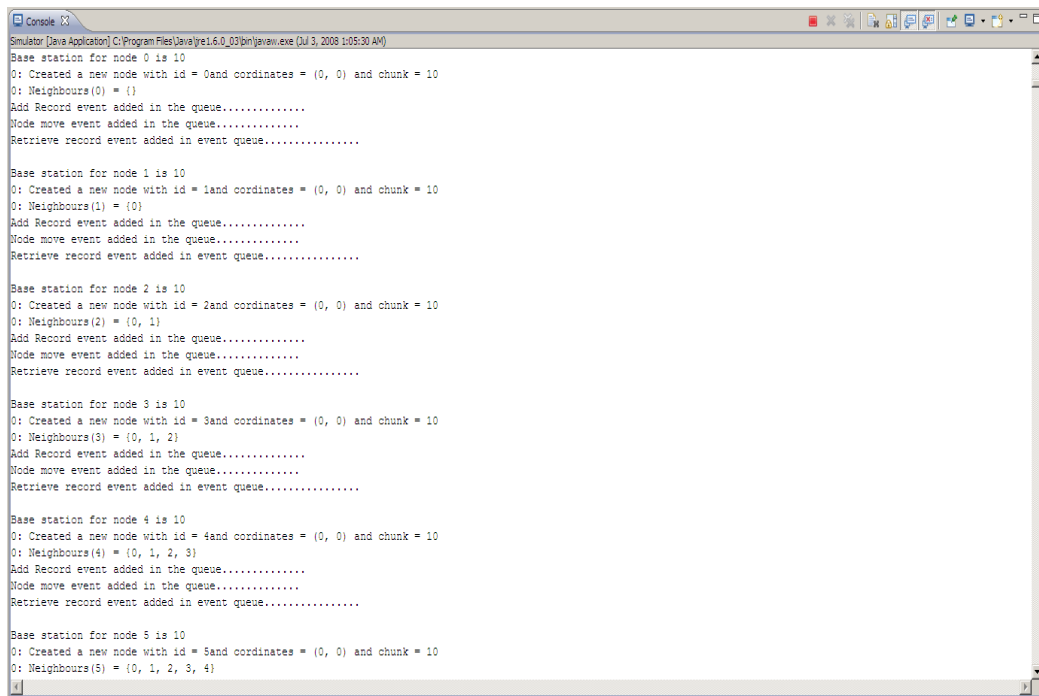
Base station id : 5
Base station range : 5
Bucket File5 created

Base station id : 6
Base station range : 5
Bucket File6 created

Base station id : 7
Base station range : 5
Bucket File7 created

Base station id : 8
Base station range : 5
```

## 3. Node Creations and Adding of Events in Simulator



```
Simulator [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 3, 2008 1:05:30 AM)
Base station for node 0 is 10
0: Created a new node with id = 0 and coordinates = (0, 0) and chunk = 10
0: Neighbours(0) = {}
Add Record event added in the queue.....
Node move event added in the queue.....
Retrieve record event added in event queue.....

Base station for node 1 is 10
0: Created a new node with id = 1 and coordinates = (0, 0) and chunk = 10
0: Neighbours(1) = {0}
Add Record event added in the queue.....
Node move event added in the queue.....
Retrieve record event added in event queue.....

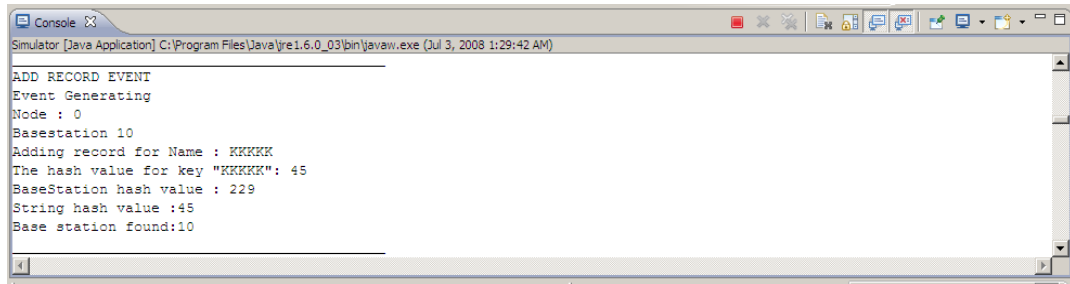
Base station for node 2 is 10
0: Created a new node with id = 2 and coordinates = (0, 0) and chunk = 10
0: Neighbours(2) = {0, 1}
Add Record event added in the queue.....
Node move event added in the queue.....
Retrieve record event added in event queue.....

Base station for node 3 is 10
0: Created a new node with id = 3 and coordinates = (0, 0) and chunk = 10
0: Neighbours(3) = {0, 1, 2}
Add Record event added in the queue.....
Node move event added in the queue.....
Retrieve record event added in event queue.....

Base station for node 4 is 10
0: Created a new node with id = 4 and coordinates = (0, 0) and chunk = 10
0: Neighbours(4) = {0, 1, 2, 3}
Add Record event added in the queue.....
Node move event added in the queue.....
Retrieve record event added in event queue.....

Base station for node 5 is 10
0: Created a new node with id = 5 and coordinates = (0, 0) and chunk = 10
0: Neighbours(5) = {0, 1, 2, 3, 4}
```

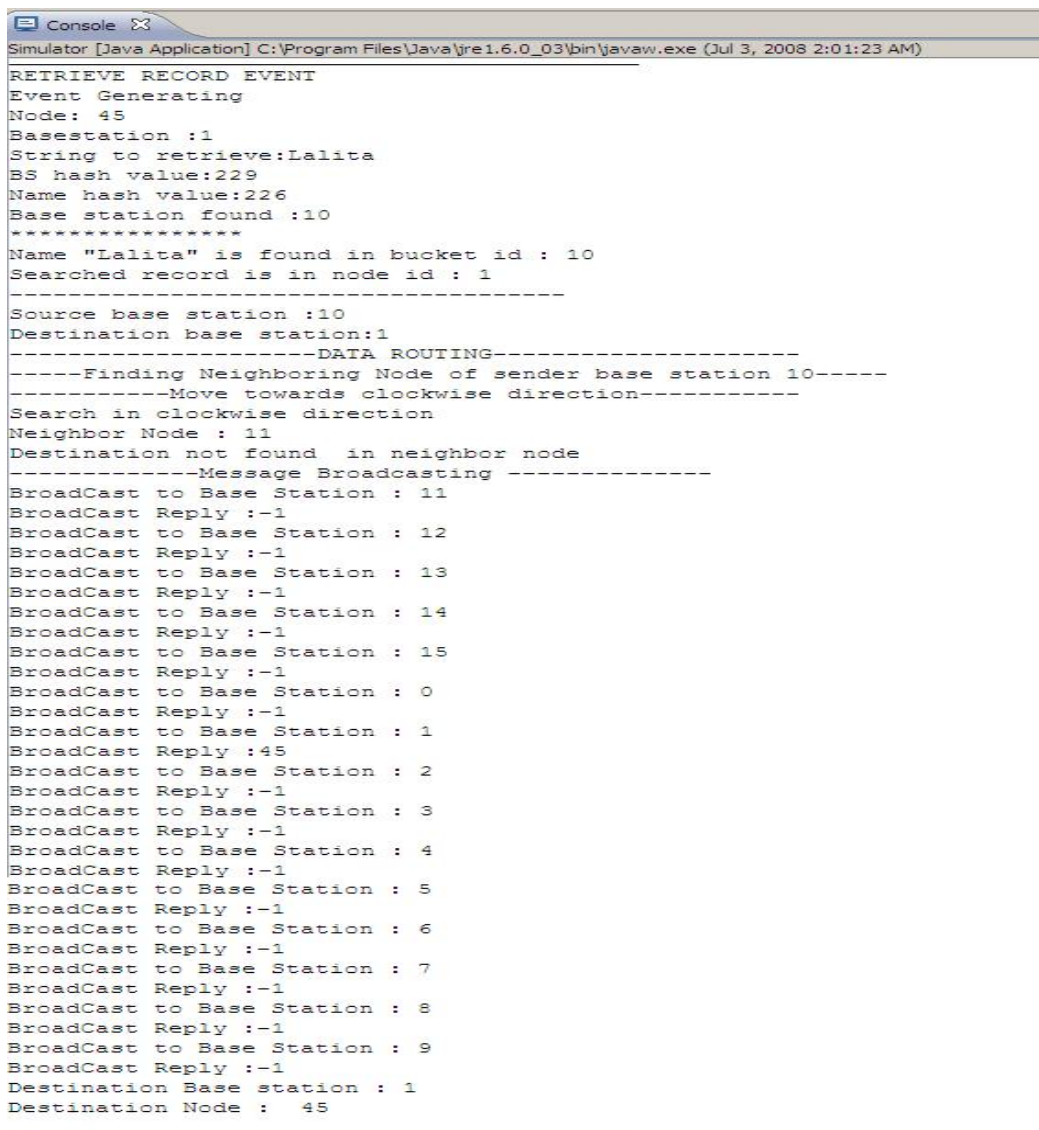
#### 4. Add Record Event



```
Console [X]
Simulator [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 3, 2008 1:29:42 AM)

ADD RECORD EVENT
Event Generating
Node : 0
Basestation 10
Adding record for Name : KKKKK
The hash value for key "KKKKK": 45
BaseStation hash value : 229
String hash value :45
Base station found:10
```

#### 5. Retrieve Record Event



```
Console [X]
Simulator [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 3, 2008 2:01:23 AM)

RETRIEVE RECORD EVENT
Event Generating
Node: 45
Basestation :1
String to retrieve:Lalita
BS hash value:229
Name hash value:226
Base station found :10
*****
Name "Lalita" is found in bucket id : 10
Searched record is in node id : 1
-----
Source base station :10
Destination base station:1
-----DATA ROUTING-----
-----Finding Neighboring Node of sender base station 10-----
-----Move towards clockwise direction-----
Search in clockwise direction
Neighbor Node : 11
Destination not found in neighbor node
-----Message Broadcasting -----
Broadcast to Base Station : 11
Broadcast Reply :-1
Broadcast to Base Station : 12
Broadcast Reply :-1
Broadcast to Base Station : 13
Broadcast Reply :-1
Broadcast to Base Station : 14
Broadcast Reply :-1
Broadcast to Base Station : 15
Broadcast Reply :-1
Broadcast to Base Station : 0
Broadcast Reply :-1
Broadcast to Base Station : 1
Broadcast Reply :45
Broadcast to Base Station : 2
Broadcast Reply :-1
Broadcast to Base Station : 3
Broadcast Reply :-1
Broadcast to Base Station : 4
Broadcast Reply :-1
Broadcast to Base Station : 5
Broadcast Reply :-1
Broadcast to Base Station : 6
Broadcast Reply :-1
Broadcast to Base Station : 7
Broadcast Reply :-1
Broadcast to Base Station : 8
Broadcast Reply :-1
Broadcast to Base Station : 9
Broadcast Reply :-1
Destination Base station : 1
Destination Node : 45
```

## **Chapter 8 : Conclusions and Future works**

This simulator has been designed on top of adhoc network simulator ASSET. Though it tries to cover the cellular network designs, there are lots of cellular network features missing in this dissertation. Current trend in overlay networks do not intend to eliminate IP usage. This dissertation covers the simulation of routing done over DHT overlay network with limited number of base station and only one mobile switching center. In the routing process, after the record is retrieved from DHT, the destination node is obtained by means of broadcasting. This eliminates the overhead of routing through every node connected in the network and finding the destination node when the destination node is at the end of the network. As well as, while adding record in the bucket, the hashed value of key is compared with every base station one by one sequentially, this can also be overhead if the bucket to be used is at the end of the sequence.

In this dissertation, only one Mobile switching center has been considered but communication between two and more Mobile switching center has not been considered. These cellular network features are not covered in this dissertation and yet to be covered.

## Chapter 9 : References

- [1] Sanket Patil, Srinath Srinivasa. A Data centric Abstraction Middleware for Mobile Networks. International Institute of Information Technology – Bangalore, India, 2006.
- [2] Frank Dabek. A Distributed Hash Table. Massachusetts Institute of Technology. September 2005.
- [3] Wikipedia, the free encyclopedia, SHA hash function.
- [4] Thomas J. Schriber, Daniel T. Brunner. INSIDE DISCRETE-EVENT SIMULATION SOFTWARE: HOW IT WORKS AND WHY IT MATTERS. Computer and Information Systems. The University of Michigan. Ann Arbor, Michigan 48109-1234, U.S.A. Proceedings of the 1997 Winter Simulation Conference.
- [5] Wikipedia, the free encyclopedia, Overlay Network.
- [6] Ion Stoica, Robert Morris, David Karger, M.Frans Kaashoek, Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Services for Internet Applications. August 2001.
- [7] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. Microsoft Research Ltd., 7 J J Thomson Avenue, Cambridge, CB3 0FB, UK. Rice University, 6100 Main Street, MS 132, Houston, TX 77005-1892, USA. December 2002.
- [8] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. November 2001.
- [9] How overlay networks will make IP irrelevant without actually killing it  
<<http://www.arl.wustl.edu/~jst/reInventTheNet/?p=72> >.
- [10] Pastry (DHT) <[http://en.wikipedia.org/wiki/Pastry\\_%28DHT%29](http://en.wikipedia.org/wiki/Pastry_%28DHT%29)>.

- [11] Tapestry (DHT)  
<[http://en.wikipedia.org/wiki/Tapestry\\_%28DHT%29](http://en.wikipedia.org/wiki/Tapestry_%28DHT%29)>.
- [12] Ozgur D. Sahin Divyakant Agrawal Amr El Abbadi. Techniques for Efficient Routing and Load Balancing in Content-Addressable Networks.
- [13] Nicholas J.A. Harvey, Michael B. Jones., Stefan Saroiu, Marvin Theimer, Alec Wolman. SkipNet: A Scalable Overlay Network with Practical Locality Properties. Department of Computer Science and Engineering, University of Washington, Seattle, WA.
- [14] AODV < [www.answers.com](http://www.answers.com)>.
- [15] Ian D. Chakeres, Elizabeth M. Belding-Royer, Dept. of Electrical & Computer Engineering, Dept. of Computer Science University of California, Santa Barbara. AODV Routing Protocol Implementation Design.
- [16] Charles E. Perkins, Elizabeth Belding-Royer. AODV Next Generation (AODVng) 2002 Workshop, In Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc) 2002. June 8, 2002.
- [17] C. Perkins, E. Belding-Royer, S. Das Ad hoc On-Demand Distance Vector (AODV) Routing. University of California, Santa Barbara, University of Cincinnati. July 2003.
- [18] David B. Johnson David A. Maltz Josh Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. Computer Science Department Carnegie Mellon University.
- [19] Yinghui Wu, Ming Li, Weimin Zheng. ONSP: Parallel Overlay Network Simulation Platform.
- [20] Sanket Patil, Srinath Srinivasa, Shrisha Rao. Distributed Garbage Detection of Replicated Objects in Mobile Networks. International Institute of Information Technology – Bangalore, India. May 29, 2006.

- [21] Sanket Patil. Localization and Garbage Collection in Ad Hoc Networks. Master's Thesis IIT-b, India. June 2005.
- [22] Himabindu Pucha, Saumitra M. Das and Y. Charlie Hu. Poster. How to Implement DHT in Mobile Ad Hoc Networks? August 2004.
- [23] Asset Documentation.
- [24] Walters, L.O. & Kritzinger. Cellular Networks: Past, Present, and Future. University of Cape Town, Department of Computer Science. November 2000.
- [25] Scott Shenker. The Data-Centric Revolution in Networking. In Proceedings of the 29th international conference on Very large data bases. ICSI and U. C. Berkeley, Berkeley, CA, USA. 2003.
- [26] Sanket Patil . A Data centric Abstraction Middleware for Collabration over Mobile Networks. International Institute of Information Technology, Bangalore. 2006.
- [27] Vasilis Koudounas, Omar Iqbal. Mobile Computing: past, present and future, [referred 10.10.1999].  
  
<<http://www.dse.doc.ic.ac.uk/~nd/surprise96/vol4/vk5/report.html>>.
- [28] Wikipedia, the free encyclopedia, Cellular Network.
- [29] Lourens O Walters, PS Kritzinger, Cellular Network : Past, Present, and Future, November 2000.
- [30] Users Mobility and Call Blocking in Wireless Network.  
  
<<http://people.buoedu/staro/node2.html>>.

## **Chapter 10 : Bibliography**

1. William Stallings. Principles of Cellular Network in Wireless Communication and Networking,. Isbn: 81-203-2386-6, Eastern Economy Edition, Prentice-Hall Pvt.Ltd India.
2. Theodore S. Rappaport. Wireless Communication Principles and Practice. Second Edition. Prentice-Hall Pvt.Ltd India.
- 3 Narsingh Deo. System Simulation with Digital Computer. Isbn: 81-203-0028-9, Prentice-Hall Pvt.Ltd India.
- 4 Goeffrey Gordan. Discrete System Simulation in System Simulation. Isbn: 81-203-0140-4, Second Edition, Prentice-Hall Pvt.Ltd India.