UPPSALA
UNIVERSITET

IT 12 020
Examensarbete 30 hp
Juni 2012

ASSESSING THE VULNERABILITY OF DTN DATA RELAYING SCHEMES TO NODE SELFISHNESS

Balasubramanian Shyam Sundar

Institutionen för informationsteknologi
Department of Information Technology

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

ASSESSING THE VULNERABILITY OF DTN DATA RELAYING SCHEMES TO NODE SELFISHNESS

Balasubramanian Shyam Sundar

The main principle behind the working of delay tolerant networks (DTN) is the mobility of the nodes along with their contact sequences for exchanging data. Nodes which are a part of the DTN network can behave selfishly due to network reservation policy, especially when constrained to energy or storage space. Several forwarding protocols exist for spreading data but our focus is on the performance of popular data relaying protocols namely epidemic routing and two hop routing protocol in a situation where nodes exhibit various degrees of selfishness. Results of an analytical model show the performance advantage of epidemic routing over two hop routing decreases as the number of selfish nodes and intensity of the selfishness increases either deterministically or probabilistically. We practically asses the vulnerability of the above mentioned protocols using ONE simulator. We find that our result coincides with analytical results with some variations in the graph.

Index Terms- DTN, selfish nodes, ONE simulator

Handledare: Christian Rohner
Ämnesgranskare: Lars-Åke Larzon
Examinator: Anders Berglund
IT 12 020
Tryckt av: Reprocentralen ITC

Acknowledgements

I express my sincere gratitude to Christian Rohner who was of great support for me to complete my thesis. He was very helpful and guided me so well that I learnt a lot from this thesis. I am very thankful to him who was very kind enough to answer all my questions. Once again thanks a lot Christian.

I would like to thank Fredrick Bjurefors who also very helpful to assist and teach me some small basic stuffs which are very important for this thesis. Thanks a lot Fredrick.

Special thanks to Lars-Åke Larzon for being the reviewer of this project.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 DELAY TOLERANT NETWORKS

In the field of computer networking, communication of data is mainly ensured by wired and wireless networks. We are going to concentrate on the wireless aspect of data communication. On narrowing down further we could classify the nodes or agents which are responsible for communication to be stationary or mobile. Our focus is going to be on the mobile nodes which move in a specified area. Our worry is what is actually happening between these nodes? How the data is being carried among these mobile nodes?

The mobile nodes have a specified area within which they are allowed to roam around. When these mobile nodes come within the communication range they establish a connection between the nodes. The connection establishment is done with the help of some interface like Bluetooth or Wi-Fi. The established connection is up for a certain time as long as the nodes stay within the communication range. Then the connection goes down as the nodes move away from their ranges. So the transfer of data between the nodes takes place between the connections up time.

This is the overview of the functioning of the mobile nodes with in an area. Hence there are no fixed or permanent connections between any nodes there by connections keep coming and going. It is impossible to predict the network connectivity among the nodes if the nodes are going to be random. Then how is it possible to have a communication in this kind of an environment?

Here comes the concept of Delay Tolerant Networks (DTN) which is the solution for above mentioned scenario. ***Delay-tolerant networking (DTN)*** *is an approach to computer network architecture that seeks to address the technical issues in heterogeneous networks that may lack continuous network connectivity* [13].The basic motto behind delay tolerant networks is without any permanent or fixed connections, communication takes place between the source and destination but may be with a delay in the delivery of messages.

The principle behind the working of DTN is the **store-carry and forward** mechanism. As the name suggests it describes the various stages or the flow in which the message is delivered. The mobile nodes are free to move as long as they have a network to send and receive data there are always chances of the message being delivered to the destination. First the message to be sent to any node has to be stored or buffered. Then carried until it finds some interesting node and then forward the message to that node.

As depicted in the figure 1.1 below, you could see the sender is sitting at some place A and the receiver might be at place B that could be separated by some distance say 100kms for example. The beauty in this mode of communication is there is no fixed path between the sender and the receiver. Person from place A transmits a message which is carried by someone in the bus followed by someone in the traffic which is given to the girl boarding the bus which is given to a cyclist who gives it to a person who is who is jogging which is given to a person sitting in the park who finally delivers the message to the destination. This is possible when all the recipients of the message come into the communication range of one another and have a suitable device for them to send and exchange messages.

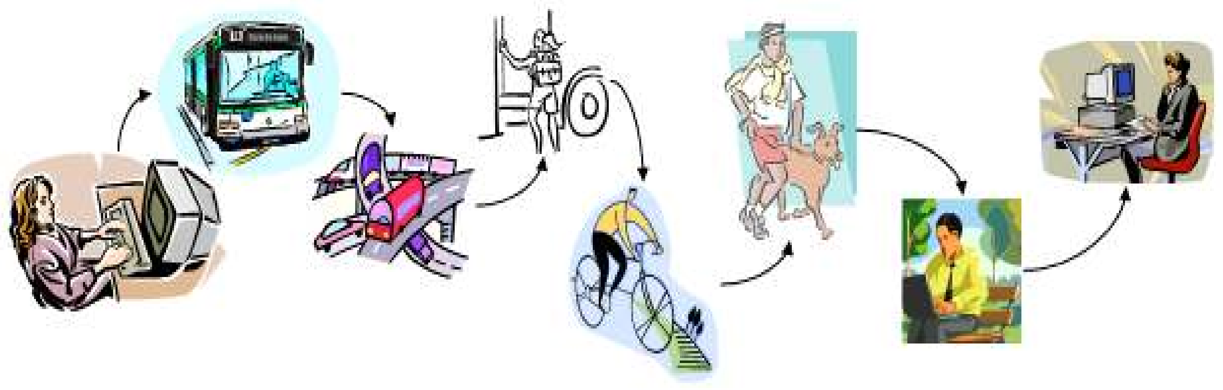

Figure 1.1DTN example [1]

As a consequence of this store-carry and forward mechanism there are few problems that prevail namely,

- Delay
    - Since there is no fixed connectivity and hence messages take time until they reach the destination which accounts for the delay.
- Resource constraints
    - Since all the nodes that are moving carry some limited buffer, it has to drop older messages if the buffer gets full in order to accommodate new messages.

But predicting the connectivity between nodes is possible if the pattern in which the nodes are going to move is known. An example for this kind of a situation is the interplanetary motions [30] in which you could predict when the other planet comes to your desired point. In this case it is possible to make an efficient use on the resource constraints.

1.2 SOME DTN APPLICATIONS:

Where can these kinds DTN approach be useful? Military service is a very good area which could make the best use of this kind of architecture. Since the military camps might be placed in very rough and difficult terrestrial area, it's not possible to have a standard communication at those places. Still we cannot neglect the communication at those places stating the reason it is very hard to get data. So here comes the use of DTN which is very handy to transmit and receive data.

You could see how the transmission and reception of data is being carried out in the figure 1.2 below.

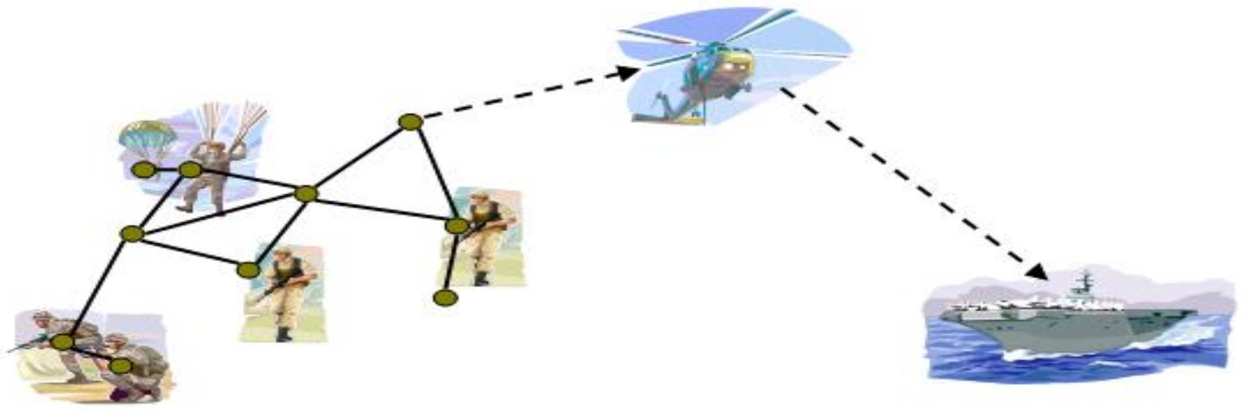

Figure 1.2DTN application [1]

One more application that could be mentioned is about the Wildlife Monitoring: ZebraNet and Swim [1]. In these applications they try to monitor the behavior of the wildlife along with their interactions with the nature and also the reaction to the changes caused by the humans in their ecosystem. The animals carry a tag and there are several base stations to collect the data from these tags and finally the data is sent from the base stations to the destination processing center. The base station can be either fixed or also mobile depending upon the needs. What exactly happens here is an animal carries data of its own along with information of other animals which it has encountered and finally it is collected at the base stations for further analysis. These are some of the applications that are being done with DTN as the back bone or the main theme which is used to work upon these ideas.

1.3 STORE - CARRY AND FORWARD:

The mobile ad hoc networks also known as the MANET's is known for the node mobility and lack of infrastructure. But there has been several discussions going on whether this kind of mobility is to be possible or impossible for DTN [2]. It could be possible only under the assumption that they still have path through the network which is not always the case in DTN.

The existing protocols used in MANET's namely Dynamic source routing (DSR), Ad hoc On-Demand Distance Vector (AODV), Optimized Link State Routing Protocol (OLSR) and Zone routing protocol (ZRP). These protocols are based upon that the nodes are connected and have a connected graph of the network. But the node mobility is considered to be a side issue and recovered using some recovery schemes. If there is any node movement and thereby causing a connection break or loss, it is fixed by route rediscovery or some reactive approach to find a way. They try to establish a complete route and then try to send the data. If the instantaneous end to end paths come up, then it is difficult to establish complete route and hence the above mentioned protocols fail.

The nodes in DTN are capable of having instantaneous end to end paths, highly mobile and frequently disconnected nodes. Therefore we introduce the concept of **store-carry and forward** which could handle conditions mentioned above. The store carry forward paradigm is one where there is a mobile node also called as a ferry which carries the messages from the source, then travels around by sending it to the intermediate nodes or the destination. This model supports the connectivity of the disconnected nodes through virtual connectivity node movement [2].

Store Carry Forward mechanism guides the mobile nodes which could exchange the data between the other nodes. The data is moved in an incremental approach from the source until it hopefully reaches the destination[15][16][17].The store- carry and forward mechanism buffers the messages until the next hop or the link is established to the next node or destination. This is also called as 'mobility assisted routing' since the mobility of the nodes are exploited for reaching the destination. This store-carry-forward approach is the main backbone behind the delivery of data in intermittent connected networks.

Figure 1.3 shows the diagrammatic working of the store carry and forward mechanism for a particular node. The buffer in the node can store the messages that it copies from other nodes. It carries the message in its buffer until it finds another node to which it forwards the message.

Copy:

Copying in networking term means when a node accepts some data or message from another node it is termed as copying. The node that accepts the message could also check for certain properties whether to copy the data or not. For example the properties could be hop count or message id etc. Just to ensure that it does not copy unnecessary data which is of no use to it.

Forwarding:

Forwarding is the process of sending the data or the message from one node to another. Similar to copying, some conditions could be checked at the forwarding node to prevent the data being sent to some nodes that is of no interest to it.

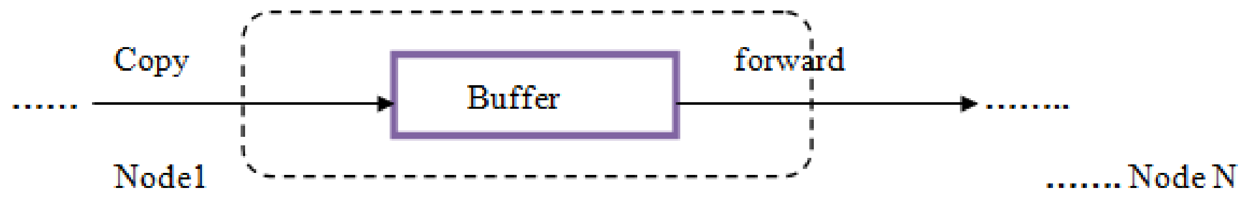

Figure 1.3 Store carry forward

Data relaying schemes

The routing schemes for the DTN are based on the node mobility for delivery of the message and are also categorized on the single or multi-copy algorithms depending whether they are allowed to replicate the messages in the network [5].

Epidemic Routing (Unrestricted Relay)

Epidemic routing[1] *is flooding-based in nature, as nodes continuously replicate and transmit messages to newly discovered contacts that do not already possess a copy of the message. In the most simple case, epidemic routing is flooding;[14]*

Pros:

Minimum spreading delays: Since all nodes can have the data generated by other nodes.

High probability of delivery: Due to the reason that any node can have data generated by other nodes and if the destination comes into contact to any node it gets delivered.

Cons:

Maximal resource consumption: Due to the fact that lot of data is being hold by all the nodes.

Lot of energy wasted: Since all the nodes need to have their Bluetooth or wireless on for the sake of reception and transmission.

Two Hop Routing

According to the two-hop relaying algorithm, [11], [12], the source is allowed to spread up to a maximum number of two copies within the network. Each time it encounters some other node with no copy of the message, it gives it one until it has only one copy (for the destination node only). The intermediate nodes are not allowed to spread the message copy they may have to any other node than the destination [5].

Pros:

Minimum resource consumption compared to epidemic routing: Since all the nodes cannot receive data from all the other nodes due to hoping constraints so the resource used by each node for storing data is less compared to epidemic scenario.

Cons:

Increased delay in delivery when compared to epidemic routing: due to the fact that the node has to wait for the destination to come into the communication range to deliver the data and cannot pass it on to any other node due to the hop constraint.

Binary Spray and Wait:

Every node gives half of its message copies to every node with no copy it encounters until it has only one copy (to give it to the destination node). Binary spray and wait is faster than the two-hop relaying algorithm and induces no more transmissions than those of the latter [5].

1.4 NODE SELFISHNESS

Selfishness can be termed as a node that doesn't perform its duty. *Selfishness in our context can be expressed in two ways. Firstly nodes may deny copying and storing data, which are of no interest to them and destined to a third node. Secondly even if they accept to acquire such data, they may refuse to infect another node with them, i.e. relay data to other nodes* [9].

As mentioned above there is a degree of selfishness in the selfish nodes which are termed as:

- P_{nc} : the probability with which the node does not copy.
- P_{nf} : the probability with which the node does not forward.

The main reasons behind this node selfishness are,

- network reservation policy
- constrained to energy or storage space
-

1.5 PROBLEM CONCERNED

There are many works that are done in the area of delay tolerant network (DTN) data relaying schemes by taking the node selfishness into account[4][5][10][16]. Most of them either considers that all nodes are selfish that is none of the nodes cooperate in the network or all the nodes are willing to cooperate in the content dissemination process.

There is an important situation that has to be considered which is prone to take place at many circumstances. The nodes in the network can exhibit various degrees of selfishness when it comes to the forwarding and receiving of data. That too especially when operating under specific storage resources and energy constraints [4].

Our main focus is on assessing the vulnerability of the data relaying schemes to the node selfishness. So the main problem that we are going to focus on is,

"How is the performance of Epidemic routing and Two Hop routing protocols affected due to the node selfish behavior? Which routing mechanism is better when there are selfish nodes in the network?"

1.6 Goals

The results from [4] tells that the performance advantage of epidemic routing over the two hop routing protocol decreases both with the number of selfish nodes and their intensity increases probabilistically or deterministically [4].

The main goal of this thesis is to experimentally validate the results that were achieved by the analytical model (statistical model based on Markov chain rule) [4] and check if both the results are matching. If the number of selfish nodes along with their probabilistic selfishness in a network increases under the same conditions used in [4], do we get the similar results as achieved in [4]?

1.7 Thesis Structure

The remaining part of the thesis is structured as follows:

Chapter 2 introduces the analytical model and the results achieved using that model. It also provides some more related work that is being done in this field and closely related with our thesis. Chapter 3 introduces ONE simulator along with implementations that were carried out in this thesis. Chapter 4 provides the picture of the mobility model used and some contact properties of the nodes. Chapter 5 provides the practical model along with parameter settings used in the analytical and practical model and evaluation of experimental results. Chapter 6 gives some conclusions and discussions as well as some possible future works.

2 RELATED WORK

2.1 ANALYTICAL MODEL

The model proposed in [4] has been validated in this thesis which assesses the vulnerability of the data relaying techniques of the two protocols namely the two hop routing protocol and epidemic routing protocol along with node selfishness. Node selfishness means, that the nodes do not copy or store the data which is intended for a third node. If at all they copy the data they don't relay the data to other node. Varying degrees of selfishness is also exhibited by these nodes. So the performance of these two routing protocols is assessed analytically using a Continuous Time Markov Chain (CTMC). The selfish nodes are modeled as an absorbing two-dimensional Continuous Time Markov Chain from which the expected delay is derived. The same model is used to find the delay when there are no selfish nodes.

From total of 'N-1' relay nodes, 'K' nodes are considered to be selfish and the progress of data transfer is considered to be a two dimensional pure birth process $(n(t),k(t))_{t>=0}$ for the number of infected and selfish relay nodes respectively at time 't'.

These are absorbing Continuous Time Markov Chains (CTMCs) with a finite number of transient

States W and one absorbing state D, denoting infection of the destination. *The generator matrix Q for both chains could be written in the general form:*

$$Q = \begin{pmatrix} 0 & 0 \\ R & T \end{pmatrix}$$

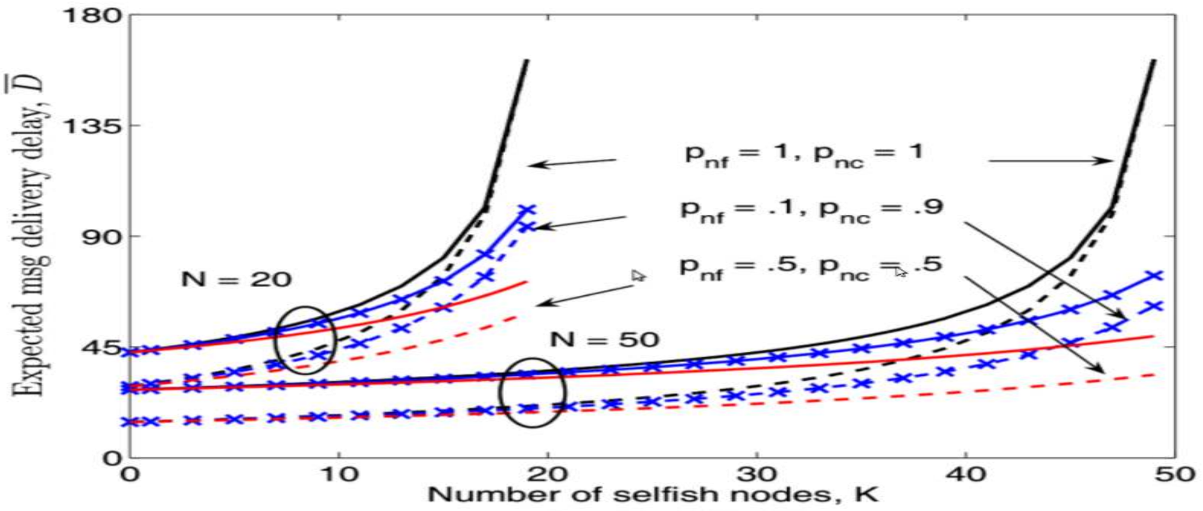

Figure 2.1 [4]: Unrestricted (dashed lines) versus two hop (solid lines) relay schemes for three (p_{nf}, p_{nc}) value sets and N∊ {20, 50}

Results from [4] states that *despite continuing transferring data faster in absolute terms, the performance of unrestricted relay protocol deteriorates faster than the two hop scheme for the same node selfishness intensity. Both of them feature inherent resilience to node selfishness in that their performance deteriorates slightly as long as even a small percentage of nodes cooperate fully [4].* The achieved graph from the paper is given above in figure 2.1

The values in the Y axis represent the delay in 'minutes' and the values in X axis represent number of selfish nodes. The solid lines belong to two hop router and the dashed lines belong to epidemic router. Also the color among the 3 lines tells that they correspond to the respective probability value sets. The values for the coordinate (0,0) in the graph tells us that they were 20 and 50 normal nodes and no selfish nodes. So the values in Y axis when the X coordinate is 0 represents the actual delays of two hop and epidemic router for 20 and 50 nodes with 0 selfish nodes. The graph actually has few things to be noticed or looked upon. They are:

- Comparison of epidemic and two hop routing for 20 nodes,
- Comparison of epidemic and two hop for 50 nodes.

As the number of selfish nodes keeps increasing in the experiment you could see that the delay also keeps increasing. We could see that there are three different curves for each router having its own probability values for copying and forwarding. As these values P_{nc} , P_{nf} varies the delays also vary for both the routing protocols.

Especially when they consider both $P_{nc} = P_{nf} = 1.0$ they delay increases steadily. The reason is the selfish nodes with probability=1 does not copy or forward anything in the network. Also as the number of selfish nodes increases the delay also keeps on increasing.

2.2 MORE RELATED WORK

*Delay-Tolerant Networks (DTNs) have the potential to interconnect devices in regions that current networking technology cannot reach[17].*It mainly makes the communication between the "challenged networks" like sensor networks, mobile ad-hoc networks, deep space networks and low cost networks. There has been a lot of work that has been done in the area of DTN to make it usable for various applications. In [1] it is explained how data is spread in an opportunistic fashion with and without of infrastructure depicted in figure 2.2.

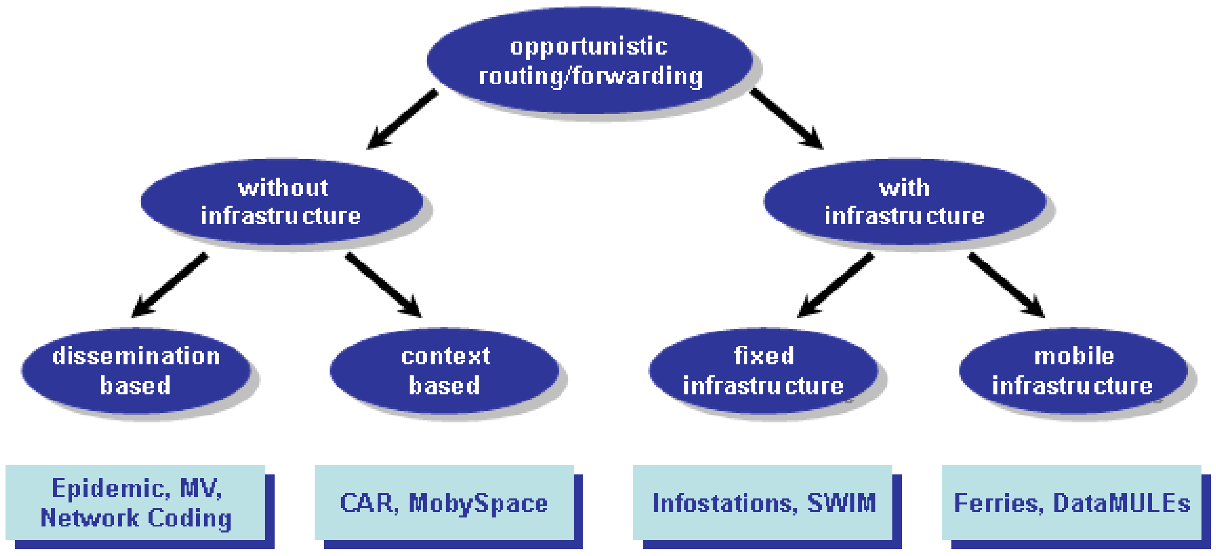

Figure 2.2 Opportunistic forwarding [1]

The key point in [24] [25] [26] is concerned about the routing strategies that would best suit the DTN. In the recent years there has been many forwarding and routing algorithms that have been proposed to improve the performance of DTN. Quite a few numbers of approaches has been proposed in order to reduce the delay [25] [26]. Epidemic routing [17] which is a flooding based protocol that keeps replicating the messages in every contact. Shorter delay could be achieved only at cost of expensive resources which is a tradeoff. Message replication is often used to have a higher probability for the message getting delivered. The big advantage in this method is the information about the network is not at all required and it achieves high delivery ratio [24]. Epidemic algorithms was first proposed for synchronizing replicated databases [28].*Vahdat and Becker applied these algorithms to forwarding data in a DTN [17]*. Epidemic routing guarantees that provided there are a sufficient number of random exchanges of messages, the destination is guaranteed to get the data.

Cooperation among the nodes is required for the delivery of messages in a DTN scheme. Lack of cooperation among the nodes termed as node selfishness results in degrading the performance of the network. In broader perspective performance degradation corresponds to decreased delivery ratio, increase in latency and transmissions as it's discussed in [24]. There are also few works that discovers the selfish nodes in a network by having ranking based schemes. The ranking is given on the basis of forwarding of the messages originated from source. By this kind of a ranking method higher ranked nodes are given more priority than the lower ranked nodes. *This is how—through message prioritization—the scheme converts an abstract quantity, rank, into network operation and physically realizes the difference among nodes originated due to their varying level of cooperation [27].*

3 METHODOLOGY

3.1 OPPORTUNISTIC NETWORK ENVIRONMENT SIMULATOR (ONE SIMULATOR) AN OVERVIEW

Evaluating the various DTN protocols requires some proper tool and the ONE simulator is one such tool. The ONE simulator is mainly used for the modeling of the nodal movement, inter node contacts, message handling and routing.

The ONE simulator is capable of:

- *"Generating node movement using different movement models*
- *Routing messages between nodes with various DTN routing algorithms and sender and receiver types*
- *Visualizing both mobility and message passing in real time in its graphical user interface."**[3]***

Then the collected results and the analysis are done by post processing tools, visualization and the reports [4]. The interactions of the elements are depicted in the figure 3.1

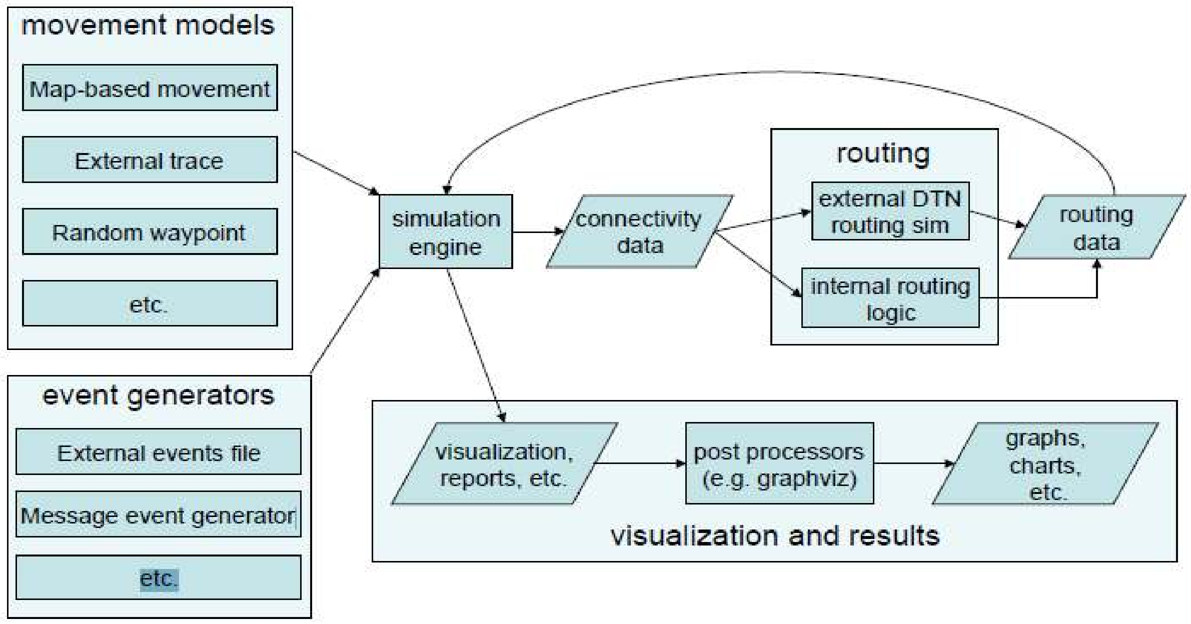

Figure 3.1*: overview of the ONE simulation environment [3]*

The simulator also allows the users to create their scenarios based on the mobility models that are available and making use of the real world traces that are available. The default trace that comes with this package is the map of Helsinki.

“Interactive visualization and post processing tools support evaluating experiments and an emulation mode allows the ONE simulator to become part of a real world DTN test bed”[29].

Once the configuration is set and start running the simulator the GUI screen appears showing all the nodes in the network. It is possible to customize the nodes as per your wish. The customizations could be altering the speed of the nodes, message creation intervals, buffer size etc and so on.

Finally after the simulation gets over then the log reports are collected which could be used for analyzing the problem. Some log files that are logged are created messages report, delivered messages report, statistics of the simulation etc. It is also possible to customize the simulator to behave as per the needs and get the things that is required to perform the analysis.

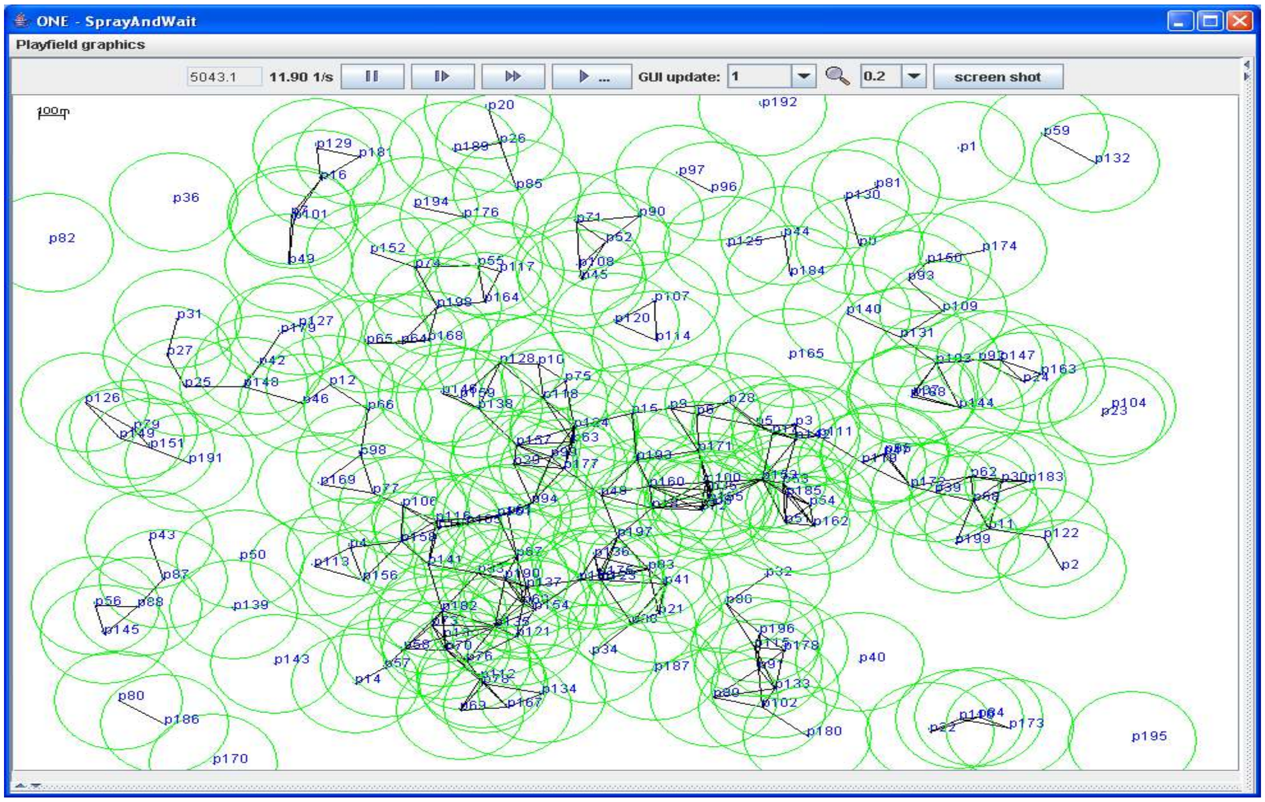

Figure 3.2 GUI interface of ONE simulator

3.2 IMPLEMENTATIONS

The ONE simulator had many things that came with the package by default. But still it was not sufficient to carry on with the thesis. So there were quite a few implementations that were done to the simulator which are listed below:

- Two Hop router
- Epidemic Selfish router
- Two Hop Selfish router
- Some Log reports

Two Hop Router

According to the two-hop relaying algorithm, [11], [12], the source is allowed to spread up to a maximum number of two copies within the network. Each time it encounters some other node with no copy of the message, it gives it one until it has only one copy (for the destination node only). The intermediate nodes are not allowed to spread the message copy they may have to any other node than the destination [5].

The two hop router is a specialized form of epidemic router and was implemented since it was not available with the ONE simulator package.

The code for the ONE simulator was in java. The two hop router was a special case of epidemic router and the reason why we call it a special case is due to the fact that source is allowed to spread just copies of the message in the entire network. If that restriction was absent then it becomes an epidemic router. Hence a clear understanding of the functionalities and the flow of the data in the epidemic router was important to do the customizations. After analyzing the code of epidemic router it had many functions implemented in its super class called Active router. Few important functions from Active router were:

- addToSendingConnections()
    - Adds a connection to sending connections which are monitored in the update.
- canStartTransfer()
    - Makes rudimentary checks that there is at least one message and one connection so this router can start transfer.
- isSending()
    - returns true if this router is currently sending a message with msgId

- receiveMessage()
    - try to start receiving a message from another host.
- requestDeliverableMessage()
    - requests for deliverable message from this router to be sent through a connection.
- startTransfer()
    - Tries to start transfer of message using a connection.

Among these, the function needed for implementation of two hop router was,

- **startTransfer()**

Restricting the number of hops that a router can make to be equal to 2 we could make the router function as a two hop router.

startTransfer()

```
if((m.getHopCount() >= 1) && (m.getTo() !=
con.getOtherNode(this.getHost()))){
      return 10;
}

retVal = con.startTransfer(getHost(), m);
if (retVal == RCV_OK){ // started transfer
      addToSendingConnections(con);
}
```

.

So the above constraint was placed at the startTransfer() function as the check to implement two hop router.

Epidemic Selfish Router

The epidemic selfish router has the same functionality as the normal epidemic router but has to behave selfish. Selfishness was the customization part of epidemic selfish router. The degree of selfishness was given as the input and the router's selfish intensity had to be equal the value that is specified. To make it selfish, the input was got from the configuration file. Random number generator was used to generate random values.

```
If
        The generated random number was greater than or equal to
        the value specified in the configuration file the message
        can be forwarded or copied.
Else
        The message cannot be forwarded or copied to the next node.
```

The random number generator had to generate unique values each time between the range 0 to 1. So the use of "nextDouble" message ensured that.

```
(eg) double r = generator.nextDouble();
```

Input of the selfish values for both no forward and no copy was got and compared with the random number generator and the corresponding action was performed.

```
start transfer()
        if   (probability Noforward > random number genarated)
                exit
```

Note that the above condition was checked under the startTransfer() which was responsible for the deciding the forwarding of messages.

```
checkReceiving()
        if   (probability Nocopy > random number genarated)
                exit
```

Note that the above condition was checked under the checkReceiving() which was responsible for accepting the messages.

If the probability for no copy and no forward is set to 1, then the selfish nodes that are present in the network are not supposed to take part in the forwarding or copying process since they are 100% selfish.

Two Hop Selfish Router

The implementation of the two hop selfish router was the combination of the implementations of,

- the normal two hop router and
- the selfishness taken from the epidemic selfish router

The duty was to perform two hop routing in a selfish manner which customized and the router was made to perform the way that was intended

Log reports

- Calculated delays report
- Clashes report
- Dropped messages report

Calculated delays report

This was the report that was implemented to display the delays of the messages that were delivered at the destination.

Delay:

Delay of a message is the difference in the message's creation time and its delivery time.

So in order to get the delay values for each and every message that were delivered, a log report having the fields namely message id and delay values were generated. This was implemented under the messageTransferred function.

- messageTransferred(Message m, DTNHost from, DTNHost to, boolean firstDelivery)

The above function has the message, source, destination and firstDelivery as the parameters. If the firstDelivery is set to true then we are sure that the message is delivered and we also have the creation time of the message as well as the delivery time. By subtracting both these values we get the delay for the corresponding delivered messages.

Clashes report

This was the report which was implemented to log the number contacts of a particular node makes with the other nodes that comes into contact with during its simulation.

There are interfaces in the ONE simulator namely,

- ConnectionListener
    - Interface for classes that want to be informed about connections between hosts.
- UpdateListener
    - Interface for classes that want to be informed about every single update call

We create a new class to find the nodal contacts between the nodes by having these 2 interfaces as super classes. It was implemented by having an array which was used to collect all the host ids of the nodes that comes into contact with other node. The hostsConnected() was the method is called when two hosts are connected and hostsDisconnected() method is called when connection between hosts is disconnected. There by we could keep track of the nodes that come into contact with each other.

Dropped messages report

This was the report that was implemented to log the number of messages that were dropped along with the properties of the message including time at which it was dropped, message id, source and destination. Since we should not drop messages during our simulation this report was useful to check if any messages are dropped by any node.

Dropping messages or deleted messages both mean the same. So the code to identify the messages being dropped was placed under the messageDeleted().

- messageDeleted(Message m, DTNHost where, boolean dropped)
    - Method is called when a message is deleted

If the Boolean for dropped is set to true we display all the messages with the desired properties that are required to be logged in a file.

3.3 Measurements and Analysis

By running the simulations for different configurations many files were created which needed to be analyzed but the main log file that was of important to us for getting the graph was the ‘delays log report’. There were quite a few log reports mentioned above were implemented in order to do some preliminary analysis and to have a better understanding about the nodal behavior and actual characteristics of the processes that are taking place between the nodes.

AWK

The AWK utility is a data extraction and reporting tool that uses a data-driven scripting language consisting of a set of actions to be taken against textual data (either in files or data streams) for the purpose of producing formatted reports. The language used by awk extensively uses the string data type, associative arrays (that is, arrays indexed by key strings), and regular expressions [20].

Since the log files contained many thousands of records which needed to be analyzed, the help of some good tool was required. 'AWK' was a useful and helpful tool to perform the analysis of the log file which had many records. It was convenient to extract a particular column or columns of values from a log file and put in a separate file for doing the further analysis. It was easy to search for desired patterns of records and select them uniquely from the mixed set of records. In order to do some basic calculations like finding the average, checking the sequence in the arrival of messages was done easily. So several scripts were written to do some manipulation in the log files that were generated and suitable records that were of interest to the experiment were selected carefully.

Some scripts that were written using AWK:

- Calculate the average delay of messages delivered consecutively

The delivered messages reports were used and the messages from that file were read and two fields were important namely the message id and the corresponding delay.

A check was performed whether the messages were received in successive order and if not, we stopped at that point. The delay values for the consecutive messages were taken into account the average for those messages were calculated.

- Plot cumulative distribution function (CDF) for delays of messages delivered consecutively.

This is extension of the previous script that checks if the messages are delivered in order and computes the CDF for the delays.

- Check if both source and destination are proper nodes for delivered messages. Proper nodes mean they are normal and function without any selfish behavior.

Simple scripts to check if the messages have both their source and destination as proper nodes were implemented. This was implemented by checking the source and destination as proper nodes. We take created messages report and delivered message report and take those messages that are common in these two logs and having proper nodes as source and destination.

- Scripts to plot the CCDF graphs for contact pair between two nodes, time difference between successive contacts among all nodes and contact of a particular node by other node were implemented.

GNU Plot

Statistical data is only as valuable as your ability to analyze, interpret, and present it in a meaningful way. Gnuplot is the most widely used program to plot and visualize data for Unix/Linux systems and it is also popular for Windows and the Mac. It's open-source, actively maintained, stable, and mature. It can deal with arbitrarily large data sets and is capable of producing high-quality, publication-ready graphics [21].

Gnuplot is a command-driven interactive function plotting program. It can be used to plot functions and data points in both two- and three-dimensional plots in many different formats. It is designed primarily for the visual display of scientific data. GNU Plot was used to plot the graphs for the collected data [22].

There were many graphs that needed to be plotted throughout the thesis and GNU Plot was useful and convenient to get some quick graphs. The graphs were then used for further analysis and interpretations of the actual happenings in the experiment.

4 MOBILITY MODEL

There should be some realistic patterns that should be incorporated in the simulators to evaluate the performance of some protocol. These realistic patterns are nothing but the mobility models which determine how the nodes should move in an area. *By using mobility models that describe constituent movement, one can explore large systems, producing repeatable results for comparison between alternatives [18].*

A mobility model should attempt to mimic the movements of real MNs. Changes in speed and direction must occur and they must occur in reasonable time slots. For example, we would not want MNs to travel in straight lines at constant speeds throughout the course of the entire simulation because real MNs would not travel in such a restricted manner [19].

4.1 RANDOM WAYPOINT MOBILITY MODEL

This is one of the common mobility models that are used in the area of wireless communications. The way by which this model works is a node chooses a destination in the area and moves to that point with a constant speed on a straight line to this point. Then it waits at that point for a time instance and then does the same manner to a new destination point until the end of the simulation [6]. This is how all the nodes behave in the random waypoint model.

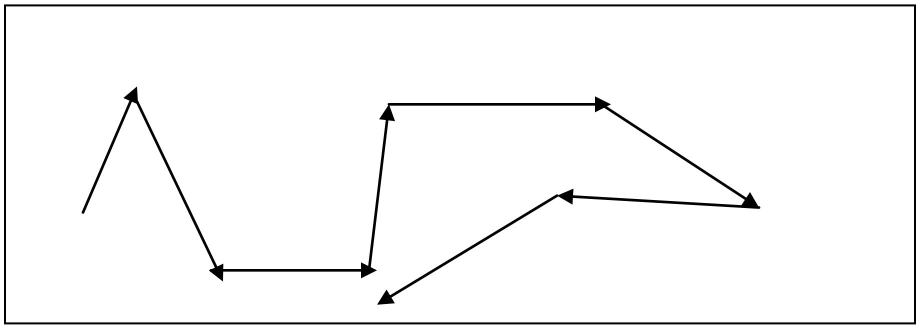

Figure 4.1 Path of a node in Random Waypoint model

The main properties that the random way point mobility model has which are advantageous are [6]:

- No limitation to any specific geometry
- Few parameters and easy to describe
- Analytical results can be derived since it is elementary
- Synthetic (not dependent on any traces)

4.2 RE-CHOICE OF PARAMETERS TO MATCH ANALYTICAL MODEL'S PARAMETERS

ANALYTICAL MODEL'S PARAMETERS

The formula for Contacts per hour (λ) that has been used in the base paper [4] is

$$\boxed{\lambda = C * \frac{v * R}{A}}$$

Where,

- The meeting time epochs of each node pair follow a Poisson distribution of intensity ' λ' giving rise to exponentially distributed intermeeting times between nodes
- The mobility models that are considered are 'random waypoint' or 'random direction' mobility models.
- Communication range of 'R' (meters).
- Area 'A' - area in which the nodes move, the area is considered to be a square in the case of random direction and circle in the case of random waypoint in (kilometers)

The settings values mentioned in [4] are,

C=1 (1.368) constant for random direction (random way point) models respectively,

ν= (0.5, 2.5) meter/second,

R=50 meters,

A= 1000000 metre^2 for square in Random direction or

A= 318528.7 metre^2 for circle in Random waypoint model.

After application of these values to the formula mentioned above the value achieved for '**λ**' is

$$\boxed{\lambda = 0.37 \text{ contacts/hr}}$$

Re-choice of parameters

Since a circle was used where the nodes moved along in a random waypoint model in [4], which was not available in the ONE simulator. Hence we had to do some alterations in order to have the same value for contacts per hour **(λ)**.

So instead of having a circle we had to consider the nodes moving in a square. The graph below in figure 4.2 shows the contacts per hour vs side of square. You could check the graph 4.2, values for contacts per hour **(λ)** started with approximately 7.5 when the side of the square was considered to be approximately 220 meters. Then several simulations were performed to achieve a value of λ=0.37 as mentioned in the paper [4] and the side of the square was fixed to be 1401 meters.

The values for other parameters were the same as used in [4] and no changes were made to it.

- Chosen mobility model was random way point
- C=1.368 constant
- ν=.5,2.5 (meter/second),
- R=50 (meters),
- A is the area which we consider it to be a square of side 1401 meters instead of circle of radius 1000/π meters due to unavailability in the simulator.

We achieved the value of **λ** to be equal to,

$$\boxed{\lambda = 0.37 \text{ contacts/hr}}$$

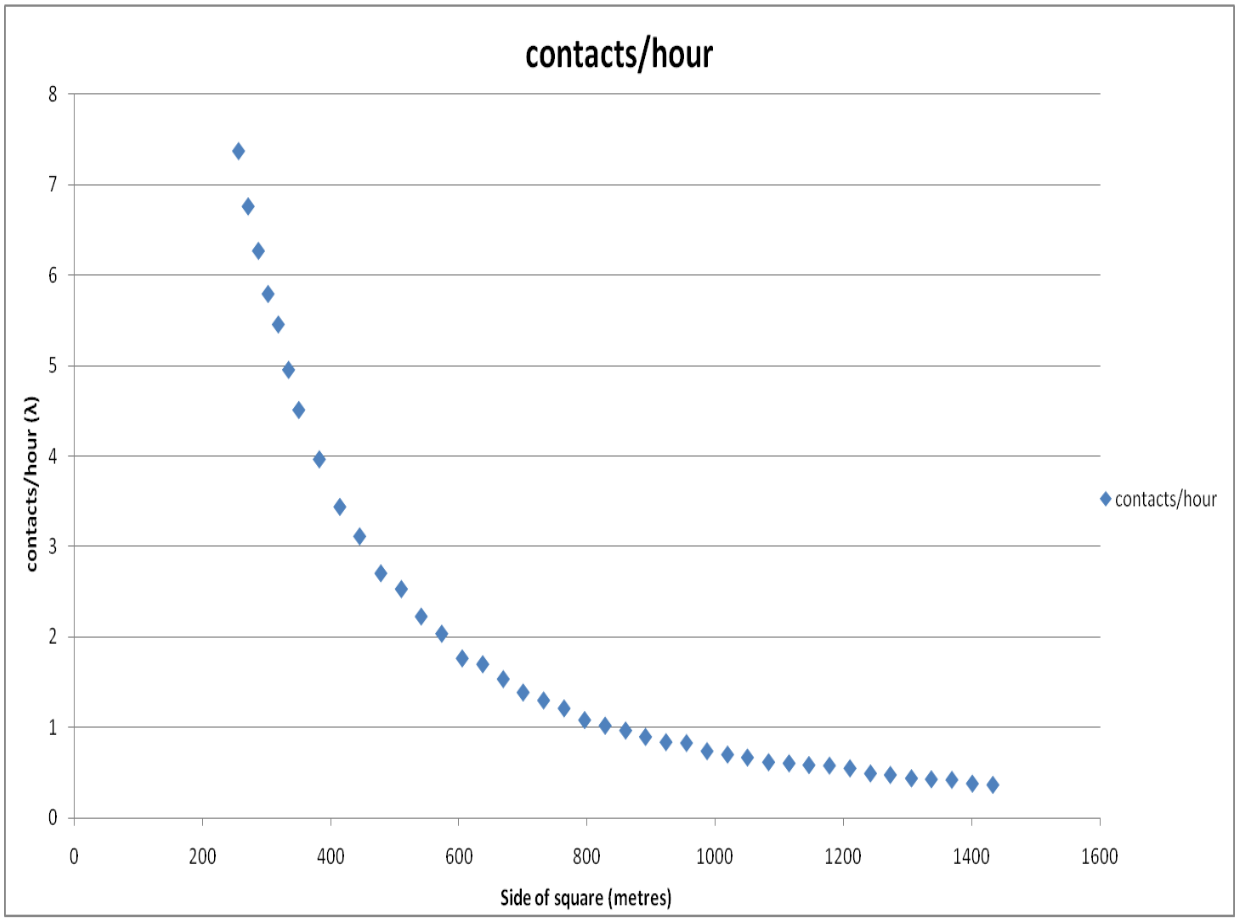

Figure 4.2 Contacts per hour

4.3 DISTRIBUTION OF CONTACTS

The contacts that were happening between the nodes were looked upon carefully. Few graphs were plotted to find the characteristics of,

- Contact between node pairs
- Contact of any node with others
- Time difference between successive contacts among all nodes
- **Contact between node pairs**

The CCDF was plotted to find the contact between the node pairs. The contact between node pairs gives us information about how frequent the node pairs do comes into contact during the simulation period.

Figure 4.3 represents the CCDF obtained for the node pairs. The main idea behind this plot is to find how often the same node pairs have connectivity in the entire simulation. This would give an idea about inter contact of node pairs taking place during the simulation.

First we find the combinations of the 'same node pairs' at different times during the simulation from the connectivity trace. Then calculate the difference in time intervals between the node pairs. The same procedure is carried out for the other node pairs also and finally the delay values are sorted. Then the CCDF graph is plotted with those values. Note that the Y axis in figure 4.3 has a log scale and the value in the X axis represents the different simulation intervals.

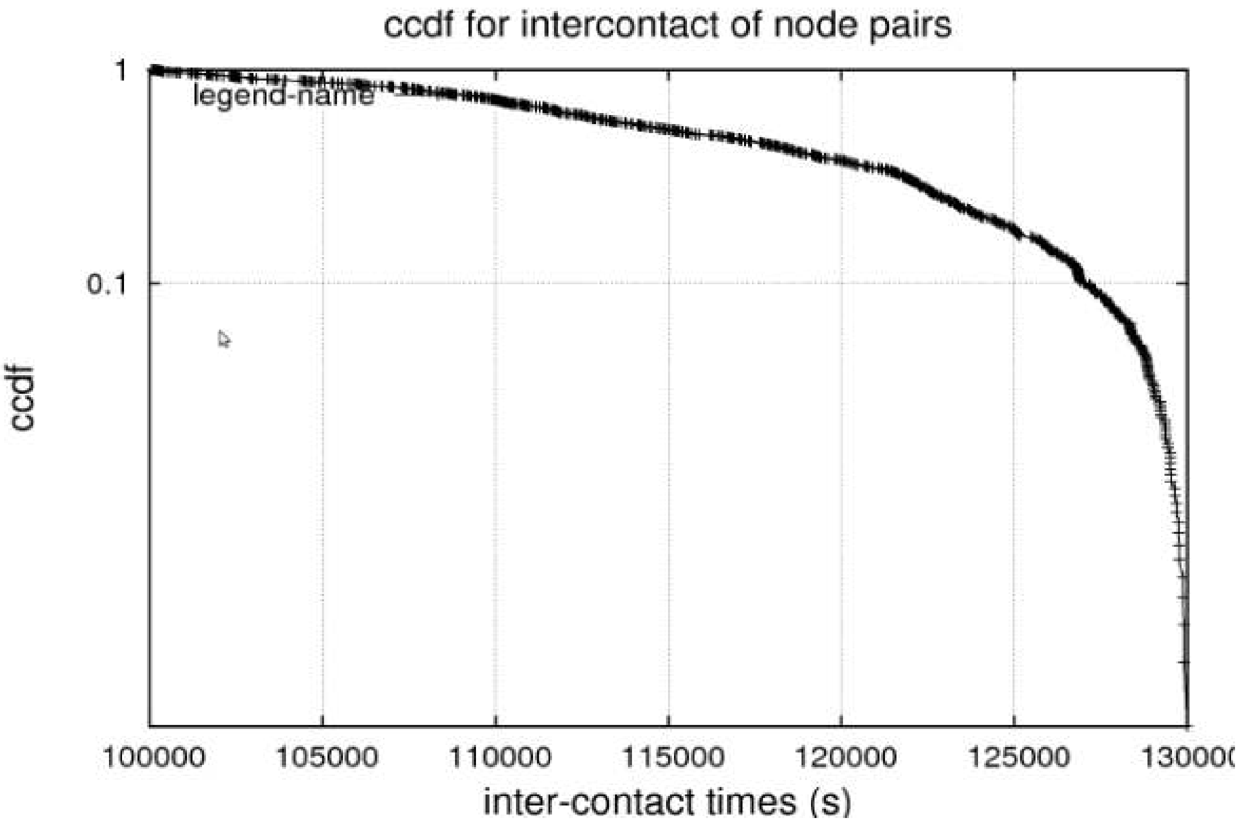

Figure 4.3 CCDF for Node Pair contacts

As a result of this graph we could see that more than 90% of the nodes comes into contact quite often. This result is important for to know about the nodal movement happening in the scenario for copying and forwarding the message.

<u>Contact of any node with others</u>

Contact of any node with a particular node is a useful graph that will make sure that most of the nodes meet that particular node during the simulation, if the simulation is not running too short. This would ensure that the message generated from one particular node has a chance to get delivered at the destination if most of the nodes meet that particular node.

Figure 4.4 represents the CCDF for inter-contact of any node with other nodes. The idea behind this CCDF graph is to get an overview of the contact of any node with other nodes throughout the simulation interval. The time gaps between the contacts of a node with other nodes are calculated. This value is used for plotting the CCDF in which the Y axis has a log scale and the X axis represents the time of the simulation in seconds.

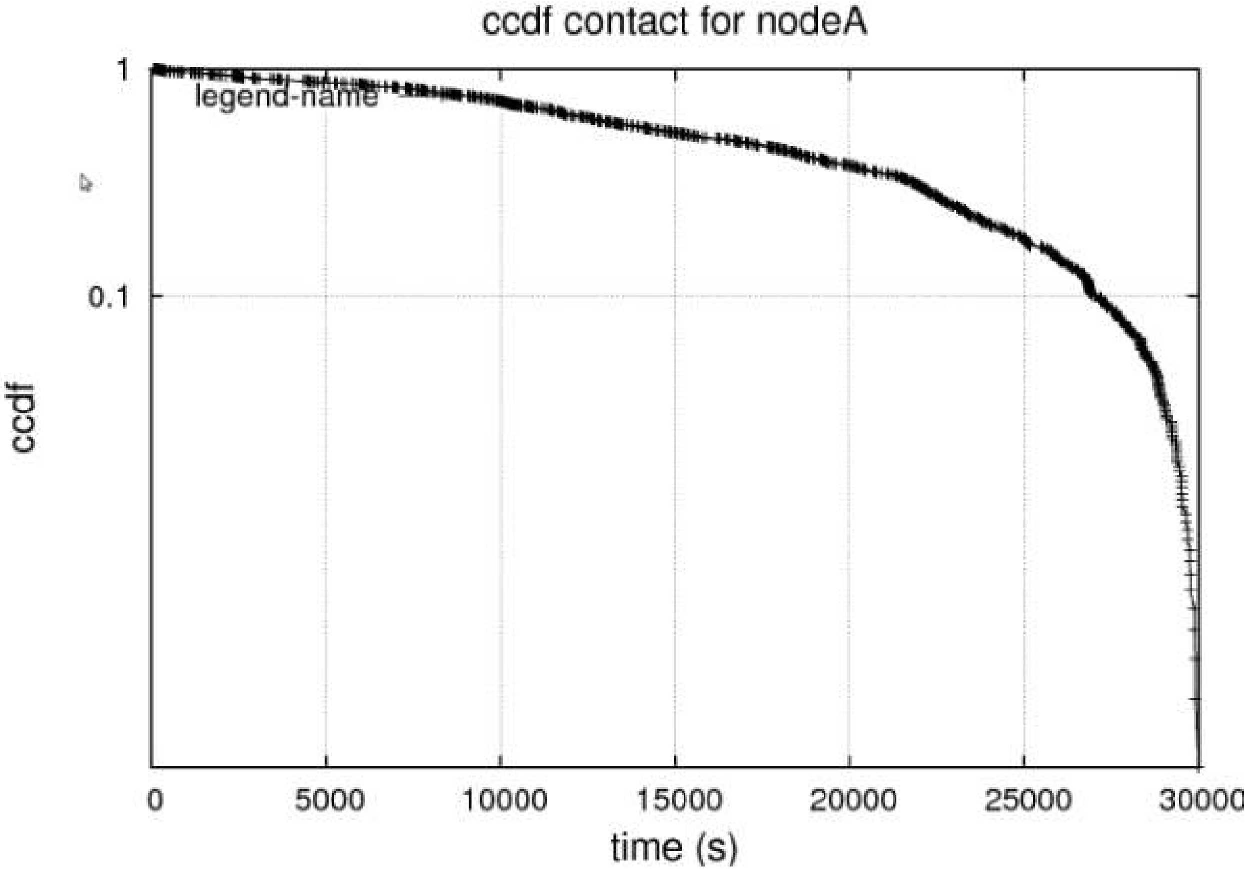

Figure 4.4 CCDF for contact of a node

From the above graph we could see that more than 90% of the nodes come into contact with a particular node approximately around 26,000 sec which is good and ensuring us that the message that is generated by a node is possible to be transmitted to other nodes.

- **<u>Time difference between successive contacts among all nodes</u>**

  The time difference between successive contacts among the nodes shows us how quick the nodes again establish a connection with the other nodes that comes into the contact region. Figure 4.5 represents the CCDF for the difference in the time interval between the connections.

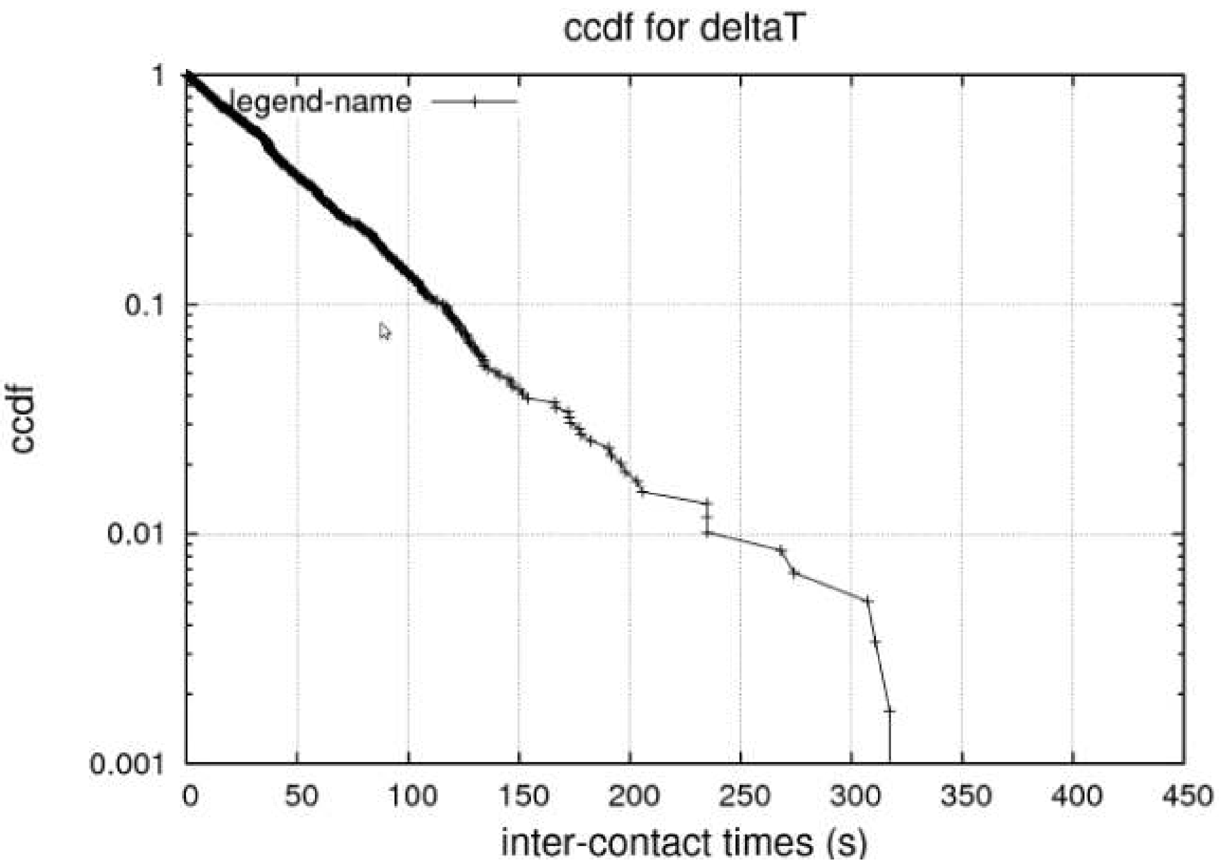

Figure 4.5 CCDF for inter contact times

The reason for us to plot such a CCDF graph is to find the how often there is a connection formation between the nodes. The connection up time indicates the time during which the nodes have their connection set up with other node and ready to transmit the messages among each other. This CCDF graph would indicate how frequent are the connection establishment among the nodes occurs. We consider the up times for all the nodes and the difference between the up times gives the deltaT values. In figure 4.5 the Y axis has a log scale and the X axis represents the time in seconds. For example 96% of the contact times are less than 150 seconds. There by we can infer that there is frequent up connections that are taking place between the nodes moving in the network.

5 VALIDATION OF THE ANALYTICAL MODEL

5.1 SIMULATION SETTINGS

The settings that was taken for the practical verification was,

- Chosen mobility model was random way point
- C=1.368 constant
- ν=.5,2.5 (meter/second),
- R=50 (meters),
- A is the area which we consider it to be a square of side 1401 meters instead of circle of radius 1000/π meters due to unavailability in the simulator.

Few more assumptions that we made in our simulations are:

1. We consider only the messages that were created only from the normal nodes and those messages that were created by the selfish nodes were not taken into account.
2. We also consider only the consecutive delivery of messages into account. Since at the end of the simulation there are many messages delivered, but we take into consideration those messages having consecutive messages id's. If there is a break in the message id in the delivery, we stop at that point and calculate delay for those messages that are delivered consecutively.

The main reason why we had to consider the consecutive delivery of messages were since we were running the simulations for a fixed time and thereby not giving a chance for all the messages to get delivered.

3. We also set a threshold limit in the message id's while analyzing the consecutive deliveries. The threshold limit is a value that is kept as a reference and we calculate the delay until this point in the consecutive delivery of the messages. Thereby in all the configurations we consider the messages to be delivered consecutively and until that threshold limit.
4. We make the buffer capacity and TTL to be a high value so that messages are not dropped.

5.2 SCENARIO

Validation of the analytical model in [4] was practically carried out using the ONE simulator as it was mentioned earlier. Since we had to analyze the working of both epidemic routing as well as two hop routing for the sake of plotting the graphs, it was required to run the simulations separately for each routing protocol. As mentioned in [4] that the delay for both the protocols are calculated only for particular value sets for both the routing mechanisms namely,

- $P_{nc} = P_{nf} = 1.0$
- $P_{nc} = P_{nf} = 0.5$
- $P_{nc} = .9$, $P_{nf} = 0.1$

The ONE simulator package had a configuration file where the values were given as input and taken up for simulation. There were several settings that were available that you could change as per your needs of your simulation. Few examples of those settings that were available were:

- **Scenario and interface settings**

Scenario settings ensures the simulation name, end time and interface settings ensures to specify the transmitting range, speed and the broadcasting interface to be used for communication.

- **Group specific settings**

It ensures to specify the number of hosts, TTL for the messages, router to be used etc for a particular number of nodes.

- **Report settings**

It ensures to specify the specific reports that are required to be logged at the end of a simulation.

Once the configuration file is set, the simulation could be started. There are two ways to carry on with the simulation which are:

- **GUI mode**

Running in the GUI mode shows the nodal movements of the trace in a screen. Just by clicking on the file named one.sh brings the GUI screen up. The speed could also be adjusted which determines at what rate the update has to appear in the screen by selecting the value from a dropdown list box.

- **Batch mode**

The batch mode could be started from the command prompt or from the terminal by giving one.sh –b 1 configuration filename. If no configuration filename is given it takes the default configuration file into account. If you have multiple values that you have given as inputs for certain parameters in your configuration file and want to run all of them at one instant, you replace '1' the number of times you need to run your experiment. This feature is called run indexing which ensures you to run large amounts of different configurations using a single configuration file. Running in batch mode also saves a lot of time.

So each time you start the simulation you tend to change the parameters namely the router and the selfish probability mentioned in [4] along with the address range depending upon the total number of nodes. After the simulation comes to an end you could see different log files that are generated at the end of the simulation. Further analyses were carried out with the help of the generated log files and the AWK scripts.

5.3 DELAY RESULTS

Graphs Obtained:

When total nodes 'N=50'

Figure 5.1 represents the graph that is obtained for the scenario of having up to 50 selfish nodes. The lines starting from the point '35' in Y axis represents the lines for two hop routing protocol and their three probabilities as mentioned in the key above. Likewise the lines starting from '16.4' in Y axis represent the epidemic routing protocol and along with the different probabilities as mentioned in the key above. The graph is achieved by taking the average delay values of 10 different seeds that constituted the random waypoint model. Each seed provided different random motions for the nodes in the network. So in order to have a more accurate result we ran the simulations for 10 different seeds and the average delay was taken into account. On comparing figure 5.1 with figure 2.1 we could see both the graphs have the same starting delays when there were no selfish nodes in the network (ie) the delay of both epidemic routing as well as two hop routing matches with the result of the analytical model.

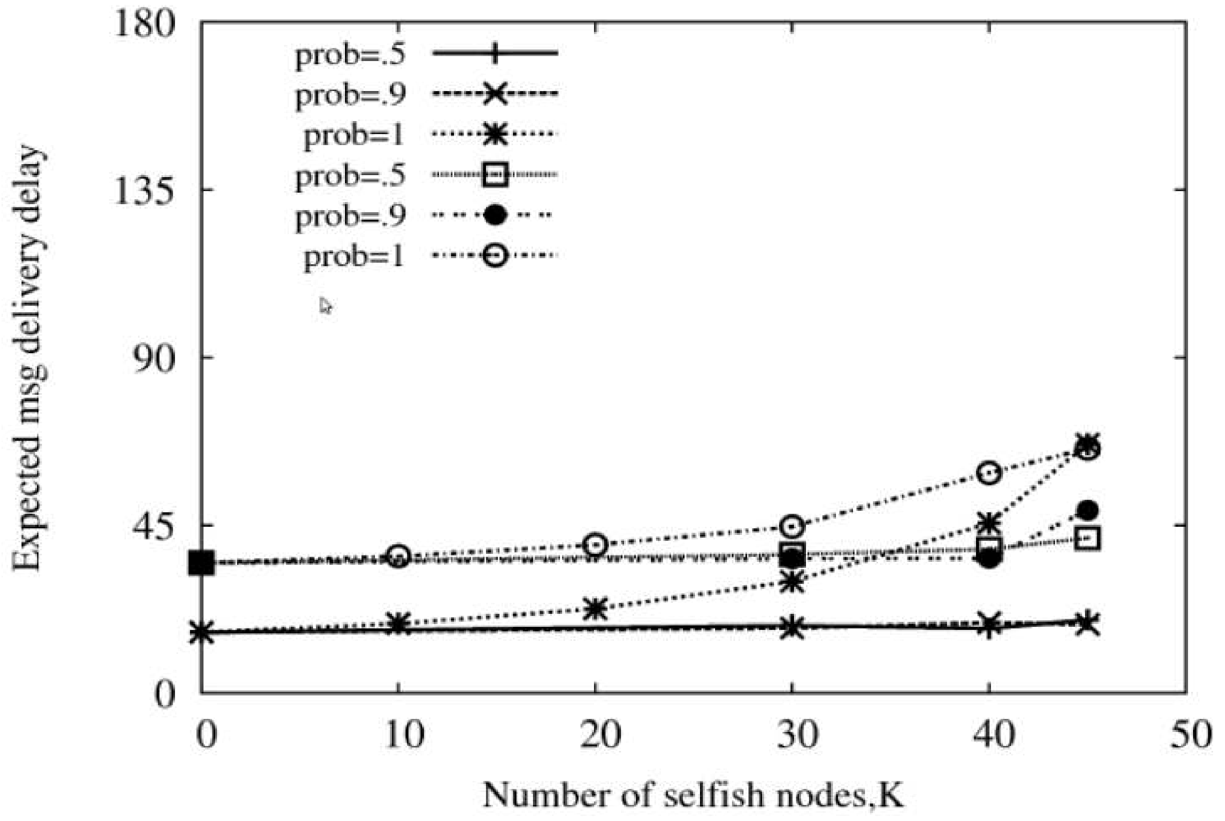

Figure 5.1 50 Nodes delay graph

Further on comparing the delay values as the selfish nodes increases in the network we see that the results for both the routing protocols when $P_{nc}=P_{nf}=.5$ and $P_{nc}=.9,P_{nf}=.1$ is lesser than results of the analytical model. The reason is due to the average of 10 different seed values there is a variation in the delays. At the same time there is the probability generator which also generates random numbers which determines whether the node could copy or forward a message. So there is an impact of both the random number generator and the seed on the results.

When we consider the remaining case which is $P_{nc}=P_{nf}=1$ (ie) the nodes are 100% selfish we could see that both the graphs have a close result with the delay values. As the selfish nodes increases in the network we could see that the delay of epidemic rises and reaches close to the two hop protocol. This is evident when the numbers of selfish nodes are 45, we could see the delay of both the routing schemes to have a very close value.

One contradicting result between figure 2.1 and figure 5.1 is the delay values for epidemic protocol when the probability values $P_{nc}=.9,P_{nf}=.1$ is lesser than $P_{nc}=P_{nf}=.5$. So in order to find out the reason we plotted a graph for Probability Nocopy (P_{nc}) vs delays.

When total nodes 'N=20'

Figure 5.2 was obtained for 20 nodes separately. The lines starting from the point '46' represents the lines for two hop routing protocol and their three probabilities as mentioned in the key above. Likewise the points starting from '27' represent the epidemic routing protocol and along with the different probabilities as mentioned in the key above.

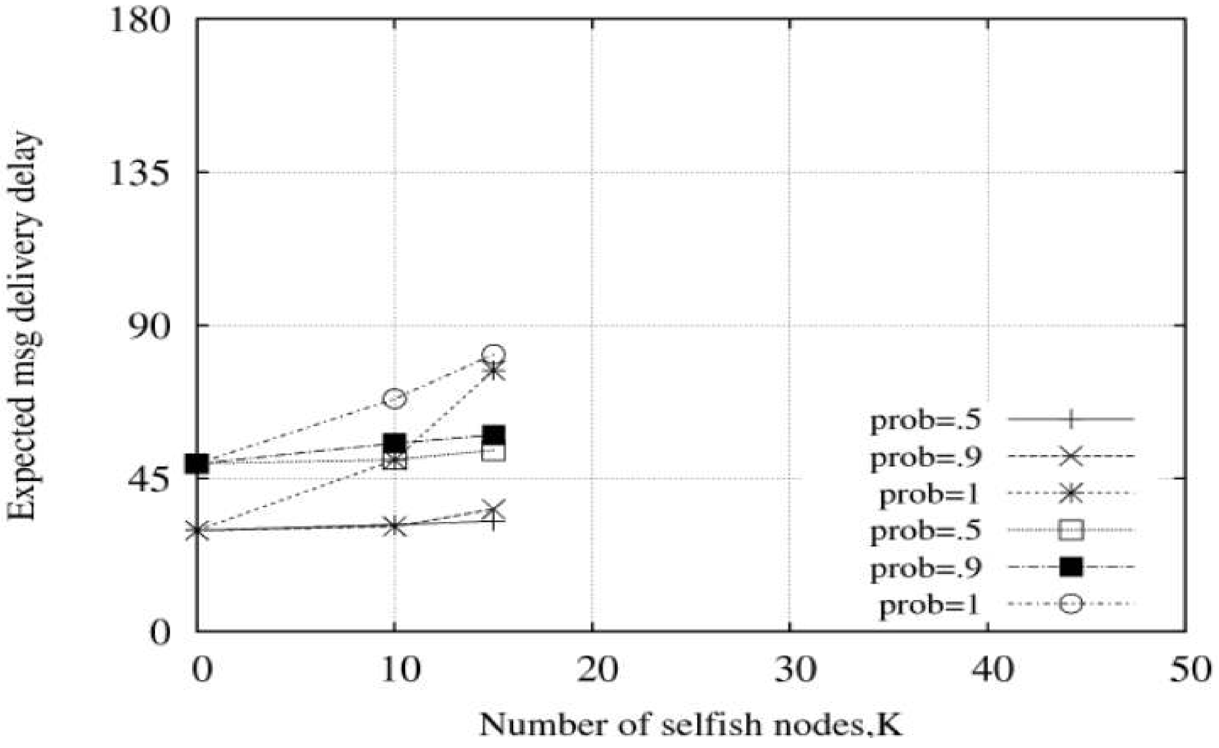

Figure 5.2 20 nodes delay graph

The delay for two hop routing protocol when there were no selfish nodes in the network was higher by few minutes. But the delay for epidemic routing when there were no selfish nodes was matching the analytical models result. On seeing the delay values for both the routing protocols we could see that the performance of epidemic routing decreases when the nodes are 100% selfish and the delay value reaches closer to two hop protocol's delay value when there are 15 selfish nodes in the network. The remaining delay values of the both the routing protocols are a bit lesser when compared to figure 2.1. The reason for it was the same as it was mentioned above for the 50 nodes scenario. Due to the random seed values and random number generator there are some variations from the result that we have achieved.

Epidemic routing:

As we could see figure 5.1 representing the graph for total nodes N= 50, the delay for both the protocols increases as the number of selfish nodes increases. The reason behind it, the normal nodes in the network does the major part for the delivery of the messages that are generated. The selfish nodes with probability equal to 1 for both no copy and no forward accounts for 0% contribution when it comes to the delivery of messages. The other specified conditions for the probability no copy and no forward account partially in the relaying process alone. Since we have assumed that the selfish nodes cannot be the source or destination of a message, they could just take part in the message relaying process in a selfish manner.

The result achieved by the analytical model in [4] was, *the performance of unrestricted relay protocol deteriorates faster than the two hop scheme for the same node selfishness intensity.* Figure 5.1 shows it practically where we could see when the selfish probability was equal to 1 for both no copy and no forward the delay for both the protocols reach almost the same peak delay. The delay values that are achieved for the other two probability set values are a bit lower from the graph of analytical model due to random number generator and seeds.

We could see the same starting values when there are no selfish nodes in figure 5.2 and figure 2.1. As the selfish nodes increases, the delay value keeps closely matching with the analytical value.

Two hop routing:

From both figure 5.1 and figure 5.2 the starting delay value for two hop routing (ie) scenarios where there are no selfish nodes when N=20 has a bit higher delay and when N=50 the delay is matching the analytical models graph. As the number of selfish nodes increases in the network we could see that the delay value is going higher.

As we could see that there is a steep increase in the delay values when both no forward and no copy probability values are set to 1. The delays are at its peak when there are more selfish nodes stating that few normal nodes are responsible for ensuring the delivery of the messages but with a higher delay.

6 CONTACT UTILIZATION

There was a particular case which was contradicting to the actual working among the nodes. The case was when few messages got delivered with lesser delays in a selfish scenario and the same messages got delivered with higher delays with no selfishness. We expect all the messages to get delivered with shorter delays when there is no selfishness in the network when compared to the presence of selfish nodes. But there were messages which had lesser delays in selfish condition. Since we got such messages with shorter delays we had to focus and analyze the reason behind it.

We considered a particular scenario in which there were 20 nodes out which 15 were selfish (K=15) in two hop routing and the intensity of selfishness was P_{nc}=P_{nf}=.5. The simulation results were obtained and we considered the messages that were consecutively delivered. The delays of those messages were stored along with the message ids. Now the simulation for 20 nodes without any selfish nodes in the network was simulated and the results were obtained. From the obtained result we just took the same messages which got delivered in the selfish scenario into account and compared their delay values.

It was contradicting to see same messages having lesser delays in the selfish scenario than the scenario without any selfish nodes. One such message was M252 and the paths of its delivery in both the normal case and the selfish case is given below.

The source and destination of M252 was P3 and P1 respectively and the message was created at the time instance 7413.

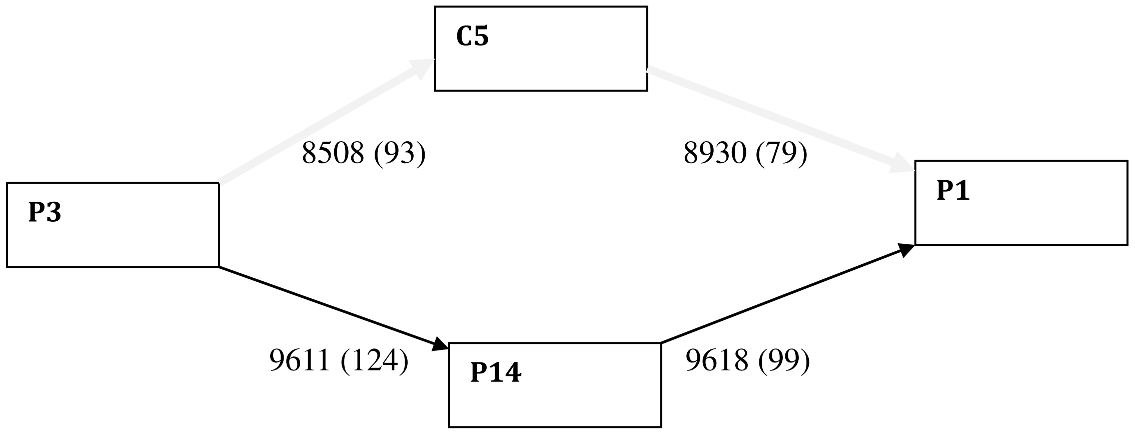

Figure 6.1 20 nodes K=15, P_{nc}=P_{nf}=.5 two hop routing

The above diagram represents the path in which M252 was relayed and delivered. In the above diagram the colored line (light grey) represents the path taken in the selfish scenario, whereas the black line represents the path taken in the case of normal scenario. The values that are given in brackets beside the time values represent the number of messages that are in the buffer of the transmitting node.

We could clearly see that the path taken with selfishness was quicker when compared to the normal path. This made us to check whether there was a connection between P3 and P5 (in the case of no selfish nodes) was established or not. After checking the connectivity log there was a connection established between those nodes but the message was not transmitted during that connection up time. The transmission speeds of all the nodes were increased to a high value so that we could transmit the messages much quickly but the increase in speed did not affect the delivery. Then we realized that the impact of the transmission speed does not influence the message in the buffer to get transmitted quickly.

You could see the number of messages that are in the buffer of the transmitting node to be specified in brackets. Since the transmission speed does not help in transmitting extremely quick transmissions all the messages gets transferred at its own pace. You could see the number of messages in buffer for the normal path is greater than the values in the selfish path.

Therefore the messages when they are transmitted in the normal case have to send all the messages in some order which is not the case that’s happening in the selfish scenario. It might send a particular message or it might not which is based on the random number generator. So in this case messages which or on the farther end of the buffer have a chance of getting delivered quickly and which is not possible in the normal case since there are several messages in the buffer to be transmitted before the last message gets transmitted.

So if it’s possible to transmit the message very quickly without having any queuing delay it would be possible to obtain the same paths even in the normal nodes scenario.

7 Probabilistic Selfishness

Graph comparing the delay variations with change in P_{nc}

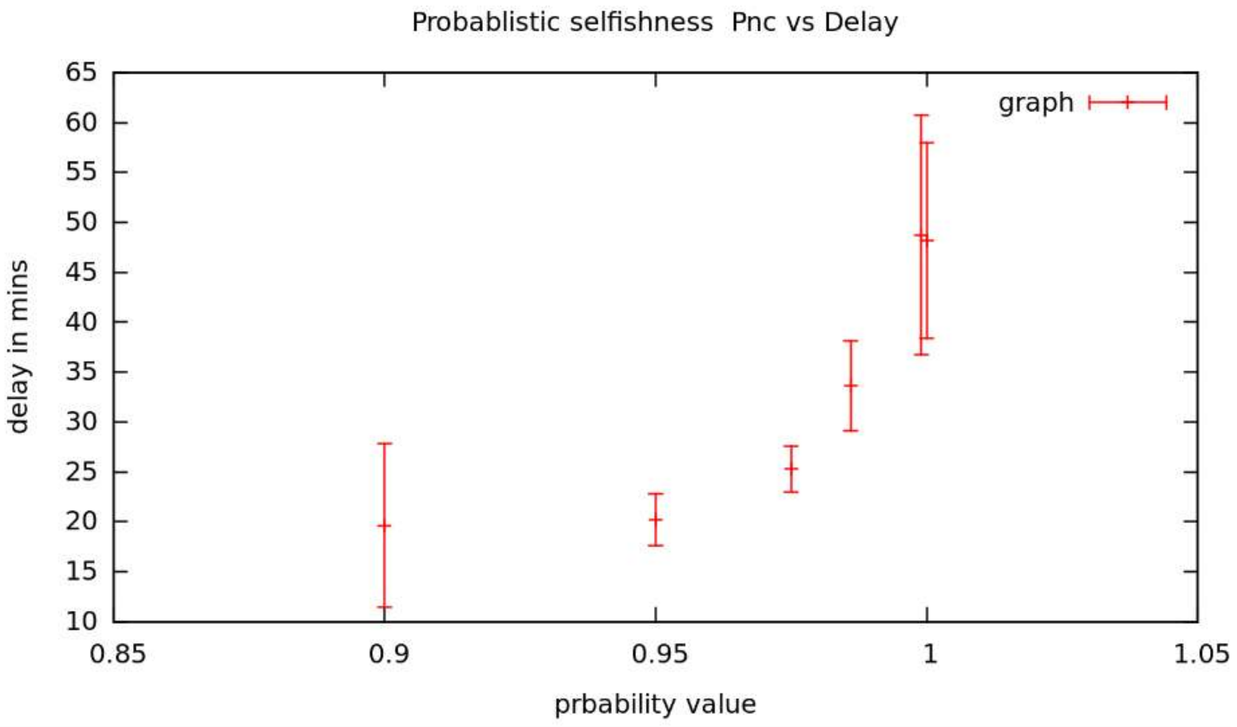

Figure 7.1 Probabilistic selfishness 50 nodes, K=45 & epidemic routing

This is the graph that is obtained when the total nodes where 50 out of which 45 are selfish (K=45) and epidemic router was used to see the delay variations when probability no copy starting from 0.9.Where the Y axis represents the delays in minutes and the X axis represents the probability values for Probability Nocopy. On seeing the result, it is evident that we get the higher delays when P_{nc} was close to 1. When start with P_{nc} =.9 we could see that delay is considerable less when compared to other values. As we increase the value from .9 we see that the delay value increases considerably. Also the delay values are expressed as error bars to indicate the errors or uncertainty in the achieved values. With the help of the error bars we could see both the upper bound and the lower bound values of the delays which are possible due to the randomness that is used. This clearly indicates that the delay results could fluctuate between the specified ranges in figure 7.1.

Since there is deviation in all the values the upper deviation from our result is closer to the delay values that are achieved in the analytical model.

8 CONCLUSION AND FUTURE WORK

8.1 CONCLUSION AND DISCUSSION

This thesis practically validates the analytical model that is proposed in [4] and check if both the results are matching. ONE simulator was the tool which was used to carry out the experiment with quite a few customizations and implementations done to it. Generated log reports were analyzed using AWK scripts and the plots were drawn using gnuplot. An understanding of all the important terminologies and concepts by reading several articles was helpful to carry out the analysis.

The experiment was carried out with the same settings as specified in [4] to have the uniformity in the process. On comparing the graphs achieved using the analytical model and the practically achieved graph we were able to see the results to be matching but with few unclear points. We were able to witness the performance of the epidemic routing was degraded and the delay values were approaching towards the values of two hop routing protocol when the selfish nodes and selfish intensity increases. We could especially point it out in the figure 5.2 when the number of selfish nodes were 45 and selfish probability was 1 the delays produced by both the protocols were almost the same value, there by matching with the results of the analytical model.

Thereby we could conclude that the performance advantage of epidemic routing decreases and approaches towards the performance of two hop routing protocol as the number of selfish nodes in the network increases along with the degree of selfishness.

8.2 FUTURE WORK

In this thesis we assume that the nodes are allowed to carry infinite buffer or buffer with a very high capacity so that there are no messages being dropped. The buffer each node carries never becomes full. Also the time to live (TTL) of each message is said to a value that is equivalent to the simulation time there by the TTL of messages never get becomes zero and get expired. But this is not at all the case in reality and often there are limitations in the buffer size as well the TTL is set to a finite value. So as a future work it would interesting to check how the delay value gets affected if we are going to have a limited buffer and finite TTL.

It would be interesting to introduce the concept of some ranks to the selfish nodes based on any metric and categorize them into some priority nodes during the routing process. Hence we could send our message or data to a selfish node which has a better priority among the selfish nodes than with nodes with a lower priority. This kind of an approach could ensure a better performance and ensure lower delays. It is possible to find the nodes which are 100% selfish in the network and avoid the message or data being forwarded to that node.

Another possible extension could be the use of two different routers in the same network one for normal nodes and another for selfish nodes. As of now we use epidemic router for routing normal nodes and epidemic selfish router for routing selfish nodes. It would be interesting to use epidemic router for normal nodes and two hop selfish router for the selfish nodes and vice versa. Repeat the same experiment and the delays could be calculated.

9 REFERENCES

[1]Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks by Luciana Pelusi, Andrea Passarella and Marco Conti,IIT-CNR.

[2] Logarithmic Store-Carry-Forward Routing in Mobile Ad Hoc Networks

Jie Wu, Senior Member, IEEE, Shuhui Yang, Student Member, IEEE, and Fei Dai

[3] http://www.netlab.tkk.fi/tutkimus/dtn/theone/

[4] Assessing the vulnerability of DTN data relaying schemes to node selfishness by Merkourious Karaliopoulos, Member, IEEE.

[5]On the Effects of Cooperation in DTNs by Antonis Panagakis, Athanasios Vaios and Ioannis Stavrakakis. Department of Informatics & Telecommunications. National & Kapodistrian University of Athens.

[6]Stochastic Properties of the Random Waypoint Mobility Model by Christian Bettstetter,Hannes Hartenstein,Xavier Perez Costa.

[7] http://mathstat.helsinki.fi/mathphys/EVERGROW/virtamo.pdf

[8]The message delay in mobile ad hoc networks by Robin Groenevelt,Philippe Nain, Ger Koole. Vrije Universiteit,Boelean 1081a,1081 HV Amsterdam, The Netherlands.

[9]Relaying in mobile ad hoc networks: The Brownian motion mobility model by R.Groenevelt,E.Altman,P.Nain.

[10]Spray and wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks by Thrasyvoulos Spyropoulos, Konstantinos Psounis, Cauligi S.Raghavendra.

[11] R. Groenevelt, Stochastic models in mobile ad hoc networks, Ph. D. thesis, University of Nice Sophia Antipolis, April 2005.

[12] M. Grossglauser and D. Tse, Mobility increases the capacity of adhoc wireless networks, IEEE/ACM Transactions on Networking, August 2002.

[13] http://en.wikipedia.org/wiki/Delay-tolerant_networking

[14] http://en.wikipedia.org/wiki/Routing_in_delay-tolerant_networking

[15] MAXProp:Routing for vehicle-based disruption-tolerant networks. In Proc. IEEE INFOCOM, April 2006. John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine

[16] Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet by Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein.

[17] Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, by Amin Vahdat and David Becker,Department of Computer Science, Duke University, April 2000.

[18] Sound Mobility Models by Jungkeun Yoon,Mingyan Liu,Brian .Noble Department of Electrical Engineering and Computer Science,University of Michigan

[19] A Survey of Mobility Models for Ad Hoc Network Research by Tracy Camp Jeff Boleng Vanessa Davies *Dept. of Math. and Computer Sciences Colorado School of Mines, Golden, CO*

[20] http://en.wikipedia.org/wiki/AWK

[21] http://www.manning.com/janert/

[22] http://www.gnuplot.info/faq/faq.html#SECTION00031000000000000000

[23] Evaluating the Impact of Social Selfishness on the Epidemic Routing in Delay Tolerant Networks by Yong Li, Student Member, IEEE, Pan Hui, Depeng Jin, Member, IEEE, Li Su, and Lieguang Zeng

[24] Routing strategies for Delay Tolerant Networks by Evan P.C.Jones and Paul A.S Ward, University of Waterloo.

[25] Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges by Z.Zhang, IEEE Communications Surveys & Tutorials, vol 8, no 1, pp.24-37, 2006

[26] Efficient routing in intermittently connected mobile networks: The multiple case copy by T.Spyropoulos, K. Psounis, and C.Raghavendra, IEEE /ACM Transactions on Networking,vol 16,no 1,pp 77-90,2008.

[27] RELICS: In-network Realization of Incentives to Combat Selfishness in DTNs by Md Yusuf Sarwar Uddin, Brighten Godfrey, Tarek Abdelzaher Department of Computer Science University of Illinois at Urbana-Champaign, IL 61801, USA

[28] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry,"Epidemic algorithms for replicated database maintenance," SIGOPS Operating Systems Review, vol. 22, pp. 8–32, January 1988.

[29] The ONE Simulator for DTN Protocol Evaluation by Ari Keränen,Jörg Ott,Teemu Kärkkäinen Helsinki University of Technology (TKK).

[30] Evaluating Mobility Pattern Space Routing for DTNs by Jeremie Leguay, Timur Friedman Vania Conan , WDTN '05 Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking