CHAPTER 1 – INTRODUCTION

An ad hoc network is a self configuring network of mobile devices which are connected through wireless links. It is a local area network that is built spontaneously as devices connect. Instead of relying on a base station to coordinate the flow of messages to each node in the network, the individual network nodes forward packets to and from each other.

A delay tolerant network is formed when the connection is intermittent and it affects the data dissemination capability. Traditional reactive and proactive routing protocols designed for the Mobile and Ad-hoc Networks (MANETs) do not perform well in the environments where the connectivity is intermittent. An ad hoc routing protocol is a convention, or standard, that controls how nodes decide which way to route packets between computing devices in a mobile ad-hoc network. Thus Epidemic, Gossip based and Probabilistic routing protocols have been proposed for this purpose.

Epidemic routing protocol supports the final delivery of messages to random destinations with minimal assumptions regarding the core topology and connectivity of the core network. Only periodic pair-wise connectivity is required to ensure the final delivery of the message. This protocol lies on the theory of epidemic algorithms. The Epidemic routing algorithm is a simple algorithm which delivers most bundles successfully but also uses a large amount of bandwidth. This takes the summary vector into use in performing its function.

Probabilistic routing protocol uses the history of encounters and transitivity. It is based on probabilistic metric called delivery predictability. This metric indicates how likely it is that each node will be able to deliver a message to a given destination. This information is used to update the internal delivery predictability vector and then the information in the summary vector is used to decide which messages to request from the other node based on the forwarding strategy used.

The routing protocols are evaluated with the aid of mobility models. In the performance evaluation of a protocol for an ad hoc network, the protocol should be tested under realistic conditions including, but not limited to, a sensible transmission range, limited buffer space

for the storage of messages, representative data traffic models, and realistic movements of the mobile users (i.e. a mobility model).

Mobility models can be divided into two categories as Entity mobility models and Group mobility models. Entity models represent mobile nodes whose movements are independent of each other. Mobility models that represent mobile nodes whose movements are dependent on each other are known as Group mobility models.

Some of the Entity mobility models are Random Walk, Random Way Point and Gauss Markov. Since many entities in nature move in extremely unpredictable ways, the Random Walk mobility model was developed to mimic this erratic movement. In this mobility model, a mobile node moves from its current location to a new location by randomly choosing a direction and speed in which to travel. This model is equivalent to the Brownian motion.

Random Way Point is a slight enhancement of the Random Walk mobility model. It includes pause times between changes in direction and/or speed. A mobile node begins by staying in one location for a certain period of time (i.e., a pause time). Once this time expires, the mobile node chooses a random destination in the simulation area and a speed that is uniformly distributed between *[min speed, max speed]*. The mobile node then travels toward the newly chosen destination at the selected speed. Upon arrival, the mobile node pauses for a specified time period before starting the process again.

The Gauss- Markov mobility model belongs to the category of entity models as well. It was designed to adapt to different levels of randomness via one tuning parameter. Initially each mobile node is assigned a current speed and direction. At fixed intervals of time, n, movement occurs by updating the speed and direction of each mobile node. Specifically, the value of speed and direction at the *nth* instance is calculated based upon the value of speed and direction at the $(n\text{-}1)^{th}$ instance.

However, all the above mentioned routing protocols have their merits and drawbacks. Through this study it is expected to evaluate the performance of all the protocols simulated with real world mobility models and to suggest the best combinations. These combinations are expected to be considered for research studies in Mobile Delay Tolerant Networks.

1.1 Objectives

Through this project, it is intended to evaluate the Gossip based, Epidemic and Probabilistic routing protocols with various mobility models which depict the real world movements and to have the best combination suggested.

The specific objectives of this project are;

- Study the behaviour of the Probabilistic and Epidemic routing protocols by associating them with mobility models. Random Way Point and Gauss Markov will be considered to as the mobility models.
- Use the JiST SWANS simulator to implement and simulate the behaviour of the routing protocols with the mobility models. Well defined performance metric will used to evaluate the data dissemination capabilities of the mentioned combinations.

CHAPTER 2 – BACKGROUND

2.1 Analysis of the problem

Epidemic routing is the protocol that has been traditionally used. This protocol is reliant on the theory of epidemic algorithm. It performs pair wise information between nodes as they receive contact with each other in order to deliver the messages to their final destination. Hosts buffer messages even if there is currently no path to the destination available. An index of these messages called a summary vector is kept by the nodes, and they are exchanged when two nodes are met. Following this exchange, it is possible for each node to determine if the other node has some message that was previously unseen to this node. If so, the node requests the messages from the other node. Thereby messages will spread like an epidemic of some disease through the network as nodes meet and "infect" each other provided, buffer space is available.

However due to interference of concurrent transmissions between nodes the ad hoc networks scale badly and also costs a lot of communication overhead. Probabilistic routing protocol has been introduced to reduce the communication overhead and to deliver more messages.

Despite the fact that the random way-point mobility model is frequently used in evaluations of mobile ad hoc protocols, it is not practical to consider real users to move around randomly, but rather move in a predictable fashion based on repeating behavioural patterns such that if a node has visited a location several times before, it is likely that it will visit that location again. This has paved the way for Probabilistic Routing as it is based on the History of Encounters and Transitivity.

The performance of the Probabilistic routing protocol is based on the probabilistic metric known as the delivery predictability. This indicates the possibility of a particular node to deliver a message to a specific destination. This protocol too, functions in a manner which is similar to the Epidemic routing protocol. Summary vectors are exchanged when two nodes meet together, but in Probabilistic routing, they also contain the delivery predictability information stored at the nodes.

Even though Probabilistic routing protocol combined with a real life mobility model, together known as Probabilistic ROuting Protocol using History of Encounters and Transitivity (PROPHET) has been introduced, this too has its drawbacks. In the evaluation First In First Out (FIFO) queue is used and whenever a new message arrives to a full queue, the message that has been in the queue for the longest time is dropped. This is not very favourable for the process and another strategy could be introduced probably to drop the message that has already been delivered to many nodes.

Another problem that prevails is the use of a considerable amount of buffer space. Therefore it might aid the reduction of buffer space and the increase of performance if an ACK is requested for the nodes, which allows messages that have already been delivered, to be eliminated from the system.

2.2 Literature Review

In delayed tolerant networks there is no guarantee of a fully connected, end to end path between source and destination nodes at any time. Traditional reactive and proactive routing protocols designed for the Mobile and Ad-hoc NETworks do not perform well in such environments where the connectivity is intermittent. Researchers around the world have proposed Epidemic, Gossip based and Probabilistic routing protocols for this purpose, and each one has its own merits.

The project intends to bring about enhancements in the Epidemic, Gossip based and Probabilistic routing protocols used in the mobile ad hoc delay tolerant networks in order to provide a more adaptive routing protocol that can adapt to the inherent properties of the network. New and better combinations of routing protocols are expected to be suggested through performance evaluation and comparison done with various combinations of the said routing protocols and real world mobility models. Thereby an enhanced probability based routing protocol will be recognized and recommended.

As discussed by (Musolesi, M. and Mascolo, C., 2006), validation of mobile ad hoc network protocols relies almost exclusively on simulation. The value of the validation is, therefore, highly dependent on how realistic the movement models used in the simulations are. Since

there are a very limited number of available real traces in the public domain, synthetic models for movement pattern generation must be used. However, most widely used models are currently very simplistic, their focus being ease of implementation rather than soundness of foundation. As a consequence, simulation results of protocols are often based on randomly generated movement patterns and, therefore, may differ considerably from those that can be obtained by deploying the system in real scenarios. Movement is strongly affected by the needs of humans to socialize or cooperate, in one form or another. Fortunately, humans are known to associate in particular ways that can be mathematically modelled and that have been studied in social sciences for years. In this paper a new mobility model founded on social network theory has been proposed. The model allows collections of hosts to be grouped together in a way that is based on social relationships among the individuals. This grouping is then mapped to a topographical space, with movements influenced by the strength of socialites that may also change in time.

Vahdat, A. and Becker, D., (2006) present a routing protocol for intermittently connected networks called Epidemic Routing. This protocol relies on the theory of epidemic algorithms by doing pair-wise information of messages between nodes as they get in contact with each other to eventually deliver messages to their destination [Demers, A. et al.(1987), Vogels, W. et al.(2002)]. Hosts buffer messages even if it there is currently no path to the destination available. An index of these messages called a summary vector is kept by the nodes, and when two nodes meet these summary vectors are exchanged. After this exchange, each node can determine if the other node has some message that was previously unseen by this node. In that case, the node requests the messages from the other node. This means that as long as buffer space is available, messages will spread like an epidemic of some disease through the network as nodes meet and infect each other. Each message must contain a globally unique message ID to determine if it has been previously seen.

Besides the obvious fields of source and destination addresses, messages also contain a hop count field. This field is similar to the TTL field in IP packets and determines the maximum number of hops a message can be sent, and can be used to limit the resource utilization of the protocol.

Messages with a hop count of one will only be delivered to their final destination. The resource usage of this scheme is regulated by the hop count set in the messages, and the

available buffer space at the nodes. If these are sufficiently large, the message will eventually propagate throughout the entire network if the possibility exists. Vahdat, A., and Becker, D., (2006) do however show that by choosing an appropriate maximum hop count, delivery rates can still be kept high while the resource utilization is lower in the scenarios used in their evaluation.

Many mobility models have been presented for the testing of protocols and algorithms of mobile ad hoc networks. A comprehensive review of the most popular mobility models used by the mobile ad hoc research community can be found in Camp, T. et al.(2006). However, it is interesting and, at the same time, surprising to note that even the best solutions and approaches have only been tested using completely random models such as the Random Way-Point model, without grouping mechanisms, or using other simple groups mobility models, like Hong, X. et al.(1999). The almost pervasive adoption of such models has generated a considerable amount of work that is predicated on the reasonableness of random mobility models.

Lindgren, A. et al.(2003) present a probabilistic routing protocol for intermittently connected networks. In such networks there is no guarantee that a fully connected path between source and destination exists at any time, rendering traditional routing protocols unable to deliver messages between hosts. However there exist a number of scenarios where connectivity is intermittent, but where the possibility of communication still is desirable. Thus, there is a need for a way to route through such networks. They show that Probabilistic ROuting Protocol using History of Encounters and Transitivity (PROPHET) is able to deliver more messages than Epidemic Routing with a lower communication overhead.

In the evaluation Lindgren, A. et al.(2003) they have used a FIFO queue at the nodes, so whenever a new message arrives to a full queue, the message that has been in the queue for the longest time is dropped. It might be better to use some other strategy here, and for example drop the message that has already been forwarded to the largest number of other nodes. To reduce the required buffer space, and to further improve performance, it would be interesting to evaluate the impact of allowing nodes to request an ACK to their message. This would allow messages that already have been delivered to be purged from the network, leaving more resources for the other messages, most likely increasing the probability of those messages being delivered. They claim that the simple forwarding strategy used by

PROPHET in their evaluation worked fairly well and outperformed Epidemic Routing. However, it is still interesting to investigate other forwarding strategies to see if performance can be enhanced further.

2.3 Scope

This study aimed to evaluate the functions of the Epidemic and the Probabilistic routing protocols combined with real world mobility models associated with the delay tolerant networks. Random Way Point and Gauss Markov were the mobility models that were used to simulate the movement patterns of the communicating nodes.

The JiST SWANS platform was used to develop the two routing protocols and the Gauss Markov mobility model. The simulation with the mobility models was also performed on this platform. JiST is a discrete event simulator which is based on Java and SWANS is a general purpose discrete event simulation engine built on top of JiST platform. The features of both JiST and SWANS are used in the simulation process and some components of SWANS are specifically used to configure the mobility models. The routing protocols were evaluated individually combined with the mobility models

Probabilistic routing protocol was assessed as a collection of nodes among which the messages are routed; known as the Probabilistic routing zone. The metric used for Probabilistic routing protocol is the delivery predictability. Similarly the Epidemic routing protocol was also evaluated and the two protocols were compared based on their performance in terms of the delivery ratio and the latency.

Different problems were faced during the different phases of implementation and they were addressed and solved using various methods. For the Probabilistic routing protocol implementation demanded a longer time than the Epidemic routing protocol as the receipt of messages was not functioning as expected. The issue was resolved after many attempts of troubleshooting and debugging. The implementation of the Epidemic protocol too faced several problems, one being a simulation exception. The root cause behind this problem was

found out to be exceeding of the buffer. Thus the Epidemic protocol was redesigned to address the issue.

However, despite the two routing protocols being successfully implemented, it was also observed and identified that further improvements could be made pertaining to different areas of implementation and execution.

CHAPTER 3 – SOLUTION DESCRIPTION

3.1 Methodology

The solution which was decided was to study the behaviour of the Probabilistic and Epidemic routing protocols by associating them with mobility models by using the JiST SWANS simulator.

3.1.1 JiST SWANS Simulator

3.1.1.1 JiST

JiST, which stands for Java In Simulation Time is a discrete-event simulator which is based on Java. In the context of this project, it has been used to simulate the motions of the mobility models.

JiST is built with the integration of systems and languages used previously with the intention of executing discrete events efficiently and transparently with the following three characteristics.

- Is **efficient** since it has a better runtime compared to the existing simulating systems.
- Is **transparent** since the simulations are automatically transformed to run with *simulation time* semantics;
- Uses a **standard** systems language and runtime.

3.1.1.2 Architecture of JiST

JiST consists of four components which are as follows:

- Compiler
- Byte code rewriter
- Simulation Kernel
- Virtual Machine

The sequence of how these four components operate to simulate a motion is illustrated in the following block diagram.

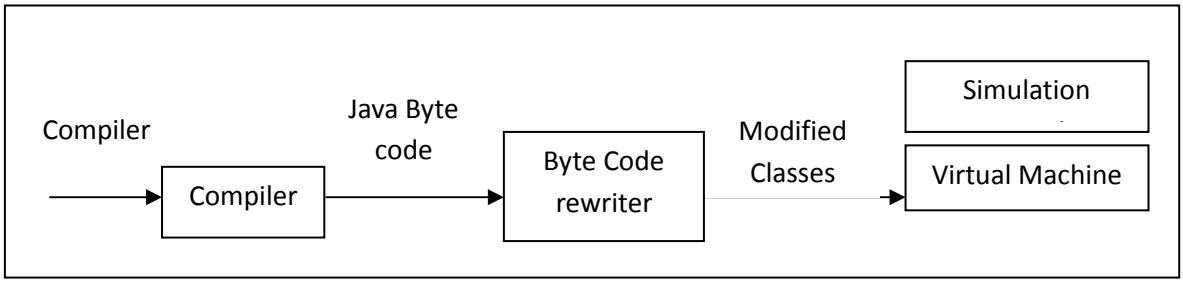

Figure 3.1 – Component diagram of JiST

JiST simulation programs are written in plain Java. They are then compiled into byte code with the usage of a compiler. The byte code rewriter then performs its function by modifying those classes to run over a simulation kernel and to support the simulation time semantics. Since the simulation program, the rewriter and the JiST kernel are all written in pure Java, it is considered that the entire process occurs within a standard, unmodified Java Virtual Machine (JVM).

3.1.1.3 SWANS

SWANS, which stands for Scalable Wireless Ad hoc Network Simulator is a general purpose discrete event simulation engine built atop a JiST platform. SWANS was created primarily because existing wireless network simulation tools are not sufficient for current research needs. SWANS also serves as a validation of the virtual machine-based approach to simulator construction.

3.1.1.4 SWANS Design

The SWANS software comprises of independent software components which can be composed and collaborated to form a complete wireless network or sensor network simulations. There are components that implement different types of applications; networking, routing and media access protocols; radio transmission, reception and noise models; signal propagation and fading models; and node mobility models.

Every SWANS component is encapsulated as a JiST entity. It has the capability to store its own local state and to interact with other components via exposed event-based interfaces. SWANS contains components for constructing a node stack, as well as components for a variety of mobility models and field configurations. This simplifies the simulation development by reducing the problem to creating relatively small, event-driven components. It also explicitly partitions the simulation state and the degree of inter-dependence between components. It also allows components to be readily interchanged with suitable alternate implementations of the common interfaces and for each simulated node to be independently configured. Finally, it also confines the simulation communication pattern.

In this project, the components of SWANS that have the capability to configure the mobility models have been specifically used.

An important factor to be noted is that in JiST, communication among entities is very efficient. The design incurs no serialization, copy, or context-switching cost among co-located entities, since the Java objects contained within events are passed along by reference via the simulation time kernel.

3.1.1.5 Embedding Java-based network applications

SWANS runs regular, unmodified Java network applications over the simulated network, thus allowing for the inclusion of existing Java-based software, such as web servers, peer-to-peer applications and application-level multicast protocols. These applications operate in simulation time within the same JiST process space, allowing far greater scalability.

This integration is achieved via a special AppJava application entity designed to be a harness for Java applications. This harness inserts an additional rewriting phase into the JiST kernel, which substitutes SWANS socket implementations for any Java counterparts that occur within the application. These SWANS sockets have identical semantics, but send packets through the simulated network. Specifically, the input and output methods are still blocking events (built using JiST continuations). To support these blocking semantics, JiST automatically modifies the necessary application code into continuation-passing style,

which allows the application to operate within the event-oriented simulation time environment.

3.1.1.6 JiST SWANS for mobility models

The SWANS physical layer components are responsible for modelling signal propagation among radios as well as the mobility of nodes. Node mobility is implemented as an interface-based discretized model. Upon each node movement, the model is queried to schedule the next movement.

The mobility models that are implemented include static and random-waypoint.

3.1.1.7 jist.swans.field

Interface	Class	Description
Placement Mobility	Random Way Point	Pick a random "waypoint" and walk towards it with some random velocity, then pause and repeat.
	Gauss Markov	Initially each mobile node is assigned a current speed and direction. At fixed intervals of time, *n*, movement occurs by updating the speed and direction of each mobile node.

Table 3.1 – Mobility models under jist.swans.field

3.2 Probabilistic Routing Protocol

The probabilistic routing protocol is ideally used in networks where the nodes comprise mobility patterns that have a certain amount of predictability. For the accuracy of the predictability, the network is expected to exist under similar mobile connectivity patterns of the nodes for a long time.

The Probability Routing Protocol is able to exchange the necessary information link as a local TCP link is established between the nodes. This TCP link is used for the protocol signalling. However when the large Delay Tolerant Networks are taken into consideration, the routing conditions might be different in each segment.

Hence Probabilistic Routing zone was considered when assessing the routing protocol in this project. It is a collection of nodes among which the messages are routed using the Probabilistic Routing Protocol.

Metric called delivery predictability is used for probabilistic routing. 0<= P_(A,B)<=1, is established at every node A for each known destination B. With the calculation of this metric, it is determined who would be a better candidate to deliver a bundle to a certain destination. The metric with the highest value would be considered as the best. If P_(A,B)>P_(C,B), bundles for destination B are preferable to forward to A rather than to C.
When two probabilistic routing nodes are in communication with each other the first information that would be exchanged would be the delivery predictability for all destinations. This information then would be used to update the internal delivery predictability vector. Subsequently, the nodes exchange information about the bundles being carried by each node. The decision about which bundles should be requested to be forwarded will be made based on this information.

3.2.1 Delivery Predictability Calculation

In the following equations, the updates are performed by node A and P_(A,B) is the delivery predictability value that node A has stored for the destination node B. If there is no predictability value is stored, P_(A,B) is considered to be zero.

Delivery Predictability calculation is inclusive of three parts. Following paragraphs will explain about the three scenarios.

3.2.1.1 When Node A meets Node B

Firstly the delivery predictability will be updated for each other, such that nodes which encounter frequently have very high delivery predictability.

If node A has not met node B for a long time;

P_(A,B) < P_first_threshold;

then P_(A,B) = P_encounter_first

else;

P_(A,B) = P_(A,B)_old + (1-delta-P_(A,B)_old)*P_encounter

where, 0<=P_encounter<=1 is a scaling constant which sets the rate at which the predictability increases following the first encounter. Delta is a number which is greater than zero , however is small and sets an upper bound for P_(A,B).

3.2.1.2 When Node A not encountering Node B

When node A does not meet node B during a certain time interval, it conveys the fact that they are not good forwarders of bundles to each other and thus the delivery predictability decreases. This process is known as aging and the equation is as follows;

$P_{(A,B)} = P_{(A,B)\_old} * gamma^{K}$

In the above equation, 0<=gamma<=1 is the aging constant. K is the number of time units that have been passed since the last time the metric aged.

3.2.1.2 When Node A meeting with Node B and Node B meeting with Node C frequently

The frequent meeting of node A with node B and node B with node C, helps to identify the transitive property of the delivery predictability. This observation indicates that node C might be ideal for node B to forward bundles to.

The following equation exhibits how delivery predictability is affected by the transitive property.

P_(A,C) = MAX (P_(A,C)_old,P_(A,B)*P_(A,C)*beta)
0<=beta <=1 is a scaling constant that controls the magnitude of the impact that transitivity should have on the predictability.

3.3 Epidemic Routing Protocol

Epidemic Routing Protocol is one of the protocols used to deliver messages when there is no connected path from source to destination or when a network is partitioned when a message is originated. Epidemic Routing has 3 main goals. They are namely;

1) maximize message delivery rate
2) minimize message latency
3) minimize the total resources consumed in message delivery

In Epidemic Routing application messages are distributed to hosts, called carriers, within connected portions of ad hoc networks. This ensures that messages are quickly distributed through connected portions of the network. The carriers then come into contact with another connected portion of the network through node mobility. At this point, the message spreads to an additional island of nodes. Because of this transitive transmission of data in Epidemic Routing, messages have a high probability of eventually reaching their destination.

The main goal of epidemic routing is to maximise the message delivery rate and minimise the latency rate. This goal is achieved by placing an upper bound on the message hop count and the per-node buffer space.

Each host maintains a buffer consisting of messages that it has originated as well as messages that it is buffering on behalf of other hosts. All these messages are listed in a hash table with a unique identifier for each message. This enhances the efficiency of the system. Each host stores a bit vector called the summary vector that indicates which entries in their local hash tables are set. When two hosts come into communication range of one another, the host with the smaller identifier initiates an anti-entropy session with the host with the larger identifier. Each host maintains a collection of hosts with which it has conversed recently to avoid redundant connections. Anti-entropy is not re-initiated with remote hosts that have been contacted within a configurable time period.

The two hosts exchange their summary vectors during anti-entropy, to determine which messages stored remotely have not been seen by the local host. Each host then requests copies of messages that it has not yet seen. The receiving host maintains total independence in deciding whether it will accept a message. For example, it may determine that it is unwilling to carry messages larger than a given size or destined for certain hosts.

The hop count field used in the Epidemic Routing Protocol, indicates the maximum number of epidemic exchanges that a certain message has to experience. The hop count is similar to the TTL field in IP packets. Messages with a hop count of one will only be delivered to their end destination. Packets which have a hop count of one are dropped due to the requirements of buffer space which is available locally. In contrary messages with larger hop counts will be distributed faster than those with lower hop counts. This reduces average delivery time, but also increases total resource consumption in message delivery. High priority messages therefore can be marked with a high hop count, while most messages can be marked with a value close to the expected number of hops for a given network configuration to minimize resource consumption.

As messages are delivered probabilistically in epidemic routing, some applications require acknowledgments of message delivery. The ack request field signals the destination of a message to provide an acknowledgment of message delivery. These acknowledgments are

simple return messages from receiver back to the sender. The acknowledgment upon successful delivery can also be attached to any other message which is being sent back to the sender..

Each host sets a maximum buffer size for epidemic message distribution. The buffer size limits the amount of memory and network resources consumed through Epidemic Routing. Usually hosts drop older messages in favour of newer ones upon reaching their buffer's capacity. To ensure the successful delivery of all messages, the buffer size on at least a subset of nodes must be roughly equal to the expected number of messages in transit at any given time. It could otherwise result in flushing off of older messages before they are delivered.

Different management strategies could be incorporated for the per-host message buffer, the simplest of them being first -in-first-out (FIFO). This policy is simple to implement and bounds the amount of time that a particular message is likely to remain resident in at least one buffer. Once enough new messages have been introduced into the system, older messages are likely to be flushed from most buffers. FIFO is considered a reasonable policy as long as the buffer size on all hosts is larger than the expected number of messages in transit at any given time. However, if available buffer size is limited compared to the number of messages, FIFO is not very suitable when the Quality of Service (QoS) is taken into consideration. For example, a host's aggregate buffer utilization is directly proportional to the number of messages it sends, which may not be fair to other hosts. FIFO does not provide any mechanisms for preferentially delivering or storing high priority messages.

CHAPTER 4 - IMPLEMENTATION

4.1 Protocol Implementation

Two routing protocols were implemented inside the jist.swans.route package. Following functionalities were implemented for both Probabilistic and Epidemic protocols:

- RREQ and RREP messages (for route discovery)
- RERR messages, HELLO messages, and precursor lists (for route maintenance)
- Sequence numbers
- Hop counts

Each router (Epidemic or Probabilistic) is essentially a state machine that processes incoming requests from the SWANS network entity. When the network entity needs to send a message to another node, it calls upon the corresponding router to determine the next-hop.

4.1.1 Route Discovery

When a node needs to determine a route to a destination node, it floods the network with a Route Request (RREQ) message. The originating node broadcasts a RREQ message to its neighbouring nodes, which broadcast the message to their neighbours, and so on. To prevent cycles, each node remembers recently forwarded route requests in a route request buffer. As these requests spread through the network, intermediate nodes store reverse routes back to the originating node. Since an intermediate node could have many reverse routes, it always picks the route with the smallest hop count.

When a node receiving the request either knows of a “fresh enough” route to the destination (see section on sequence numbers), or is itself the destination, the node generates a Route Reply (RREP) message, and sends this message along the reverse path back towards the originating node. As the RREP message passes through intermediate nodes, these nodes update their routing tables, so that in the future, messages can be routed though these nodes to the destination.

In the flooding protocol described above, when a node originates or forwards a route request message to its neighbours, the node will likely receive the same route request message back from its neighbours. To prevent nodes from resending the same RREQs (causing infinite cycles), each node maintains a *route request buffer*, which contains a list of recently broadcasted route requests. Before forwarding a RREQ message, a node always checks the buffer to make sure it has not already forwarded the request. RREQ messages are also stored in the buffer by a node that originates a RREP message. The purpose for this is so a node does not send multiple RREPs for duplicate RREQs that may have arrived from different paths. The exception is if the node receives a RREQ with a better route (i.e. smaller hop count), in which case a new RREP will be sent.

Each entry in the route request buffer consists of a pair of values: the address of the node that originated the request, and a route request identification number (RREQ id). The RREQ id uniquely identifies a request originated by a given node. Therefore, the pair uniquely identifies a request across all nodes in the network.

To prevent the route request buffers from growing indefinitely, each entry expires after a certain period of time, and then is removed. Furthermore, each node's buffer has a maximum size. If nodes are to be added beyond this maximum, then the oldest entries will be removed to make room.

4.1.2 Scalability of searching nodes

Whenever a node requests a route, it sends a message that passes through potentially every node in the network. When the network is small, this is not a major concern. However, when the network is large, this can be extremely wasteful, especially if the destination node is relatively close to the RREQ originator. Preferably, we would like to set the TTL value on the RREQ message to be just large enough so that the message reaches the destination, but no larger. However, it is difficult for a node to determine this optimal TTL without prior global knowledge of the network.

As a solution to this problem this problem, below described methodology is used. When a node initiates a route request, it first broadcasts the RREQ message with a small TTL value; i.e. 1. If the originating node does not receive a RREP message within a certain period of

time, it rebroadcasts the RREQ message with a larger TTL value (and also a new RREQ identifier to distinguish the new request from the old ones). The node continues to broadcast messages with increasing TTL and RREQID values until it receives a route reply. If the TTL values in the route request have reached a certain threshold, and still no RREP messages have been received, then the destination is assumed to be unreachable, and the messages queued for this destination are thrown out.

4.1.3 Sequence Numbers

Each destination (node) maintains a monotonically increasing sequence number, which serves as a logical time at that node. Also, every route entry includes a destination sequence number, which indicates the "time" at the destination node when the route was created. The protocol uses sequence numbers to ensure that nodes only update routes with "newer" ones. Doing so, we also ensure loop- freedom for all routes to a destination.

All RREQ messages include the originator's sequence number, and its (latest known) destination sequence number. Nodes receiving the RREQ add/update routes to the originator with the originator sequence number, assuming this new number is greater than that of any existing entry. If the node receives an identical RREQ message via another path, the originator sequence numbers would be the same, so in this case, the node would pick the route with the smaller hop count.

If a node receiving the RREQ message has a route to the desired destination, then we use sequence numbers to determine whether this route is "fresh enough" to use as a reply to the route request. To do this, we check if this node's destination sequence number is at least as great as the maximum destination sequence number of all nodes through which the RREQ message has passed. If this is the case, then we can roughly guess that this route is not terribly out-of-date, and we send a RREP back to the originator.

As with RREQ messages, RREP messages also include destination sequence numbers. This is so nodes along the route path can update their routing table entries with the latest destination sequence number.

4.1.4 Route Maintenance

Each node keeps track of a *precursor list*, and an *outgoing list*. A precursor list is a set of nodes that route through the given node. The outgoing list is the set of next-hops that this node routes through. In networks where all routes are bi-directional, these lists are essentially the same.

Each node periodically sends HELLO messages to its precursors. A node decides to send a HELLO message to a given precursor only if no message has been sent to that precursor recently. Correspondingly, each node expects to periodically receive messages (not limited to HELLO messages) from each of its outgoing nodes. If a node has received no messages from some outgoing node for an extended period of time, then that node is presumed to be no longer reachable.

Whenever a node determines one of its next- hops to be unreachable, it removes all affected route entries, and generates a Route Error (RERR) message. This RERR message contains a list of all destinations that have become unreachable as a result of the broken link. The node sends the RERR to each of its precursors. These precursors update their routing tables, and in turn forward the RERR to their precursors, and so on.

To prevent RERR message loops, a node only forwards a RERR message if at least one route has been removed.

4.2 Mobility Model implementation

Random Way Point and Gauss Markov were the two mobility models that were focused on, in this study. Random Way point is already implemented within the Mobility.java file under jist.swans.filed package.

Gauss Markov was implemented within the mobility.java file to enhance the evaluation procedure. It was designed to adapt to different levels of randomness via one tuning parameter. Initially each mobile node is assigned a current speed and direction. At fixed intervals of time, *n*, movement occurs by updating the speed and direction of each mobile

node. Specifically, the value of speed and direction at the *nth* instance is calculated based upon the value of speed and direction at the $(n\text{-}1)^{th}$ instance.

CHAPTER 5 – RESULTS AND EVALUATION

5.1 Results

Initially results were generated for 12 nodes within 500m*500m, 1000m*1000m and 2000m*2000m areas for the Probabilistic and Epidemic routing protocols. The performance of the two routing protocols in combination with the different mobility models was then evaluated. The Following figures depict the delivery ratio and latency of the messages. Simulations were run for 1 hour, 2 hours, 4 hours, 6 hours, 8 hours and 10 hours respectively.

The following figure elaborates the delivery ratio for each mobility model association within an area of 500m*500m, when the simulation was conducted for the mentioned time periods. Refer Appendix A for the dataset.

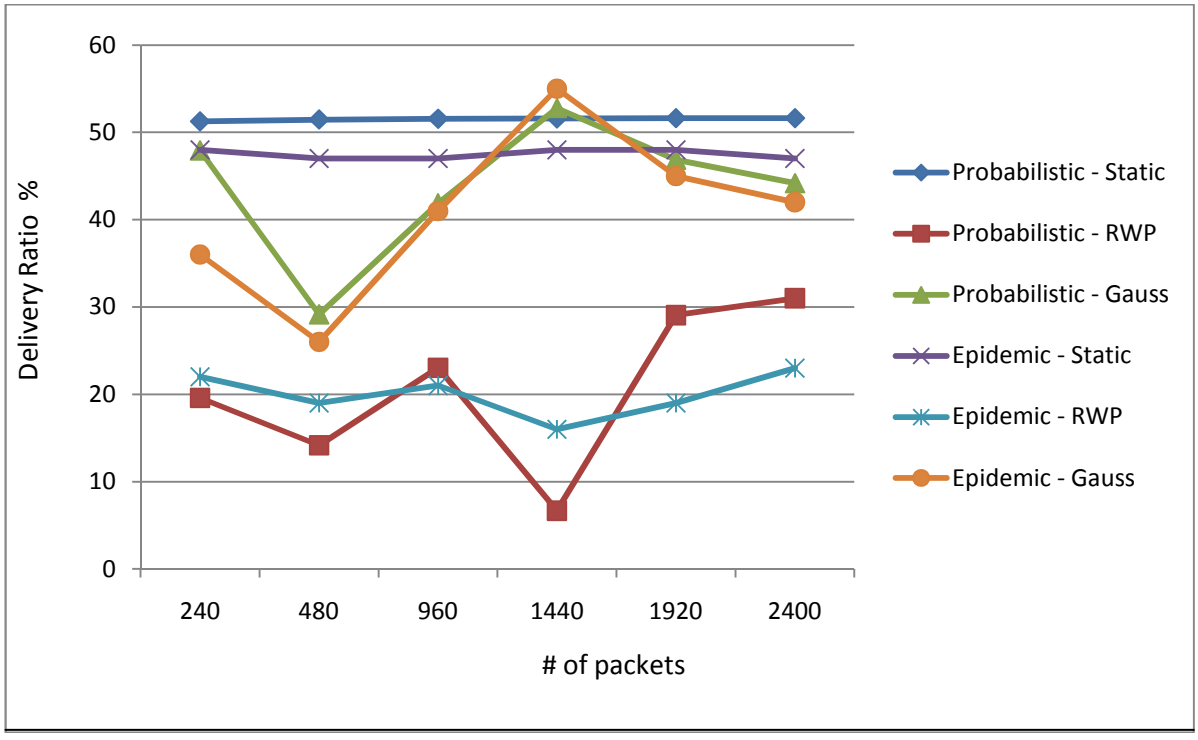

Figure 5.1 – Delivery ratio Vs # of packets for 500m*500m area

This figure is a depiction of the latency in an area of 500m*500m.

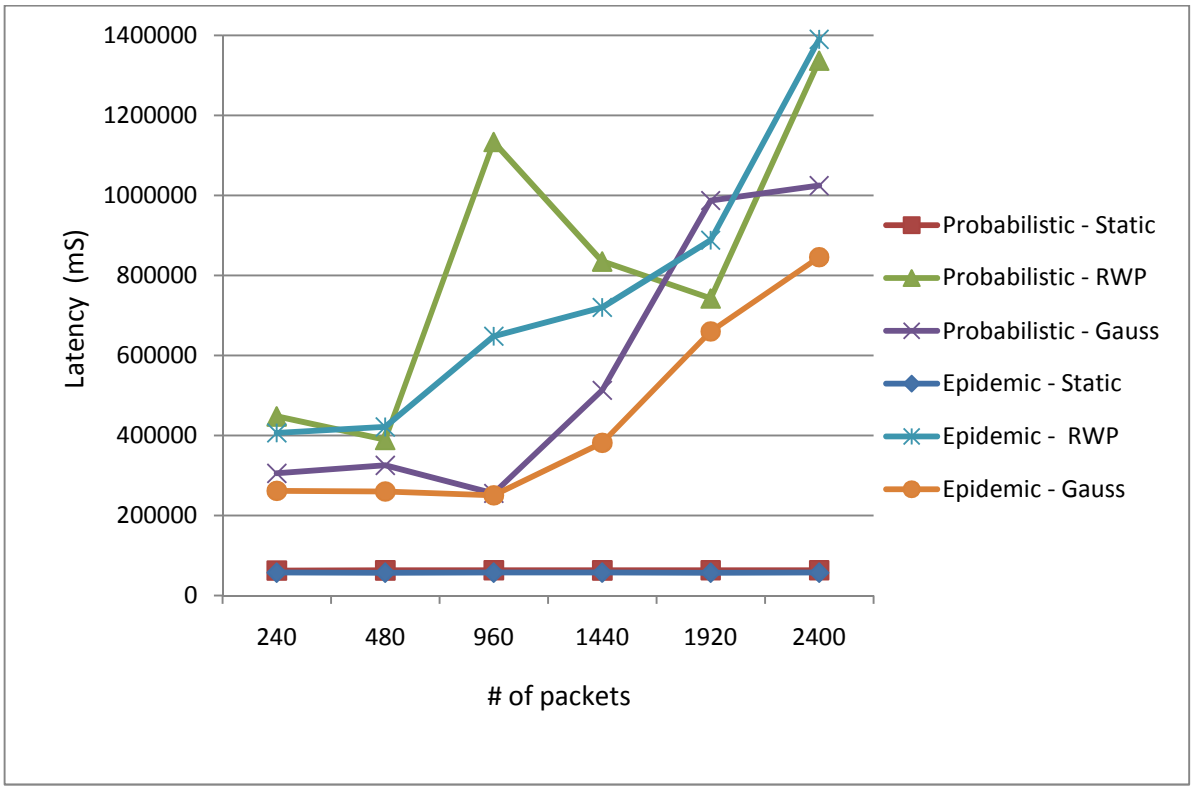

Figure 5.2 – Latency (mS) Vs number of packets for 500m*500m area

The results obtained for the delivery ratio for an area of 1000*1000m are as follows:

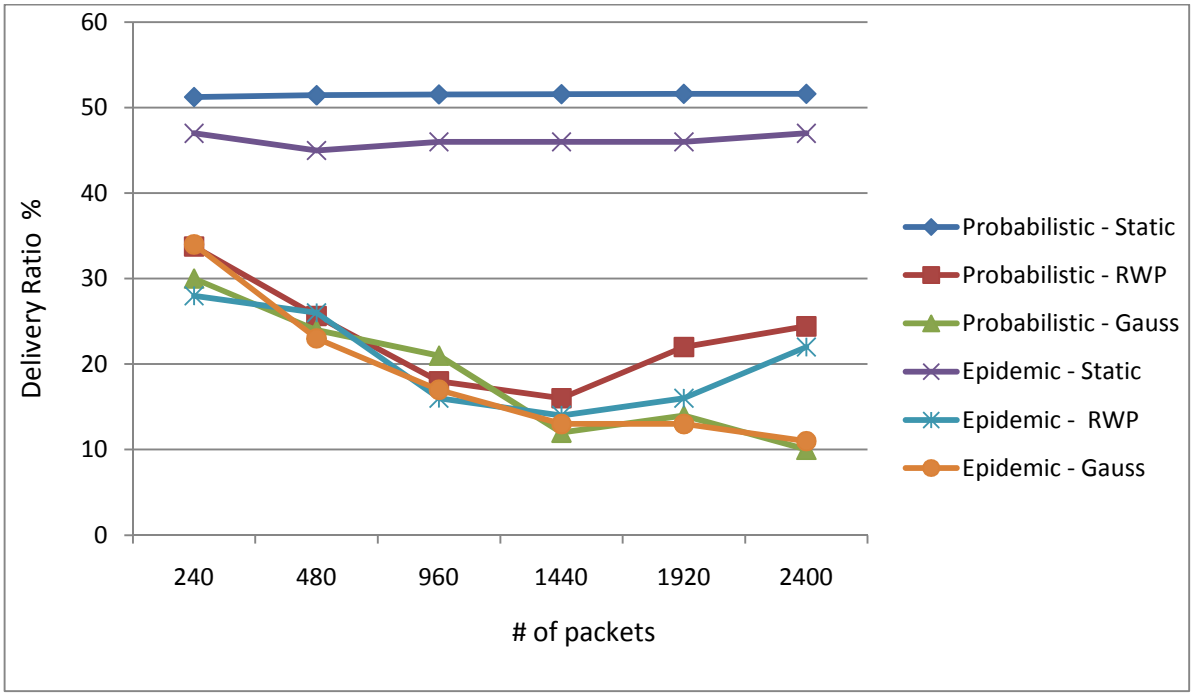

Figure 5.3 – Delivery ratio Vs # of packets for 1000m*1000m area

The latency for an area of 1000m*1000m:

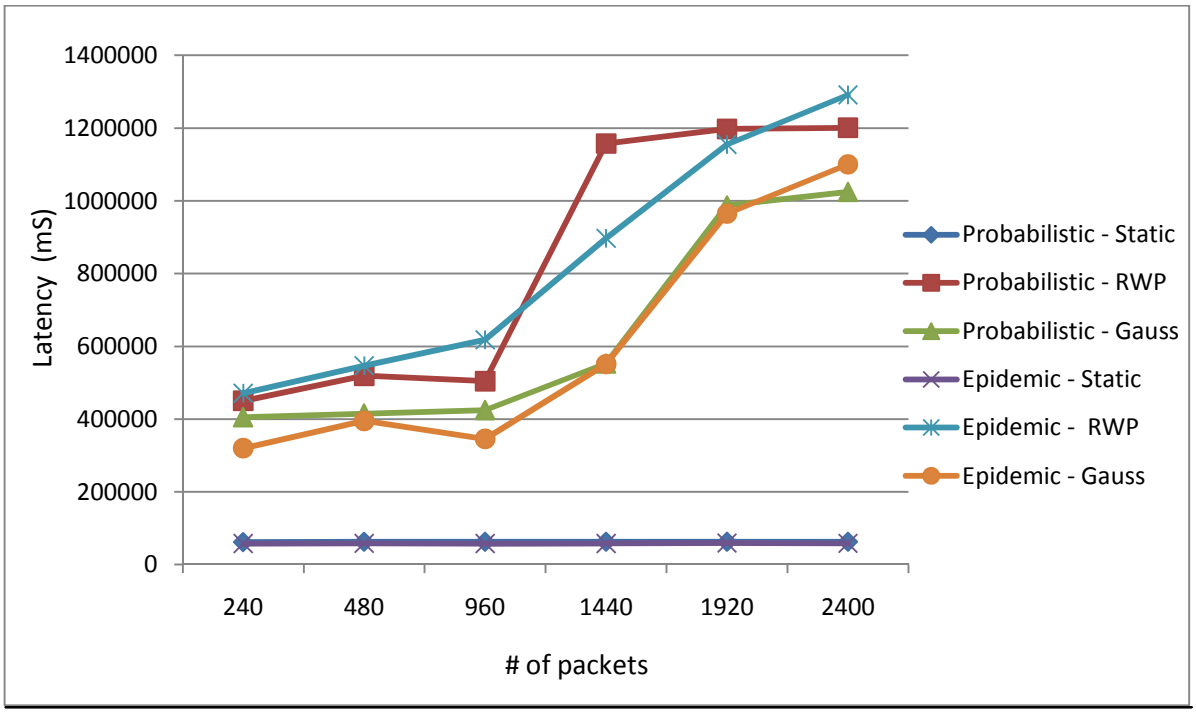

Figure 5.4 – Latency (mS) Vs number of packets for 1000m*1000m area

The following illustrates the delivery ratio for 2000*2000 m.

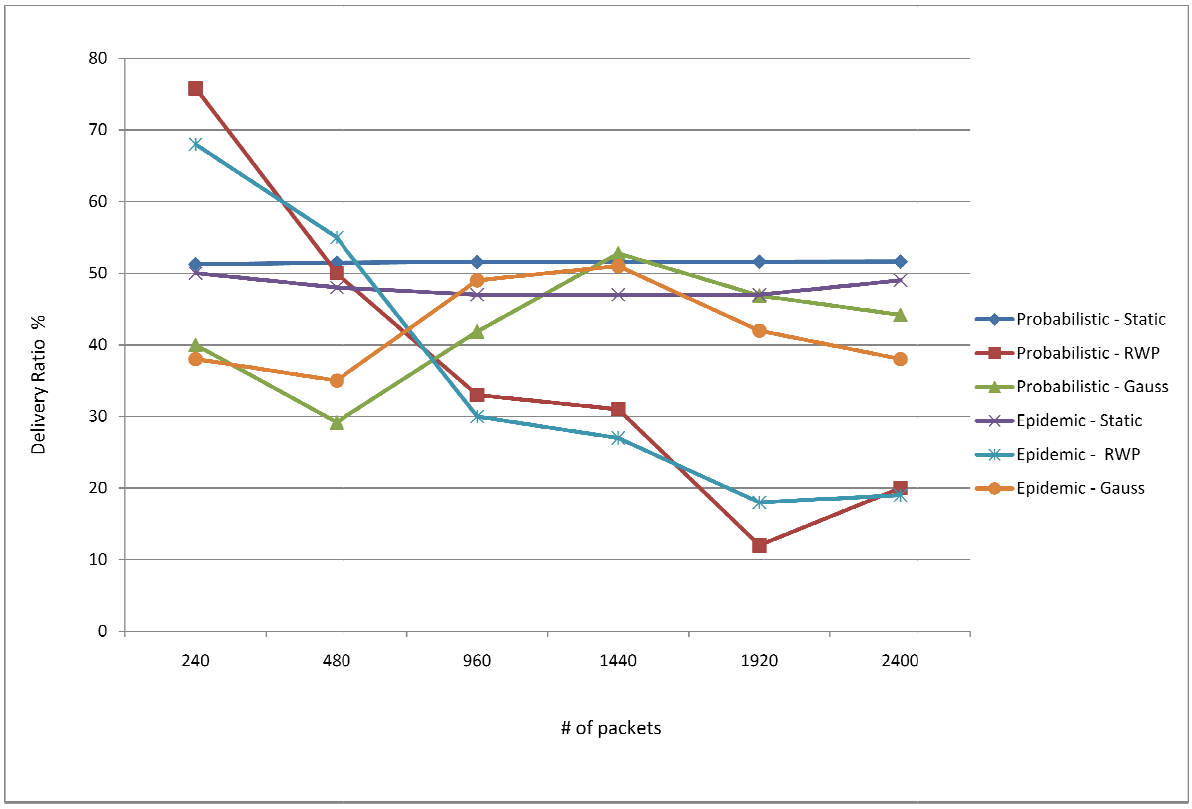

Figure 5.5 – Delivery ratio Vs # of packets for 2000m*2000m area

The latency when tested in an area of 2000m*2000m is shown in the graph below:

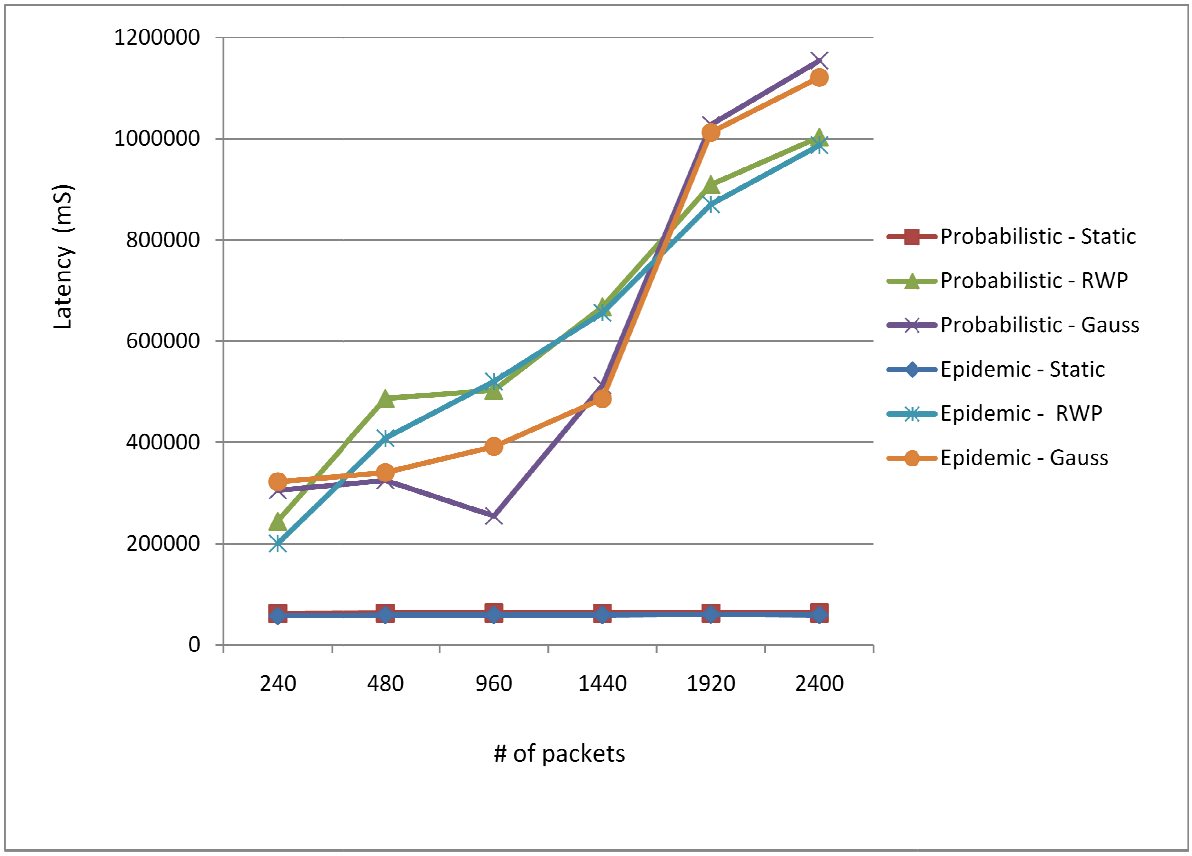

Figure 5.6 - Latency (mS) Vs number of packets for 2000m*2000m area

5.2 Evaluation

The figures above illustrate the different combination patterns received for the routing protocols with the mobility models. With the results having been carried out within areas of different dimensions, it was interesting to note that similar patterns could be identified.

With reference to figures 5.1, 5.3 and 5.5, the Random Way Point mobility model has performed better with the Probabilistic routing protocol compared to the combination with the Epidemic routing protocol. It could also be observed that the difference between the two graphs, respectively Probabilistic and Epidemic combined with RWP, has decreased when the area has increased. Thus the delivery ratio of Random Way Point model with the Probabilistic routing protocol is more than it is with the Epidemic routing protocol. It could

further be observed that the other mobility models namely, Static and Gauss Markov too performed better in terms of delivery ratio when they were combined with Probabilistic routing protocol rather than the Epidemic routing protocol even though they were not consistent at the first few hours.

The latency of the two routing protocols when combined with the mobility models is depicted in the figures 5.2, 5.4 and 5.6. It was observed that the combination of the Random Way Point mobility model with the Probabilistic routing protocol fared better than it did with the Epidemic routing protocol. In all three areas, the latency of the Probabilistic routing protocol combined mobility models was higher than the Epidemic routing protocol. However, the trend changed during the later hours. The change in the graphs where the latency of the Epidemic routing protocol being higher than that of the Probabilistic routing protocol was mainly evident during the last two to three hours. The same trend could also be observed for the Gauss Markov and Static mobility model combinations with the routing protocols. Thus it could be identified that the Probabilistic routing protocol fared better than the Epidemic routing protocol in terms of latency as well.

The performance of a routing protocol is deemed acceptable and useful when the delivery ratio is higher and the latency is lower. In the study that was conducted the Probabilistic routing protocol in comparison with the Epidemic routing protocol was observed to have performed better.

CHAPTER 6 – CONCLUSION AND FUTURE WORKS

6.1 Discussion

The concepts behind Epidemic and Probabilistic routing protocols were carefully and successfully studied before the implementation of the project. The calculations of metrics pertaining to each protocol were given more attention as a thorough understanding of the calculations was required for the implementation.

After gaining the required knowledge for the implementation, the routing protocols were implemented on JiST SWANS platform by using Java language. Apart from the two routing protocols, the Gauss Markov mobility model too was implemented on the same platform as it was not an embedded model of JiST SWANS, unlike Static and Random Way Point mobility models.

Various issues and problems that arose during the process of implementation were addressed successfully. The Probabilistic routing protocol implementation demanded a longer time than the Epidemic Routing protocol as the receipt of messages was not functioning as expected. The issue was resolved after many attempts of troubleshooting and debugging.

After the successful implementation of the Probabilistic routing protocol, it was successfully combined with the mobility models and the expected outputs could be gained.

However during the implementation of the Epidemic protocol another problem occurred which was a simulation exception. The root cause behind this problem was found out to be exceeding of the buffer. Thus the Epidemic protocol was redesigned to address the issue. When the buffer was increased it led to various other bugs which consumed a lot of time to be resolved. The Epidemic routing protocol was thereby successfully implemented. However the time consumed for the multiple attempts of troubleshooting and debugging caused constraints in evaluating the mobility models combined with the Epidemic protocol.

Despite the various problems faced in the process of implementation the objectives of the project were successfully achieved.

6.2 Future Works

Even though the two routing protocols were successfully implemented, it was observed and identified that further improvements could be made pertaining to different areas of implementation and execution.

Despite the rise of new network possibilities due to the lightweight wireless devices, applicable infrastructure may not be duly available at all times. Thus more attention could be paid to solutions that could handle periods of intermittent disconnectivity.

In this project FIFO queue at the nodes have been used and therefore the message that has been in the queue for the longest time is dropped whenever a new message arrives at the node. It might be beneficial to use some other strategy here, and for example drop the message that has already been forwarded to the largest number of other nodes.

Also, the nodes could be made to request for an acknowledgment to their message in order to reduce buffer space and further improve performance.

With respect to Epidemic routing protocol, data could be evaluated in future to prevent the problems that occurred in this project. Also, it is built upon strong efforts in both ad hoc routing protocols and distributed consensus protocols. In real life scenarios, a hybrid approach that first attempts to use end-to-end ad hoc routing and falls back to Epidemic routing if a path is not available could be explored. Further, it may be possible to exploit the expected number of hops from source to destination to adaptively switch from epidemic to ad hoc routing with the expectation that a message has reached a connected network subset that includes the destination.

In this project both Probabilistic and Epidemic routing protocols have been evaluated using delivery ratio and latency as the metrics. However many other metrics such as number of forwarded messages could be incorporated for evaluation, which would then result in a more solid and a reliable evaluation.