

COMPARISON OF DIFFERENT ROUTING PROTOCOLS IN DTN

PROJECT REPORT

Submitted By

Rupdeep Saha, Roll No: 12618002044, Registration No:
181260110231

Sayan Banerjee, Roll No: 12618002047, Registration No:
181260110234

Under the Supervision of

Prof. Sandipan Dutta

Department of Information Technology

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

HERITAGE INSTITUTE OF TECHNOLOGY,KOLKATA

**MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY,
KOLKATA**

July, 2022

HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA

An Autonomous Institution under

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report on “COMPARISON OF DIFFERENT ROUTING PROTOCOLS IN DTN” is the bonafide work of “RUPDEEP SAHA and SAYAN BANERJEE”, who carried out the project under my supervision.


SIGNATURE

Prof. (Dr.) Siuli Roy

HEAD OF THE DEPARTMENT

Information Technology,

Heritage Institute of Technology,

Kolkata – 700107.


SIGNATURE

Prof. Sandipan Dutta


PROJECT MENTOR

Information Technology,

Heritage Institute of Technology,

Kolkata – 700107.

SIGNATURE


EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to take this opportunity to thank **Prof. (Dr.) Siuli Roy**, Head of the Department of Information Technology for giving us the opportunity to work on this project.

We would also like to thank **Prof. Sandipan Dutta**, our project mentor for constantly supporting us and guiding us throughout the project. Their guidance and words of encouragement motivated us to achieve our goal and impetus to excel.

We also thank the technical staffs of our department, Mr. Shantanu Ghosh for providing us with all necessary facilities required for this project.

Rupdeep Saha

Rupdeep Saha

Sayan Banerjee

Sayan Banerjee

ABSTRACT

In today's world, due to the internet, many devices (computers, Mobiles) are communicating with each other all around the world. For the humans better and easier life the technologies are increasing everyday communications. So, networks are emerging that are different and more massive. The networks (MANETs, VANETs, Wireless Sensor Networks (WSN)) have given new opportunities for the different types of applications. There are a number of networks used in connecting and communicating with different devices and it is only possible by some set of protocols known as TCP/IP protocol suite. Principle on which TCP/IP works is based on end-to-end data transfer. However, there are many regions where the assumptions of the internet cannot be satisfied. If in any environment where there is no path between the source and destination, then TCP/IP fails to work properly or might even stop working completely. Because of such circumstances, a newer network has evolved which is independent of end to end connectivity between nodes. This network is called Delay Tolerant Networks (DTN). In such networks, routing follows the well known store-carry-forward paradigm since an end-to-end path between two nodes is never guaranteed to exist. Through the evolution of various routing protocols in different scenarios using simulation tool comparative study can be done. This paper focuses on the existing routing protocol techniques and comparative study of existing routing protocols of Delay-Tolerant Networks based upon the metrics like Delivery Ratio and Overhead Ratio.

TABLE OF CONTENTS

<u>CHAPTER</u> <u>NO.</u>	<u>TITLE</u>	<u>PAGE</u> <u>NO.</u>
	ABSTRACT	4
1.	<u>Introduction</u> 1.1 Delay Tolerant Network 1.2 Importance of Delay Tolerant Network 1.3 Objective	7
2.	<u>Literature Survey</u>	9
3.	<u>Preliminary Ideas</u> 3.1 Epidemic Routing Protocol 3.2 Spray and Wait Routing Protocol 3.3 Prophet Routing Protocol 3.4 MaxProp Routing Protocol	13
4.	<u>Our Work</u> 4.1 Flow chart of entire process 4.2 Implementation of Epidemic Routing Protocol 4.3 Implementation of Spray and Wait Routing Protocol 4.4 Implementation of MaxProp Routing Protocol 4.5 Implementation of Prophet Routing Protocol	17

5.	<u>Result and Discussion</u> 5.1 Configuration Parameters 5.2 Analysis Chart/graph 5.3 Discussion of Results 5.4 Table of results	23
6	Conclusion 5.1 Conclusion 5.2 Future Work	28
7	References	30

1. INTRODUCTION

1.1 Delay Tolerant Network

Today's communication over the Internet is done by TCP/IP where an end to end path has been established and then a message is transferred from source to destination with high bandwidth and low delay. Also the message delivery probability is very high with a very low error rate. In Challenged Networks (such as Interplanetary Network, Military Battle Field, Sensor Network, Mobile Network) Communication where the destination is not always in direct touch with sender or far away from sender or having no Internet access TCP/IP scenario doesn't work. In this case, the Delay Tolerant Network concept will provide necessary facilities for data transfer.

The main difference between Internet and DTN communication is the absence of an end to end communication path which leads to disconnection, variable delay, and high error rate in communication. DTN uses the store and forward concept to send messages or packets from source to destination. DTN has various routing protocols based on knowledge or replication strategy for successful delivery of packet from sender to receiver. Protocols which work on knowledge of nodes or networks (such as location based routing, Gradient Routing, Link Metrics) decrease the delay but delivery probability is very low. On the other hand, the routing using replication of messages (such as in Direct Contact, Two way Hope, Tree Based routing, Epidemic Routing) delivery ratio can be increased but resource consumption is high. DTN uses a store, carry and forward approach. Nodes should carry the message until a proper custodian is not found. According to resource limitation each node has a fixed size buffer to store messages. Node stores the message in its buffer until the next custodian is found in the path towards the destination. As the buffer size is limited, nodes should follow some policy to decide which message is dropped when the buffer size is full.

1.2 Importance Of Delay Tolerant Network

Delay tolerant networks have many benefits when it comes to an environment where there is no availability of internet.

Benefits of Delay Tolerant Network are as follows :-

1. Operate effectively in extreme conditions.
2. Works with or without a terrestrial network.
3. Interoperability of ground stations, spacecraft.
4. More efficient data transmissions, usable bandwidth.
5. Improves link reliability.
6. More secure communications.
7. Improved Quality of Service (QoS).

Characteristics of DTN :-

1. Intermittent Connectivity
 - a. No end-to-end path between source and destination
2. Long Variable Delay
 - a. Long propagation delays between nodes
3. Dynamic Topology

1.3 Objective

The objective of the project is to compare and analyze the performance of different routing protocols present in delay tolerant networks. The routing protocols to be analyzed in this project are Epidemic Routing Protocol, Spray and Wait Routing Protocol, ProPhet Routing Protocol and MaxProp Routing Protocol. The parameters to be indulged for performance analysis are Delivery probability, Overhead ratio with varying transmission time in milliseconds. After successful performance analysis, the question “What is the most efficient routing protocol with varying time for transmission?” can be answered with this project.

2.

LITERATURE SURVEY

1. Performance Analysis of Delay Tolerant Network Routing Protocols in Different Mobility Environments

Authors:-

Bhed Bista (Iwate Prefectural University)

Danda B Rawat (Howard University)

Published in 2016 International Journal of Simulation: Systems, Science & Technology

They tried to provide a novel and efficient method of carrying out the performance analysis on other routing protocols based on DTN based on different map based movements. In this study, they perform analysis of three important DTN routing protocols to investigate their resource utilization, especially energy consumption under three different mobility environments. Three protocols they used are Epidemic Routing protocol, Prophet Routing protocol and Spray and Wait Routing protocol.

They concluded that there is no single mobility environment suitable for all DTN routing protocols. They recommended that Shortest path map based movement or random waypoint is best suitable for Spray and Wait Routing protocol, while only random way point is best suitable for Epidemic and Prophet Routing protocol.

Drawbacks:-

The major drawback in this paper is they only carried out the analysis based on different map movements. But, time is also a concern, that is, for how long the message transmission should carry on.

2. Routing based protocols in delay tolerant network: Survey study

Authors:-

Shaik Zahid Hussain Research Scholar, Department of Electrical and

Computer Engineering, International Islamic University, Malaysia

Shibab A Hameed Assistant Professor Department of Electrical and Computer Engineering, International Islamic University, Malaysia

Published in 2019 International Journal of Engineering Research & Technology (IJERT)

They stated that in order to understand the challenges and performance parameters of networks a rigorous survey is carried out in this paper with different routing protocols. The strategies and issues employed in routing protocols are widely discussed in terms of latency, data delivery and packet loss. The major problems addressed in the work deals with routing a packet from one node to the other using DTN. The major considerations of the network are discussed in the form of a novel approach to show the improvements in routing the data by matching the literature through an optimal scheme.

They concluded that the Epidemic and Prophet routing protocols as the most efficient one and achieves better performance. They also mentioned in their conclusion that the Prophet has a minimum overhead ratio.

Drawbacks:-

This study is based on qualitative notions of the protocols. A simulation based notion would have firmly established the conclusion drawn.

3. Performance Analysis of Prophet Routing Protocol under Random Mobility

Authors:-

P.SAMHITHA, K.POOJITH SINGH, M.NAVEEN, M.SAI SHANAKAR (Computer Science & Engineering, Lendi Institute of Engineering and Technology, Vizianagaram)

Published in 2020 International Journal of Engineering Research & Technology (IJERT)

They carried out their work efficiently based on a single routing protocol of Delay Tolerant Network. They set their map movement as Random Waypoint Mobility and carried out the performance analysis on different routing

parameters like delivery probability, Overhead ratio, etc.

They concluded that in this paper, they propose in enhancing the Performance of Prophet protocol using ONE simulator, which could reduce the detection overhead effectively. They showed that an appropriate probability setting could assure the security of the DTNs at a reduced detection overhead.

Drawbacks :-

After thoroughly going through their result case, it has been observed that their overhead ratio is still large enough for concluding it as a better overhead ratio. So, they should have taken a few different and recent routing protocols in consideration before providing their conclusion.

3. PRELIMINARY IDEAS

3.1 Epidemic Routing Protocol

The Epidemic Routing Protocol is a flooding-based forwarding algorithm for DTNs. In the epidemic routing scheme, the node receiving a message, forwards a copy of it to all nodes it encounters. Thus, the message is spread throughout the network by mobile nodes and eventually all nodes will have the same data. Although no delivery guarantees are provided, this algorithm can be seen as the best effort approach to reach the destination. Each message and its unique identifier are saved in the node's buffer. The list of them is called the summary vector. Whenever two adjacent nodes get the opportunity to communicate with each other, they exchange and compare their summary vectors to identify which message they do not have and subsequently request them. To avoid multiple connections between the same nodes, the history of recent contacts is maintained in the node caches.

Assuming sufficient resources such as node buffers and communication bandwidth between nodes, the epidemic routing protocol finds the optimal path for message delivery to destinations with the smallest delay. The reason is that the epidemic routing explores all available communication paths to deliver messages and provides strong redundancy against node failures. The major disadvantage of epidemic routing is wastage of resources such as buffer, bandwidth and node power due to forwarding of multiple copies of the same message. It causes contentions when resources are limited, leading to dropping of messages. It is especially useful in those conditions when there are no better algorithms to deliver messages.

Algorithm

1. Initialize the number of copies
2. Set the message for transferring
3. Copy the message
4. Deliver the copy of the message to the encountered node.
5. Once delivery confirmation is received, Set the transmitting message as delivered.
6. Remove the connection and again start rebuilding the connection for the next message
7. Stop.

3.2 Spray and Wait Routing Protocol :-

The Spray and wait routing protocol is an efficient routing protocol based on Delay Tolerant Network. It provides a very smooth transmission without degrading the

performance of the network. It is a fixed copy routing scheme, that means only a fixed number of messages will be routed. It reduces excess flooding of packets within the network in a very optimized manner, that is, a node with “n” copies of message would distribute half of the message, which is, $n/2$ copies to the next node it encounters, if $n > 1$ and forwards the message if the other node is the destination. It consists of two phases:

1. Spray Phase : It sprays “n” number of copies of the message from source node to nearby nodes which establishes successful connection with the source node.
2. Wait Phase : It is the waiting phase, till one of the relaying nodes meets with the destination node.

Algorithm

1. Initialize the number of copies
2. Set the message for transferring
3. Divide the message copies to shared by $n/2$ to the relay nodes (Spray Phase)
4. Next, wait for the message to be delivered (Waiting Phase)
5. If delivered, spread the message in the same way to the source node.
6. Once delivery confirmation is received, Set the transmitting message as delivered.
7. Remove the connection and again start rebuilding the connection for the next message
8. Stop.

3.3 PROPHET Routing Algorithm :-

The PROPHET routing protocol is a probabilistic routing protocol which uses history of encounter and transitivity. In this routing, it is assumed that the mobile nodes pass through some locations more than the others so passing through the previously visited location's probability is higher than the others so this approach is implied in the routing scheme. So, from that approach the node will meet the other node more preferably in the future which they met each other in the past.

The major concern of this routing protocol is to improve the delivery predictability and reduce the wastage of the resources in the epidemic routing. In this scenario, initially in the network it will estimate the probabilistic metric for each known node. Delivery predictability $(a,b) \in [0,1]$ for the node A for the each known destination B. Whenever a node encounters with the other nodes then they exchange summary vectors as same as the epidemic routing. This summary vectors contain the delivery predictability values for the destinations for each node. The calculations of delivery predictabilities of nodes have three parts. Nodes update their delivery predictability metrics whenever they meet each other. Visiting more nodes would result in high delivery predictability values. The calculation for that is shown in the following equation.

$$P(a,b) = P(a,b)_{old} + (1 - P(a,b)_{old}) \times P_{init} \text{ (Where } P_{init} \text{ is initialization constant)}$$

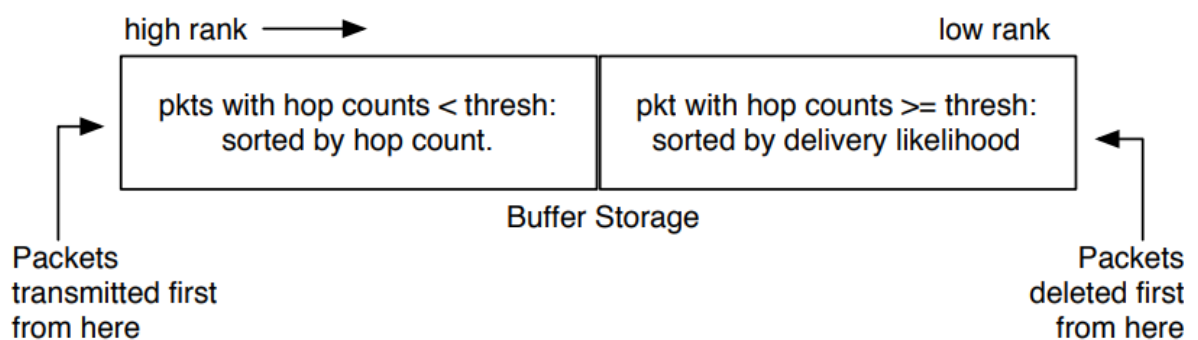
In the conventional routing protocols that discover the route and forward the packet to the destination on the basis of the shortest path or the lower cost, on the other hand, in the PROPHET whenever the node receives the message there is no path to the destination and node carries the message in the buffer and forwards it to the encountered node. The limitation of the PROPHET routing is that whenever the node meets with the low delivery predictability then there is no guarantee that it would meet the other with higher predictability value during the message lifetime. The basic difference of Prophet than Epidemic Routing is its forwarding strategy, when two nodes meet, Prophet allows the transfer of a message to the other node only if the delivery predictability of the destination of the message is higher at the other node.

Algorithm

1. Initialize the number of copies
2. Set the message for transferring
3. Put the message in buffer
4. Deliver the message to the encountered node.
5. Update delivery predictability values.
6. Remove the connection and again start rebuilding the connection for the next message
7. Stop.

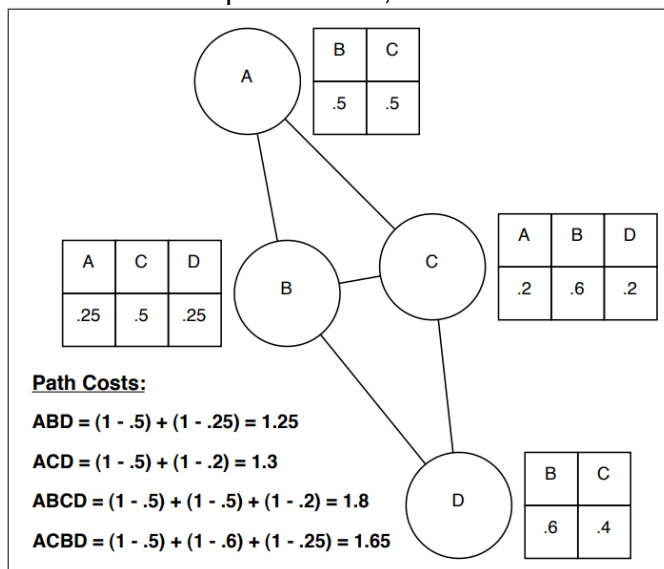
3.4 Max Prop Routing Protocol:-

MaxProp was developed at the University of Massachusetts, Amherst and was partly funded by DARPA and the National Science Foundation. MaxProp is flooding-based in nature, in that if a contact is discovered, all messages not held by the contact will attempt to be replicated and transferred. The main intelligence of MaxProp comes in when determining which messages should be transmitted first and which messages should be dropped first. MaxProp maintains an ordered queue based on the destination of each message, ordered by the estimated likelihood of a future transitive path to that destination. When two nodes meet each other, firstly, they exchange their estimated node meeting likelihood vectors. Preferably, each node will have an up to date vector from every other node. With these n vectors at hand, the node can compute the shortest path on the basis of a depth-first search where path weights indicate the probability that the link does not occur. To be noted, in MaxProp path cost calculation, the path cost ranges from 1 to 0. So, the highest value is 1. So, in a nutshell, MaxProp prioritizes both the schedule of packets to be transmitted (based on their hop count) and the schedule of packets to be dropped (based on their delivery likelihood) by splitting the buffer into two portions.



The above picture shows the MaxProp Routing strategy.

Let us take a simple scenario, for better understanding of the MaxProp shortest path selection.



Basic concept is to apply Dijkstra's method to find the shortest path from source to destination. The above example starts from source 'A' to destination 'D'.

Let us take any case from the above example:

If path $A \Rightarrow B \Rightarrow D$ selected, then;

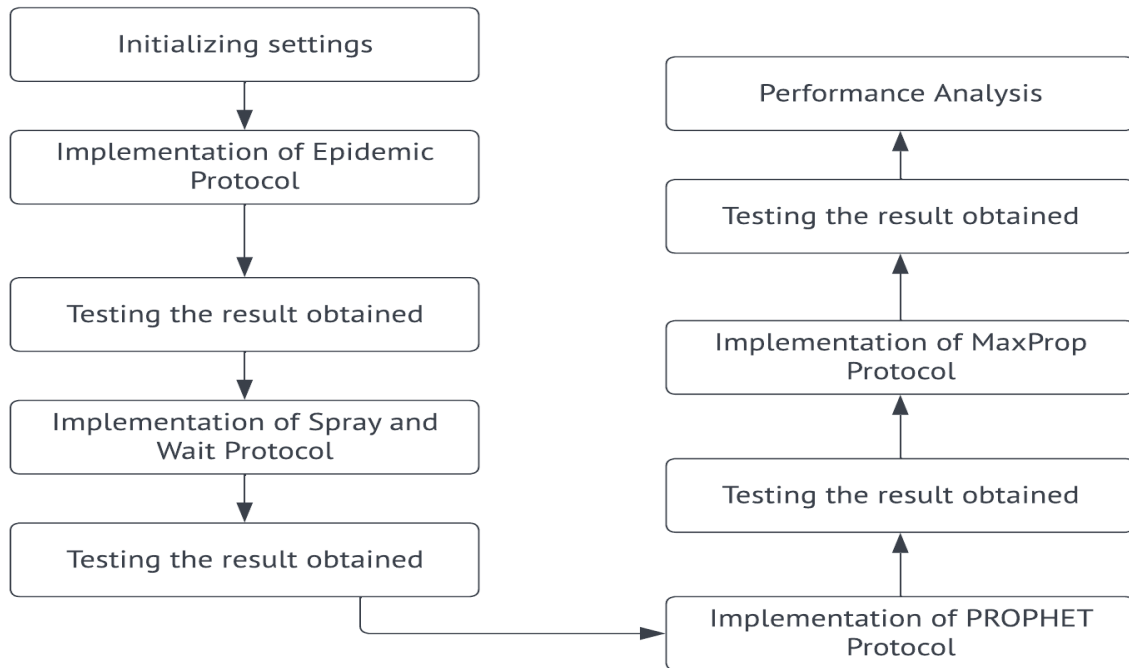
$A \Rightarrow B$ path cost = 0.5 and $B \Rightarrow D$ path cost = 0.25

So, total cost is $(1 - 0.5) + (1 - 0.25) = 1.25$.

Similarly, other paths are calculated and whichever path cost comes to be the least, that one is selected.

4. OUR WORK :

4.1 FLOWCHART OF THE ENTIRE PROJECT :



4.2 IMPLEMENTATION OF EPIDEMIC ROUTING PROTOCOL:-

1. Initiate the required settings

```
/**
 * Constructor. Creates a new message router based on the settings in
 * the given Settings object.
 * @param s The settings object
 */
public EpidemicRouter(Settings s) {
    super(s);
    //TODO: read&use epidemic router specific settings (if any)
}

/**
 * Copy constructor.
 * @param r The router prototype where setting values are copied from
 */
protected EpidemicRouter(EpidemicRouter r) {
    super(r);
    //TODO: copy epidemic settings here (if any)
}
```


2. Try to send the messages to the connected nodes. Here if the host is already transferring or cannot transfer then we do not proceed. Or else if the host is transferring the message to a final recipient then we do not proceed. Else, we try to send the message to all current connections.

```
public void update() {
    super.update();
    if (isTransferring() || !canStartTransfer()) {
        return; // transferring, don't try other connections yet
    }

    // Try first the messages that can be delivered to final recipient
    if (exchangeDeliverableMessages() != null) {
        return; // started a transfer, don't try others (yet)
    }

    // then try any/all message to any/all connection
    this.tryAllMessagesToAllConnections();
}
```

3. Repeat the process until simulation is completed.

4.3 IMPLEMENTATION OF SPRAY AND WAIT ROUTING PROTOCOL

1. Set the initial number of copies
2. Initiate the required settings

```
public SprayAndWaitRouter(Settings s) {
    super(s);
    Settings snwSettings = new Settings(SPRAYANDWAIT_NS);

    initialNrofCopies = snwSettings.getInt(NROF_COPIES);
    isBinary = snwSettings.getBoolean(BINARY_MODE);
}
```

3. Set the receive message functions, such that the message received can be put to the received pool with the host name.

```
public int receiveMessage(Message m, DTNHost from) {
    return super.receiveMessage(m, from);
}
```

4. Set the message transferring function with the required setting for the Spray and Wait, so the parameters are message id and from whom the message is coming.

```
public Message messageTransferred(String id, DTNHost from) {
    Message msg = super.messageTransferred(id, from);
    Integer nrofCopies = (Integer)msg.getProperty(MSG_COUNT_PROPERTY);

    assert nrofCopies != null : "Not a SnW message: " + msg;

    if (isBinary) {
        /* in binary S'n'W the receiving node gets ceil(n/2) copies */
        nrofCopies = (int)Math.ceil(nrofCopies/2.0);
    } else {
        /* in standard S'n'W the receiving node gets only single copy */
        nrofCopies = 1;
    }

    msg.updateProperty(MSG_COUNT_PROPERTY, nrofCopies);
    return msg;
}
```

5. After successful transfer, get the message and set it as done and reduce the number of copies of the message left with the host.

```
protected void transferDone(Connection con) {
    Integer nrofCopies;
    String msgId = con.getMessage().getId();
    /* get this router's copy of the message */
    Message msg = getMessage(msgId);

    if (msg == null) { // message has been dropped from the bu-
        return; // ..start of transfer -> no need to reduce am
    }

    /* reduce the amount of copies left */
    nrofCopies = (Integer)msg.getProperty(MSG_COUNT_PROPERTY);
    if (isBinary) {
        nrofCopies /= 2;
    }
    else {
        nrofCopies--;
    }
    msg.updateProperty(MSG_COUNT_PROPERTY, nrofCopies);
}
```

6. Stop or Repeat the process with a different number of copies of messages.

4.4 IMPLEMENTATION OF MAXPROP ROUTING PROTOCOL

The only difference in MaxProp routing protocol is the shortest path calculation after establishing connection with a bunch of nodes.

1. To obtain these estimated likelihoods, each nodes maintain a vector of $n-1$ (Where n is the number of nodes in the network)
2. Each $n-1$ nodes, is initially set to $\frac{1}{|n|-1}$, meaning the node is equally probable.
3. When the node meets another node, j , the j th element of the node is incremented by 1, and then the entire vector is normalized such that the total summation of the entries for that node is equal to 1.
4. When two nodes meet, they first exchange their estimated node-meeting likelihood vectors.
5. With these n vectors at hand, the node can then compute a shortest path via a depth-first search where path weights indicate the probability that the link does not occur.
6. These path weights are summed to determine the total path cost, and are computed over all possible paths to the destinations desired
7. The path with the least total weight is chosen as the cost for that particular destination.
8. The messages are then ordered by destination costs, and transmitted and dropped in that order.

```
private void updateTransitiveProbs(Map<Integer, MeetingProbabilitySet> p) {  
    for (Map.Entry<Integer, MeetingProbabilitySet> e : p.entrySet()) {  
        MeetingProbabilitySet myMps = this.allProbs.get(e.getKey());  
        if (myMps == null ||  
            e.getValue().getLastUpdateTime() > myMps.getLastUpdateTime() ) {  
            this.allProbs.put(e.getKey(), e.getValue().replicate());  
        }  
    }  
}
```

Here the transitive probabilities are updated, with the above code snippet.

```
public double getCost(DTNHost from, DTNHost to) {  
    /* check if the cached values are OK */  
    if (this.costsForMessages == null || lastCostFrom != from) {  
        /* cached costs are invalid -> calculate new costs */  
        this.allProbs.put(getHost().getAddress(), this.probs);  
        int fromIndex = from.getAddress();  
  
        /* calculate paths only to nodes we have messages to  
         * (optimization) */  
        Set<Integer> toSet = new HashSet<Integer>();  
        for (Message m : getMessageCollection()) {  
            toSet.add(m.getTo().getAddress());  
        }  
  
        this.costsForMessages = dijkstra.getCosts(fromIndex, toSet);  
        this.lastCostFrom = from; // store source host for caching checks  
    }  
  
    if (costsForMessages.containsKey(to.getAddress())) {  
        return costsForMessages.get(to.getAddress());  
    }  
    else {  
        /* there's no known path to the given host */  
        return Double.MAX_VALUE;  
    }  
}
```

The above code snippet returns the shortest path cost calculation for each node such that it can better be transmitted through the path with the least cost.

4.4 IMPLEMENTATION OF PROPHET ROUTING PROTOCOL

1. Initiate the required settings.

```
/**
 * Constructor. Creates a new message router based on the settings in
 * the given Settings object.
 * @param s The settings object
 */
public ProphetRouter(Settings s) {
    super(s);
    Settings prophetSettings = new Settings(PROPHET_NS);
    secondsInTimeUnit = prophetSettings.getInt(SECONDS_IN_UNIT_S);
    if (prophetSettings.contains(BETA_S)) {
        beta = prophetSettings.getDouble(BETA_S);
    }
    else {
        beta = DEFAULT_BETA;
    }

    initPreds();
}

/**
 * Copyconstructor.
 * @param r The router prototype where setting values are copied from
 */
protected ProphetRouter(ProphetRouter r) {
    super(r);
    this.secondsInTimeUnit = r.secondsInTimeUnit;
    this.beta = r.beta;
    initPreds();
}
```

2. Initiate the predictability values

```
/**
 * Initializes predictability hash
 */
private void initPreds() {
    this.preds = new HashMap<DTNHost, Double>();
}
```

3. Update predictability values on encountering other nodes

```
/**
 * Updates delivery predictions for a host.
 * <CODE>P(a,b) = P(a,b)_old + (1 - P(a,b)_old) * P_INIT</CODE>
 * @param host The host we just met
 */
private void updateDeliveryPredFor(DTNHost host) {
    double oldValue = getPredFor(host);
    double newValue = oldValue + (1 - oldValue) * P_INIT;
    preds.put(host, newValue);
}
```

4. Updating transitive predictabilities

```

/**
 * Updates transitive (A->B->C) delivery predictions.
 * <CODE>P(a,c) = P(a,c)_old + (1 - P(a,c)_old) * P(a,b) * P(b,c) * BETA
 * </CODE>
 * @param host The B host who we just met
 */
private void updateTransitivePreds(DTNHost host) {
    MessageRouter otherRouter = host.getRouter();
    assert otherRouter instanceof ProphetRouter : "PROPHET only works " +
        " with other routers of same type";

    double pForHost = getPredFor(host); // P(a,b)
    Map<DTNHost, Double> othersPreds =
        ((ProphetRouter)otherRouter).getDeliveryPreds();

    for (Map.Entry<DTNHost, Double> e : othersPreds.entrySet()) {
        if (e.getKey() == getHost()) {
            continue; // don't add yourself
        }

        double pOld = getPredFor(e.getKey()); // P(a,c)_old
        double pNew = pOld + ( 1 - pOld) * pForHost * e.getValue() * beta;
        preds.put(e.getKey(), pNew);
    }
}

```

5. Age all entries in delivery predictions.

```

/**
 * Ages all entries in the delivery predictions.
 * <CODE>P(a,b) = P(a,b)_old * (GAMMA ^ k)</CODE>, where k is number of
 * time units that have elapsed since the last time the metric was aged.
 * @see #SECONDS_IN_UNIT_S
 */
private void ageDeliveryPreds() {
    double timeDiff = (SimClock.getTime() - this.lastAgeUpdate) /
        secondsInTimeUnit;

    if (timeDiff == 0) {
        return;
    }

    double mult = Math.pow(GAMMA, timeDiff);
    for (Map.Entry<DTNHost, Double> e : preds.entrySet()) {
        e.setValue(e.getValue()*mult);
    }

    this.lastAgeUpdate = SimClock.getTime();
}

```

6. Repeat the process until the simulation is completed.

5. RESULTS & DISCUSSION

5.1 CONFIGURATION PARAMETERS

To test the different routing protocols and obtain their delivery probability and overhead ratios, we have to edit the default_settings configuration file every time. In our project, we have tested the different protocols for different simulation times namely 5000ms, 10000ms, 15000ms, and 20000ms. And we also have to define the movement model on which each protocol will run.

```
1#
2# Default settings for the simulation
3#
4
5## Scenario settings
6Scenario.name = default_scenario
7Scenario.simulateConnections = true
8Scenario.updateInterval = 0.1
9# 43200s == 12h
10Scenario.endTime = 20000
11
12
13
14
15
16
17
18
19
20
21
22
23
24# Common settings for all groups
25Group.movementModel = ShortestPathMapBasedMovement
26Group.router = MaxPropRouter
27Group.bufferSize = 5M
28Group.waitTime = 0, 120
29# All nodes have the bluetooth interface
30Group.nrofInterfaces = 1
31Group.interface1 = btInterface
32# Walking speeds
33Group.speed = 0.5, 1.5
34# Message TTL of 300 minutes (5 hours)
35Group.msgTtl = 300
```

Certain routing protocols also have some default configurations

```
157## Default settings for some routers settings
158ProphetRouter.secondsInTimeUnit = 30
159SprayAndWaitRouter.nrofCopies = 6
160SprayAndWaitRouter.binaryMode = true
```

5.2 Discussion:

Epidemic Routing Protocol

Time(in milliseconds)	Delivery Probability	Overhead Ratio
5000	0.1667	100.67
10000	0.212	86.79
15000	0.22	87.16
20000	0.21	96.66

As we can see in Epidemic Routing Protocol, the delivery probabilities keep increasing until it gets to a constant value as we increase the simulation times. Overhead Ratio doesn't show such a trend, but both delivery probability and overhead ratio are the worst in Epidemic Routing as compared to other routing protocols.

SprayAndWait Routing

Time(in milliseconds)	Delivery Probability	Overhead Ratio
5000	0.2262	17.02
10000	0.33	13.29
15000	0.389	11.74
20000	0.4133	11.16

In our testing, Spray and Wait gives a good delivery probability and the lowest overhead ratio. Also in SprayAndWait routing, the delivery probability increases with increase in simulation time and Overhead ratio decreases. Although SprayAndWait Routing is just an extension of Epidemic Routing, it can provide some good results in some scenarios.

MaxProp Routing

Time(in milliseconds)	Delivery Probability	Overhead Ratio
5000	0.2619	60.04
10000	0.4071	42.63
15000	0.459	40.41
20000	0.5	36.76

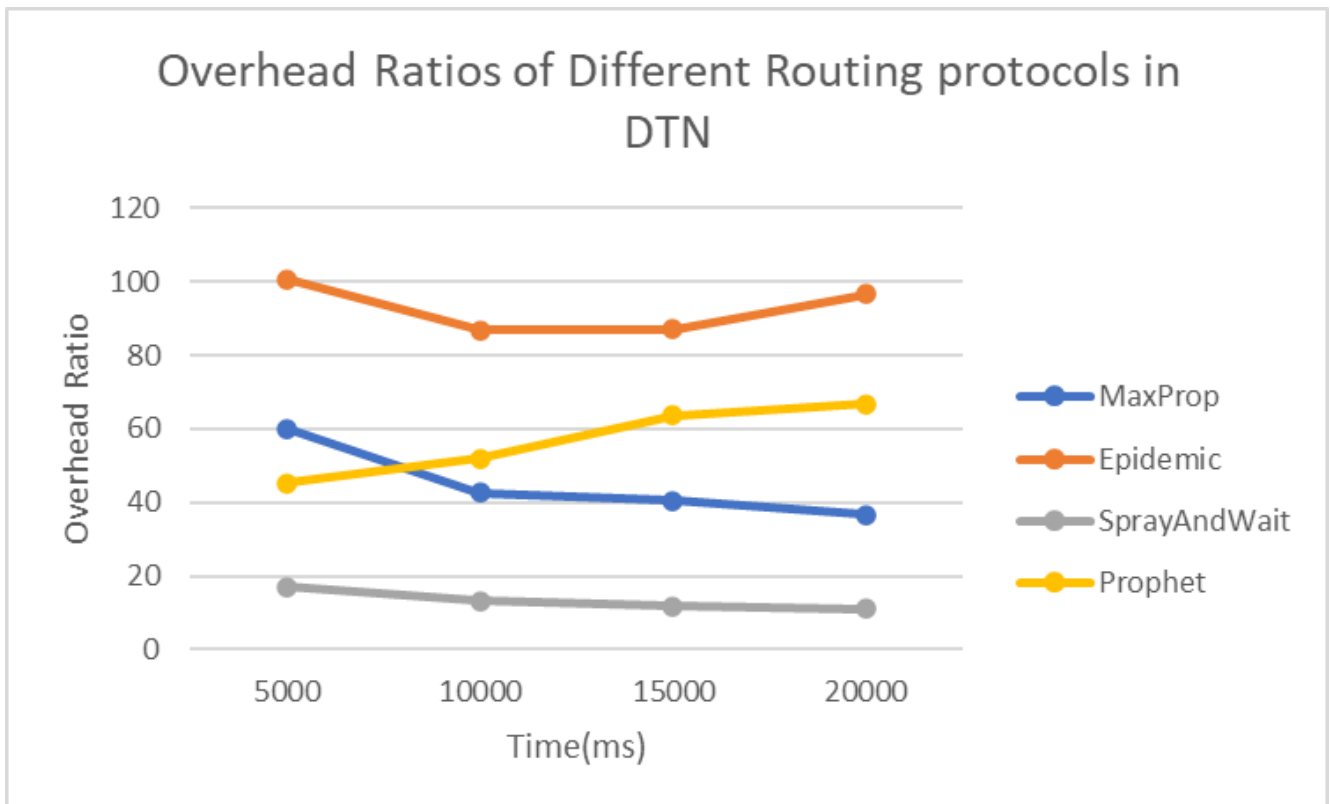
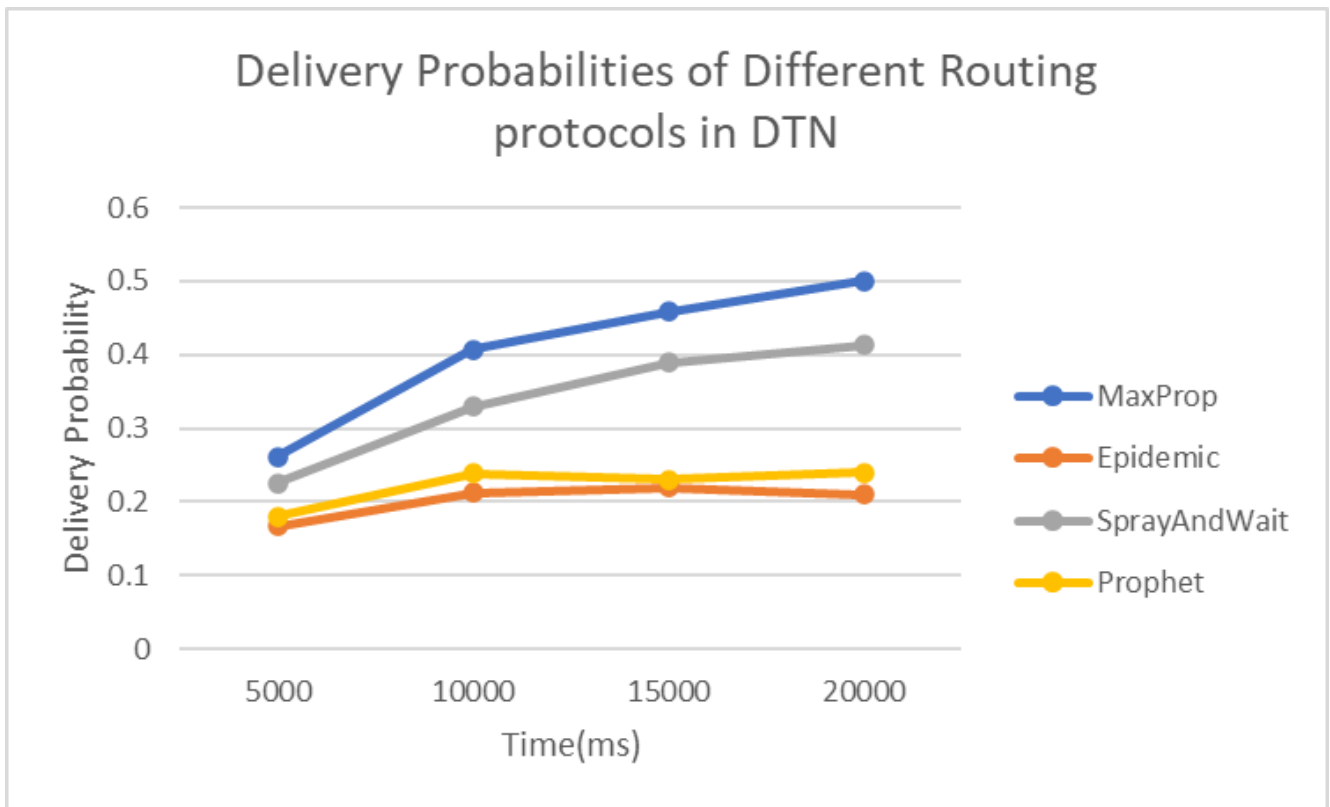
In MaxProp Routing Protocol, the delivery probability is the highest among all other routing protocols and the overhead ratio is also very low. Also the delivery probability increases with increase in simulation time and overhead ratio decreases with increase in simulation time. Therefore, MaxProp is the one of the most efficient routing protocols and its effectiveness increases with increase in simulation time.

Prophet

Time(in milliseconds)	Delivery Probability	Overhead Ratio
5000	0.18	45.22
10000	0.238	51.82
15000	0.23	63.71
20000	0.24	66.7

In our testing, PROPHET routing gives slightly better delivery probability than Epidemic routing but lesser than SprayAndWait routing. It also gives slightly better overhead ratio than epidemic but it is worse than both MaxProp and SprayAndWait. And also generally delivery probability increases with increase in simulation time, but overhead ratio also increases.

5.3 Obtained Analysis Graph:-



5.4 Table Of Results :

Routing Protocols	Delivery Probability (Time = 5000ms)	Delivery Probability(Time = 10000ms)	Delivery Probability (Time = 15000ms)	Delivery Probability (Time = 20000ms)	Average
MaxProp	0.2619	0.4071	0.459	0.5	0.407
SprayAndWait	0.2262	0.33	0.389	0.4133	0.34
Epidemic	0.1667	0.212	0.22	0.21	0.202
Prophet	0.18	0.238	0.23	0.24	0.22

Routing Protocols	Overhead Ratio(Time = 5000ms)	Overhead Ratio(Time =10000ms)	Overhead Ratio(Time = 15000ms)	Overhead Ratio(Time = 20000ms)	Average
MaxProp	60.04	42.63	40.41	36.76	44.96
SprayAndWait	17.02	13.29	11.74	11.16	13.30
Epidemic	100.67	86.79	87.16	96.66	92.82
Prophet	45.22	51.82	63.71	66.7	56.86

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this project, we have discussed about Delay Tolerant Networks and their routing protocols. We also have compared 4 routing protocols Epidemic routing, Spray and Wait routing, MaxProp routing, and Prophet routing. After testing the 4 routing protocols under different simulation times, we have come to the conclusion that MaxProp and SprayAndWait are the best routing protocols among the ones tested and they provide the best delivery probabilities and the lowest overhead ratios. We also conclude that the algorithms get more efficient with an increase in simulation time.

6.2 FUTURE SCOPE

Further in this project, we can improve the performance of the MaxProp Algorithm in a most defined and prominent way. If we can incorporate the message distribution technique similar to that of spray and wait protocol, that is, passing $n/2$ number of messages to the next node, keeping the other things in MaxProp algorithm constant, we can achieve a much more efficient algorithm by adding the logic of message distribution like spray and wait will reduce the overhead ratio of MaxProp routing protocol largely.

References:-

- [1] Nirbhay Kumar Chaubey(Ganpat University), Pooja Mistri Routing Protocols in Delay Tolerant Network (DTN): A Critical Study published in 2016 International Journal in IT and Engineering
- [2] Mr. Nikunj A. Dudharejiya(PG Student, CE dept., C. U. Shah College of Engineering and Technology. Wadhwan City, Gujarat, India) , Mr.Ashish V. Nimavat(Assistant Professor, CE dept., C. U. Shah College of Engineering and Technology. Wadhwan City, Gujarat, India) ,Mr.Jay M.Patel (Assistant Professor, CE dept., C. U. Shah College of Engineering and Technology. Wadhwan City, Gujarat, India): Comparative study of routing protocols in DTN published in 2014 International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
- [3] Biren Patel (Assistant Professor, Department of Computer Science, Ganpat University, Ganpat Vidyanagar, Gujarat, India.), Dr. Vijay Chavda (Principal, N P Patel College of Computer Studies and Management, Kadi, Gujarat, India): Study of DTN Routing Protocols published in 2015 International Journal of Advanced Research in Computer and Communication Engineering
- [4] P.SAMHITHA, K.POOJITH SINGH, M.NAVEEN, M.SAI SHANAKAR (Computer Science & Engineering, Lendi Institute of Engineering and Technology, Vizianagaram) Published in 2020 International Journal of Engineering Research & Technology (IJERT): Performance Analysis of Prophet Routing Protocol under random mobility
- [5] Shaik Zahid Hussain Research Scholar, Department of Electrical and Computer Engineering, International Islamic University, Malaysia, Shibab A Hameed Assistant Professor Department of Electrical and Computer Engineering, International Islamic University, Malaysia Published in 2019 International Journal of Engineering Research & Technology (IJERT): Routing based protocols in Delay Tolerant Networks: Survey Study

[6] Bhed Bista (Iwate Prefectural University), Danda B Rawat (Howard University)
Published in 2016 International Journal of Simulation: Systems, Science &
Technology: Performance Analysis of Delay Tolerant Network Routing Protocols in
Different Mobility Environments