An Improved Prophet - Machine Learning Based Optimized Multi-Copy Routing Algorithm for IoT and Opportunistic Mobile Networks

Srinidhi N N, Sagar C S, Deepak Chethan S, Vinay G P, Vaishak, Dilip Kumar S M†

†*Dept. of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore, India*
srinidhinagesh@gmail.com

***Abstract*—Opportunistic networks are one of the important categories of adhoc networks in Internet of Things (IoT), which exploits human social characteristics like daily routines, similarities etc., to facilitate sending and receiving of messages. In Opportunistic networks, mobile nodes are used to establish communication between nodes despite of non-availability of a dedicated route between them. Furthermore, nodes don't acquire any knowledge in advance about the characteristics of the network such as the network topology, the location of the other nodes. Hence, designing a routing algorithm becomes a challenging task since traditional routing protocols used in the Internet are not feasible for the characteristics inherent in this type of network. The proposed work propounds a multi-copy routing algorithm based on machine learning named iProphet or improved Prophet. iProphet, uses dynamically changing contextual information of nodes and the delivery probability of Prophet to carry out message transfer. The multi-copy routing algorithm uses a machine learning classifier known as random forest to classify the node as a good forwarder or a bad forwarder based on the supplied contextual information during each routing decision. The classifier trained with large amount of data extracted using simulation leads to precise classification of the nodes as reliable or unreliable for carrying out the routing task. The simulation results show that the proposed algorithm outperforms with respect to delivery probability, hop count, overhead ratio, latency but over costs with respect to average buffer time in par with similar multi-copy routing algorithm. The uniqueness of this paper lies in data extraction, categorization and training the model to obtain reliable and unreliable nodes to provide efficient multi-copy routing in IoT scenario.**

***Index Terms*—Internet of Things, Machine Learning, Mobility, Multi-Copy, Random Forest**

I. Introduction

With the development in technogies, more and more devices are getting connected to the Internet, which is heading to the new criterion in computing and communication known as the Internet of Things (IoT) [1]. As a result, more devices are getting prevalently monitored and cannot be deployed statically. These IoT devices while being transported by humans or different carriers will interact either statically or dynamically. For instance, in vehicular based IoT network statically deployed devices may necessitate to recognize the mobile agents and employ them in assembling and delivering their information to the intended target [2]. Along the way opportunistic mobile networks [3] rose as a novel mechanism of transmissions in wireless systems. Unlike mobile ad hoc networks (MANETs) that needs end-to-end connection routes for information transfer, the communication in opportunistic mobile networks takes place on the corporation of opportunistic connections among mobile nodes, without the availability of end-to-end communication paths. These mobile devices establishes connection when people comes in contact and these network type are considered as human social networks. Hence, the opportunistic networks utilize the individual behaviors and social relations to provide more effective and reliable message distribution systems. In OppIoT [4], data acquisition, propagation, and distribution of data which typically necessitate opportunistic connections utilizing mobility, short-range transmission, and technologies. During such operations, the configuration of the routing protocol is a challenging task due to this inherent complexity in securing connectivity among devices and recognizing a relevant data packet forwarder that can convey the packet to its desired destination. These networks are the derived class of OppIoT and considered as evolved form of MANETs. In OppNets, nodes are allowed to communicate with several other nodes even though a routing connecting does not exist between them. Nodes are connected momentarily and the routes keep dynamically changing due to nodes mobility or frequent nodes activation and deactivation. When the packets are en route between the source node and the target node(s), any viable node can be opportunistically employed as the next hop, granted it is plausible to bring the packet closer to the intended destination. Several protocols related to OppNets that have been proposed such as epidemic, HBPR, HibOp etc., which uses parameters like message propagation pattern, context erudition of nodes (i.e., current position, the possibility of delivery, target node distance, prior and existing history of the node), last synergy time and various other parameters in order to predict which node is opted as next hop to deliver the message to the intended destination. In this work, a machine learning based algorithm which can be employed to OppNets is proposed. The proposed algorithm that differ radically from the existing algorithm as this algorithm uses the random forest to decide the neighboring node to route the data packet based on the present context and location information. Random forest or otherwise referred to as random decision forest are one of the most popular ensemble learning techniques that are used in both classification and

regression. Several decision trees are randomly created with several subsets of data during the training period. The class is assigned based on the mode. Random forest algorithm uses the technique of bootstrap aggregation, which helps in improving the stability and accuracy of the algorithm along with reducing the variance and overfitting of the proposed iProphet.
The rest of the paper is organized as follows. Section II presents a summary of various existing works with respect to multi-copy and oppurtunisitic networks. Problem statement is presented in Section III. Proposed algorithm has been presented in section IV. The experimental evaluation and results has been explained in section V. Conclusion along with future work has been presented in Section VI.

II. Related Work

Many protocols that are used in OppNets are related to the nodes context-information and message distribution. Representative ones are summed as follows. Epidemic routing [5], is a flood-based routing algorithm whose foremost objective is to deliver the packet with high probability to the desired destination with a minimum premise about the topology and the network. The work is based on epidemic algorithm which manages two buffers one for the message originating from the node itself and the other buffer is used to store messages generated from different nodes. Each node contains a summary vector that carries a short description of messages currently cached in the buffer. When nodes come in contact of one another, they revise their summary vectors and each node has an extra buffer to prevent verbose connection with other nodes. Nodes exchange their summary vector with each other after a predefined time and after exchanging their vectors the nodes rival their vectors to ascertain which message is missing. Then, the nodes request for the copies of the messages which they do not contain. But the proposed epidemic protocol uses higher bandwidth and buffer space which will cause network congestion and it do not examine nodes parameters to predict the possibility of message delivery. The HBPR (History Based Prediction for Routing), protocol which was proposed by Dhurandher et al. [6], uses election of the intermediate node as the message carrier is decided based on specific parameters. These parameters are the duration for which a pair of nodes meet, node's history, time at which a couple of nodes meet, and the path of the mobility of the node concerning the root and the target node. The schema consists of three stages, i.e., identifying the home location, wherein the nodes movement is utilized to recognize the subsequent mode location; the message production stage, during which the intermediate node is determined, and also the home location is renewed and lastly, the Subsequent hop selection phase, where metric is determined to help to judge on the assortment of the best succeeding forwarder of the message. This protocol has been confirmed to produce improved results than Epidemic concerning the message delivery number and Overhead ratio, but considering the individual mobility paradigm only. Prophet routing protocol [7], uses non-arbitrary movement and association patterns in practical application situation to replicate the message to different nodes in order to enhance the routeing performance. It is established on the basis that if a node has communicated with a different node or visited a region regularly, then there is a higher likelihood of revisiting the region or the node. In order to accomplish this, a delivery probability is defined for every node which associates itself with other nodes. Let assume that a node T wants to remit a packet to node U if this node T comes in association with another node R which has a higher delivery probability to node U compared to T only then a copy of the packet is copied to the node R. The deliverability is established based on the buffer size aggregate, position, popularity and predictability of delivery. If a source node comes in association concurrently with various nodes the message is copied to the node which has the highest value of deliverability. In distance-based Prophet, routing algorithm distance is used as an added factor. Here each node containing the packet checks the distance from all the bordering nodes and picks the nodes with is closed as the packet forwarder. Thereby improving the probability of delivery and lessening the delay probability. Prowait routing algorithm which was proposed by Dhurandher et. al. [8], can be considered as an amalgamation of the Prophet and Spray and wait routing protocol. Prowait model utilities an uncomplicated forwarding schema, in which when two nodes engage, the transaction of packet takes place from the node with lower probability of delivery to the node have higher probability concerning the desired destination. They mainly weigh a couple of parameters during packet transmission 1) election of an adjacent node 2) choosing the node for the next hop. For the election of the node, the Prophet protocol is used to estimate delivery predictability and transferring of the data is done using the spray and wait routing protocol. Choosing the node for the next hop is done using the Prophet routing algorithm. Prowait routing algorithm keeps the replicate data packets so that the packet is delivered to the intended destination with minimum delay. It reduces the usage of resource wastage by depreciating the amount of flooding compared to the epidemic protocol. It, however, does not predict forwarding of the message packet in order to improve delivery delay in the network. HiBO protocol (history-based routing protocol for OppNets) was proposed by Boldrini et. al. [9]. which utilizes the present context statistics about the nodes to ascertain the suitable node to convey the message to the desired destination. The context statistics are collected from the simulation environment. When the node desire to transmit the message, the node utilizes the History-Table and Identity-Table to decide, whether there is a union between the context statistic of its adjacent nodes and the data associated with the message. In that case, the data packet is transmitted to the adjacent node for which the union occurred. This routing algorithm will have higher delay in deciding which is the neighboring node and requires additional effort in maintaining the tables. EDR(encounter and distance based routing algorithm) [10], protocol uses context statistics to convey the message from the source to the destination. Here the routing protocol determines dynamically contact value for every couple of nodes (i.e. the frequency at which a pair of nodes meet). Moreover, the Euclidean distance is also estimated dynamically for every couple of nodes. A parameter called the best-forwarder is evaluated based on

the values considered earlier and the packet is transmitted to those intermediate nodes only if their forwarder value is more than the threshold. As it is context related protocol, it does not function well concerning packet delivery probability and overhead ratio. A Markov chain prediction model proposed by Jeon and Lee [11], in which given nodes mobility behavior is outlined on the given nodes former mobility information state to anticipate the nodes expected movement. Using this data, mobile nodes packet delivery ratio to destination node is calculated. The proposed protocol enhances the delivery ratio and reduces the average delay compared to conventional routing protocols for OppNets. However, this protocol performs inadequately compared to other mobility models like a random waypoint. MLProph [12], which is an infrastructure-less OppNet routing protocol utilizes a machine learning procedure to decide preceding network information and nodes parameters like the size of the buffer, hop-count, momentum and delivery. Based on these parameters, node likelihood as a forwarder is determined. However, it does not take into account of node's message delivery probability which is prime parameter to be considered during routing.
However, in the proposed model random forest algorithm is considered to decide the neighboring node to route the data packet based on the present context and location information. Several decision trees are randomly created with several subsets of data during the training period, which helps in improving the stability and accuracy of the algorithm along with reducing the variance and overfitting.

III. Problem statement and challenges

A. Problem statement

There are various challenges that opportunistic network in IoT faces. This is due to the dynamic nature of nodes over time depending on the application, which makes it difficult for routing the information and to identify the intermediate nodes which are used for forwarding messages to destination.

1) The problem is to classify the node as a good forwarder or a bad forwarder based on the supplied contextual information during each routing decision in order to optimize delivery probability, hop count, overhead ratio, latency.
2) To train the random forest classifier with large amount of data extracted using simulation to precise the classification of the nodes as reliable or unreliable for carrying out the routing.

B. Challenges

- Data extraction: For training the machine learning based random forest model, a huge data set had to be obtained. Trying to get data that is close to the real world was a challenge. ONE simulator provided the closest solution. To extract the data from the ONE simulator, its source code had to be modified.
- Model Creation: With the data extracted from ONE simulator, a robust random forest classifier had to be constructed. For doing which, features and algorithm related parameters had to be properly chosen.
- Model Integration: After the model is constructed, it had to be properly integrated with the ONE simulator to classify the nodes.

IV. Proposed Algorithm

This section provides in detail explanation of the proposed iProphet algorithm.

A. Terminology

- S: Source node
- D: Destination node
- X: Selected forwarder node
- G: Set of all neighboring nodes
- N_i: Neighboring node i in the set S
- M_i: Message
- n: Current node, the node tasked with making a routing decision
- M: Set of all messages that were generated during data generation phase
- M_s: Set of messages that were delivered
- M_d: Set of messages that were dropped or aborted
- E: Set of all the events recorded during data extraction phase of Prophet
- $P(N_i)$: Delivery probability as calculated by Prophet for the neighbor node Ni by current node n
- $feature\_name(x)$: Value of the attribute *feature_name* in the record x in any set.

Consider N nodes forming an opportunistic network with a means to communicate and having sufficient energy. Every node has sufficient buffer to store the state information like the nodes movement information, current location, delivery probability, direction, distance between source and destination, past history, power status. All these attributes can be used to determine the route for the message transfer. $Prophet$ (Probability routing protocol using history of encounters and transitivity) is one of the algorithm that uses previous contact history between the nodes to calculate the delivery probability between them. This is used to deliver the packets to the destination. Even though this provides a sufficient performance, the method to select the next node could be optimized by application of machine learning models. Prophet does not completely utilize all the context information present in the nodes. With the data driven approach, we can choose the best predictors/features and use them to drive the routing decisions. In the proposed Machine Learning Based Optimized Multi-Copy Routing Algorithm for IoT and Opportunistic Mobile Networks, random forests algorithm is used to choose the neighboring node to route the data packet based on the present context and location information. Random forest or otherwise referred to as random decision forests are one of the most popular ensemble learning techniques that are used for both classification and regression. Several decision trees are randomly constructed with varied subsets of data during the training period. The class is assigned based on the mode. Random forest algorithm uses the technique of bootstrap aggregation, which helps in improving the stability and accuracy of the algorithm along with reducing the variance and overfitting.

Formally, consider a training set $X = x_1, ..., x_n$ along with their corresponding outputs $Y = y_1, ..., y_n$ giving dataset $D_n = (X_1, Y_1), ..., (X_n, Y_n)$. Random forest is a collection of random base regression trees $\{rn(x, \theta_m, D_n), m \geq 1\}$, where $\theta_1, \theta_2, ...$ are outputs of a randomizing variable θ. The outputs of these random trees are aggregated to get an estimate. Thus, random forest model is given by,

$$\bar{rn}(X, D_n) = E_\theta[rn(X, \theta, D_n)] \quad (1)$$

Where E_θ is the expectation with respect to θ. The dataset is divided into various subspaces based on this randomizing variable theta. It is also used to find out on what basis the successive cuts are done when constructing the individual trees. During the learning period, the algorithm will choose a random subset of features, called as feature bagging. If a given feature is a strong predictor, it is used repeatedly in several different trees. To find the optimal number of trees, cross-validation techniques is applied. The main advantage of Random forest over normal decision trees is that the predictions from single tree are susceptible to noise. But the aggregation of forest will overcome from this drawback. Random forest algorithm can also be viewed as weighted neighborhood schemes. Here random Forests algorithm is be used to classify the neighboring nodes as good and bad forwarders based on the contextual information. Our proposed algorithm has mainly three stages: (1) Collecting routing data from one simulator (2) Building and training the model (3) Application of the model

B. Collecting routing data from one simulator

ONE simulator is one of the best simulation environment for simulating opportunistic mobile networks. For this stage, modification of the one simulator source was done to extract the necessary information to train the algorithm. A total of 4 datasets are generated, they are.

1) Event Dataset: This dataset is used to describe every event that took place inside the simulation environment. An event is a discrete piece of entity in a discrete-event based simulation system. In our context an event represented as activity inside the environment such as relaying of the message. The features of this dataset contains the following information in columns and rows.
   a) Key: Used to identify the inter related records in various datasets.
   b) $current\_node$: This is the node which contains message related to routing decision.
   c) $destination\_node$: The node to which the message has to be delivered.
   d) $neighbor\_nodes$: The list of all the node ids which are in range to the current node.
   e) $message\_id$: id of the message which is to be forwarded.
   f) $selected\_forwarder$: The node selected as the forwarder for the message in Prophet algorithm.
   g) $predictability\_of\_the\_current\_node$: Delivery prediction for the current node as calculated by the original Prophet algorithm.
   h) $predictability\_of\_the\_selected\_forwarder$: Delivery prediction for the selected forwarder based on which it was selected as a best forwarder.
2) Message Dataset: This dataset contained rows that described the details of the message like its source, intended destination, message size, the traveled path. An important feature is it contains *delivery status*, which is used to know whether the message was finally delivered to the intended destination or whether it was aborted in halfway. Based on this feature, the dataset was further categorized into three parts:
   a) Delivered: This dataset contained messages that are ultimately delivered to the final destination.
   b) Aborted or dropped: Messages that are dropped during routing.
   c) Relayed: Messages that are transferred to intermediate nodes.

   The features of this dataset contains the following information as columns of the row:
   - key: For event dataset as described earlier.
   - $message\_id$: A unique identifier to identify the message.
   - $message\_size$: Size of the message in bytes.
   - $source$: Node from which the message is generated.
   - $destination$: Node to which intended message should be delivered.
   - $number\_of\_hops$: Number of nodes traversed in its path to destination.
   - $message\_creation\_time$: Time during which message was created in simulated seconds.
   - $message\_deletion\_time$: Time during which the message was deleted in simulated seconds.
3) Event meta-data dataset: This dataset described every entity like the source node, the current node, intermediate node, selected forwarder node that were involved in every event in the event dataset.
   a) Host info: Host info of the node is the set of information about the node at that particular point of event time. This includes nodes x coordinate, y coordinate, buffer size, next time to move, free buffer space, nodes speed recorded in 9 columns.

   The features of this dataset contains the following information as columns of the row for every event recorded.
   - key - as described for the above dataset
   - Host info of current node
   - Host info of destination node
   - Host info for all the neighbors
   - Id of the selected forwarder
   - message_id

   At the end of this stage, number of datasets containing sufficient number of rows were generated to pre-process the data and to train the algorithm.

C. Building and Training the Model

After the datasets are extracted to suitable form, pre-processing of data is performed on the event dataset to obtain

an extra feature called *delivery status*, which provides an extra feature in the form of label to the proposed algorithm to predict while making routing decisions. The label is a categorical variable with two values as *delivered* and *not delivered*. A label of *delivered* for a row in the event dataset indicates that the message that was relayed as part of the event has delivered successfully and *not delivered* indicates that this event didn't lead to a successful delivery of the message. The following section, provides a brief about how the labeling is done.

1) Messages in the *delivered* category of message dataset were considered. The path feature contains the list of every node the message traveled in an ordered fashion till it reached the destination. Let's say that two of the successive nodes in the path for the message M were node n1 and n2 with key k. This indicates that message was forwarded from node n1 to node n2 in some event that was recorded in the event dataset.
2) Using the successive pair of nodes from the message dataset, the event dataset should be labelled with the label *delivered(1)* or *not delivered(0)*. Let the pair of nodes be n1 and n2 in the path list of the message M. Let the event corresponding to such a transfer of message M from n1 and n2 contain key K. i.e, the current node and selected forwarder in the event dataset are n1 and n2 respectively with the key K. Such an event is labelled as *delivered*. This process is repeated for all the messages in the message dataset which labels the corresponding events in the event dataset. The remaining events in the event dataset after exhausting all the messages are labelled as *not delivered*.

Algorithm 1 describes the pseudocode for labelling the dataset. The random forest classifier is trained with the event dataset obtained after applying algorithm 1 up to an higher degree of accuracy without overfitting or underfitting. The trained random forest classifier is then used while making each routing decision to identify the fit nodes i.e classification label = 1(delivered) and to therefore forward the messages to them.

D. Usage of Random Forest in Proposed Algorithm

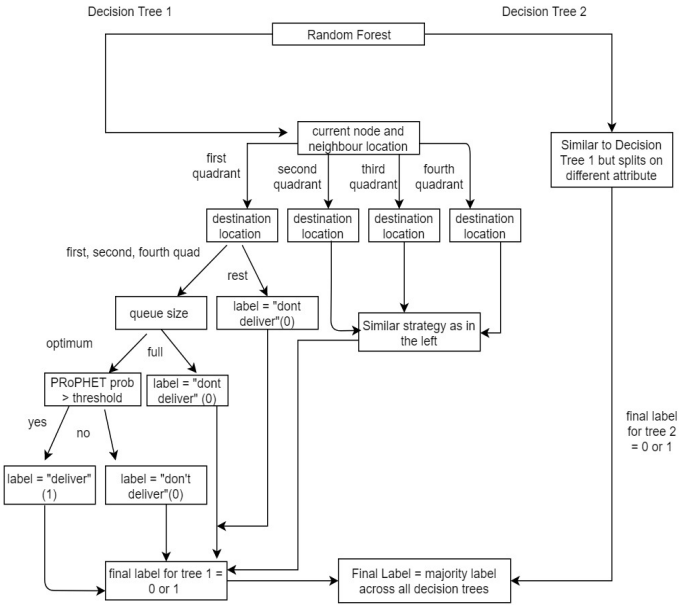

Fig. 1: Random forest architecture

Random Forest is one of the most popular machine learning algorithms for both classification and regression. The existing Prophet algorithm does not take advantage of the available past data, it only uses the current context information in its static model. Thus, it could be made more robust by integrating machine learning capabilities to it. Random forest uses the ensemble learning technique. By constructing several trees and taking an aggregate of them would only make the model robust as shown in Fig. 1. Apart from this, it also provides better classification since it uses boosting and bagging techniques. It also helps in avoiding overfitting compared to their counterparts. Random forest is better suited for this particular application because it can work on multidimensional input data and the data used to train the model has a dimension greater than 50. In an opportunistic network environment, each and every node contains the context and location information of its neighbors. This contextual information is used with the proposed model to make predictions. For a given node, extracted data about one neighbor at a time will be fed into trained model. Based on the features, the model classifies the neighbors as good forwarder or bad forwarder. After going through all the neighbors, data packets are forwarded to all of the good forwarders as explained in the Algorithm 2. Since it is an improvement over the Prophet algorithm, the Prophet probability is also considered one of the feature for classification. In order to see how the model performed in real world, the source code of the Prophet algorithm in one simulator was modified to include the trained model.

V. Experimental Evaluation

A. Simulation Settings

In this section, the performance of the proposed iProphet is evaluated using the Opportunistic Network Environment

Algorithm 1: Labelling the dataset

Result: Labelled dataset that can be fed into the model

```
Begin ;
foreach event e in set E do
    foreach message M in set M_s do
        /* if key of the records are
           same and the current_node and
           selected_forwarder_node are
           present successively in the
           path of the message and
           message_id in event e and
           message_id in message M are
           same                          */
        if message_id[M] == message_id[e] and
          Key(e) == Key(M) and current_node(e) +
          selected_forwarder(e) in path[M] then
            delivery_status[e] = "deliver";
            processed[e] = True;
        end
    end
    foreach event e in set E do
        if processed[e] != True then
            delivery_status[e] = "not_delivered";
            processed[e] = True
        end
    end
    foreach event e in set E do
        remove processed[e];
    end
end
```

(ONE) [13] simulator. The simulation is done on a terrain size of $4,500X3,400$ sq. m with 51 to 198 nodes deployed randomly. The transmission range of the node is set to 20 meters. The various parameters considered during simulation are shown in TABLE **??**. Here the proposed iProphet performance has been compared with Epidemic [5], Prophet [7] and Spray and Wait [14] routing protocols, under varying TTL values from 100 to 300 seconds, number of nodes from 51 to 198, and message creation interval from 25 to 35 seconds.

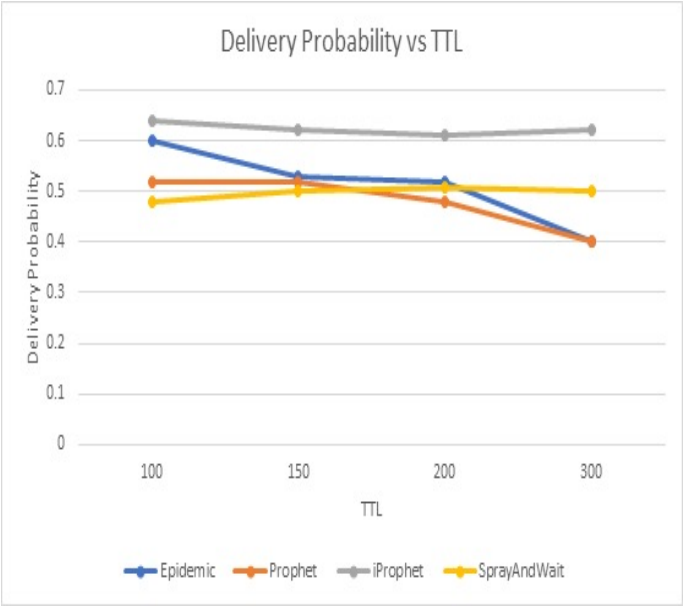

Fig. 2: Comparison of the delivery probability vs TTL.

B. Simulation results and performance analysis

In this section, Experimental evaluation of the iPRoPHET algorithm is provided by comparing it against Epidemic, PRoPHET, and SprayAndWait algorithms chosen as benchmarks, in terms of delivery probability, hop count, overhead ratio, average buffer time (in seconds), and average latency (in seconds), chosen as performance metrics, under varying number of Time-to-Live (TTL), number of nodes, and message generation interval respectively. Time-to-Live, number of nodes, message generation interval was varied between 100-300 seconds, 51-198 nodes, 25-35 to 65-75 secs respectively.

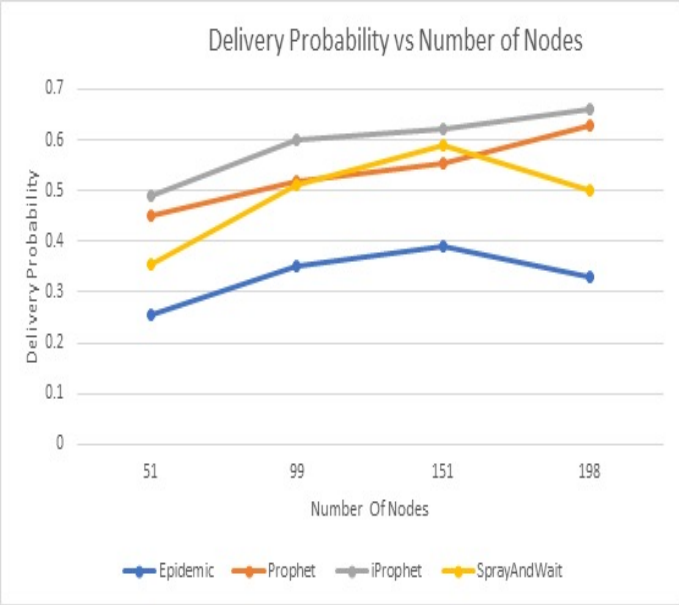

Fig. 3: Comparison of the delivery probability vs number of nodes.

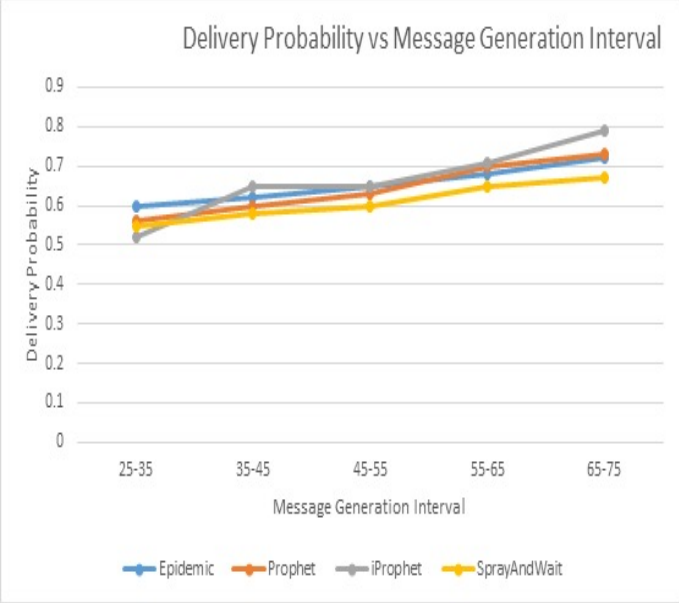

Fig. 4: Comparison of the delivery probablity vs message generation interval.

Fig 2, fig 3 and fig 4 compares the delivery probability of various algorithms when the TTL, Number of Nodes and Message Generation Interval is varied in the manner described above. It can be observed that delivery probability of algorithms other than Epidemic and iPRoPHET decreases with the increase in TTL. This is due to the fact the lifetime of messages increases with the increase in TTL and hence the messages remain in the node buffer for longer time than usual which results in more messages getting dropped. However, in iPRoPHET algorithm, the delivery probability increases because of the selection of the next forwarder using random forest classifier which considers PRoPHET probability and other contextual information of the forwarders. The next forwarders are selected not only when their PRoPHET probability is greater but also when the probability is lower which helps to clear the messages from the buffer as quickly as possible. When the number of nodes is increased, the algorithm gets to work with more forwarders enhancing the delivery accuracy of the messages. It can be observed that the iPRoPHET algorithm outperforms all other algorithms when it comes to delivery probability as the message generation interval is increased. When message interval is increased, the overhead of messages in the network reduces and the router queues are free to accommodate the messages which reduces message dropping from the nodes buffer queue. This decrease in overhead ratio is depicted in fig 8.x(Overhead ratio vs Message Generation Interval). This, accompanied with the fit forwarders helps messages to get delivered to their respective destinations with improved probability.

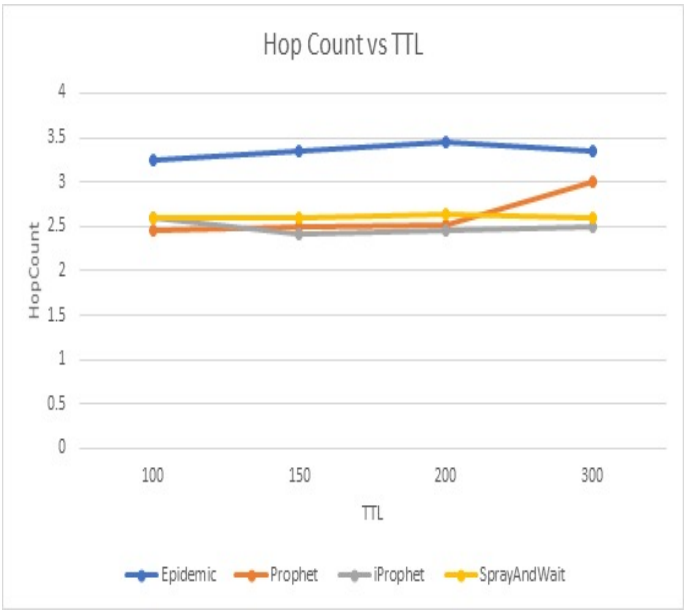

Fig. 5: Comparison of the hop count vs TTL.

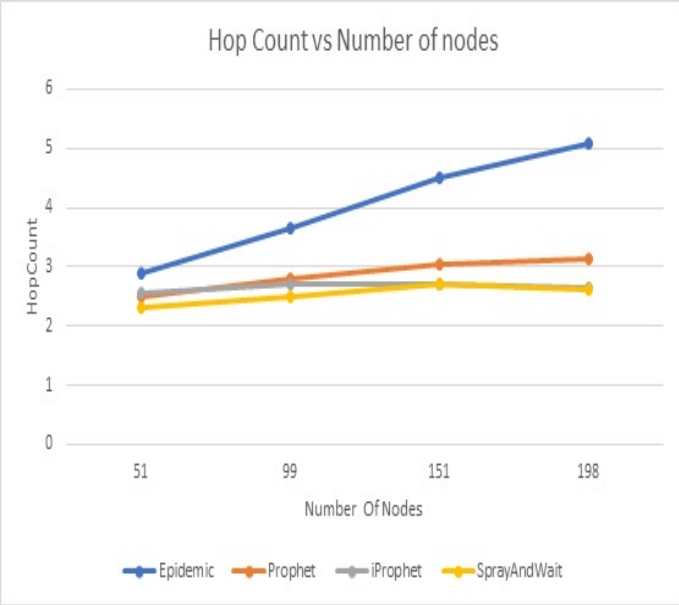

Fig. 6: Comparison of the hop count vs number of nodes.

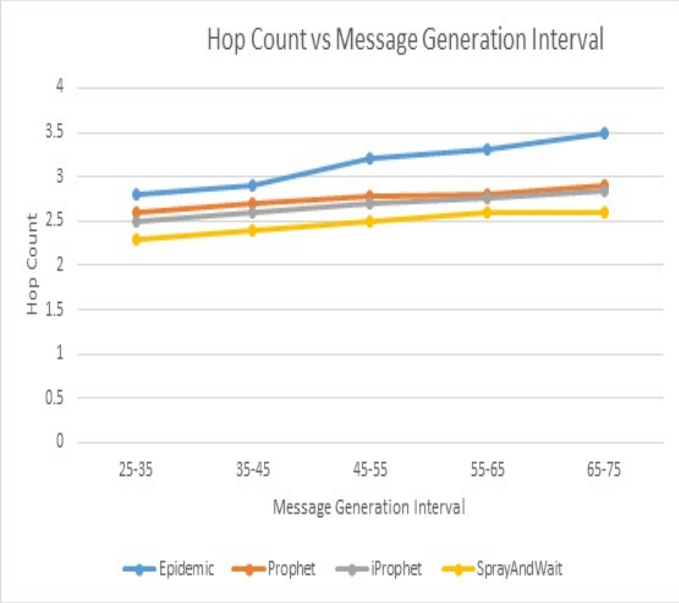

Fig. 7: Comparison of the hop count vs message generation interval.

Fig 5, fig 6 and fig 7 compares the Hop Count of various algorithms when the TTL, Number of Nodes and Message Generation Interval is varied in the manner described above. It can be seen that across all variations of TTL, number of nodes and message generation interval, iPRoPHET has a lesser hop count than PRoPHET algorithm. This is attributed to the fact that the iPRoPHET algorithm tries to maximize the delivery probability by even considering forwarders with lesser PRoPHET probability which could possibly have other better contextual attributes. When the number of nodes is increased, it leads to corresponding increase in number of forwarders available for routing the messages. Among these forwarders, the random forest classifier picks the fit nodes accurately leading to lesser overall hop count. Similar explanation holds when varied against TTL also. When TTL increases, hop count of other algorithms except SprayAndWait increases because of their nature of selection of forwarder nodes which are less likely to forward it to destination. Also, the messages stay in the buffer for longer period of time in PRoPHET since nodes won't forward the messages if the PRoPHET probability is lower for the forwarders. When the message generation interval is varied, the number of messages in the network decreases, but the precision of the classifier remains same. Therefore, the overall decrease in hop count is observed across all the graphs.

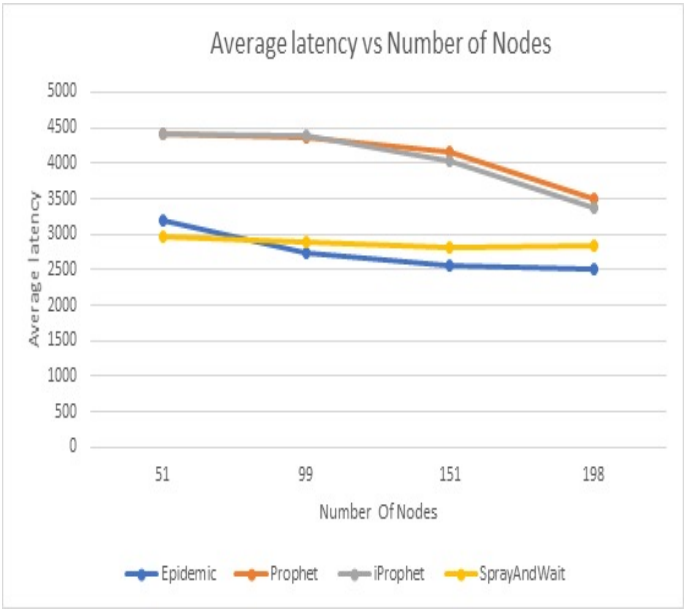

Fig. 8: Comparison of the average latency vs number of nodes.

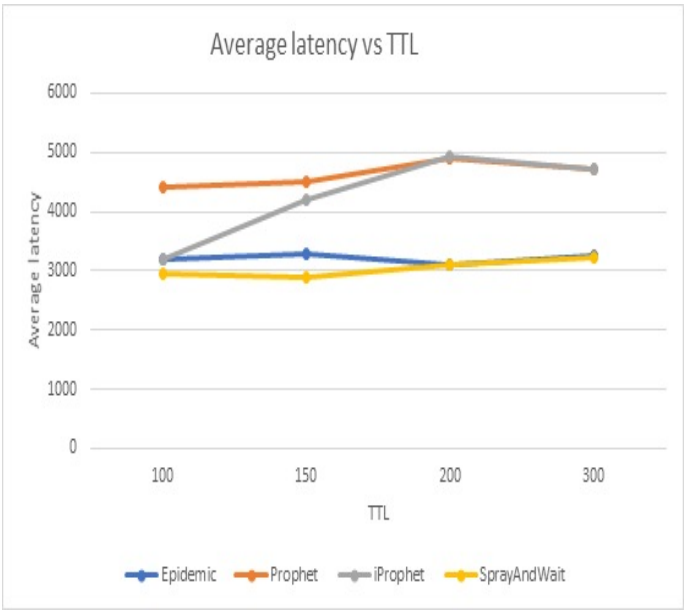

Fig. 9: Comparison of the average latency vs TTL.

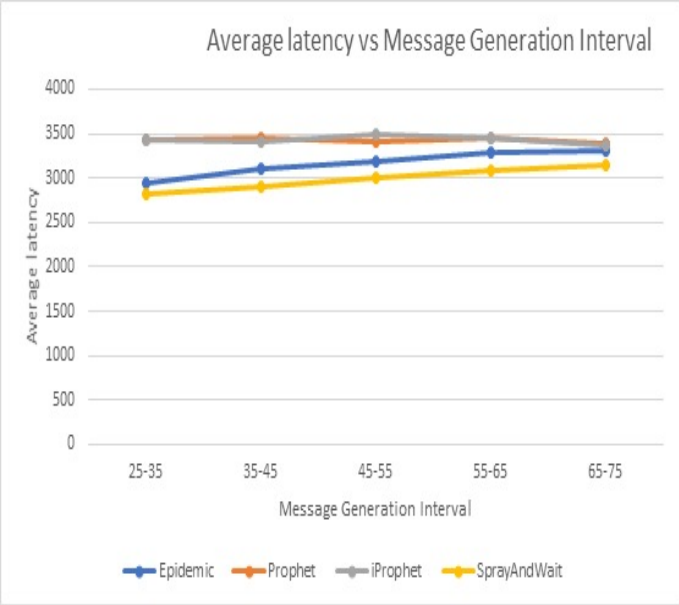

Fig. 10: Comparison of the Average latency vs message generation interval.

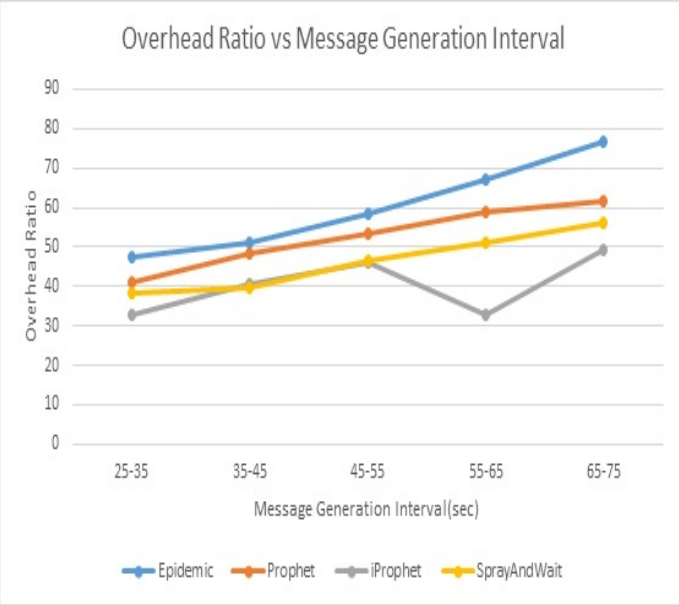

Fig. 11: Comparison of the overhead ratio vs message generation interval.

Fig 8, fig 9 and fig 10 compares the average latency of various algorithms when the TTL, Number of Nodes and Message Generation Interval is varied in the manner described above.

iPRoPHET algorithm has only slightly lesser average latency time than PRoPHET because of the selection of the nodes with lesser PRoPHET delivery probability which is required to maximize the overall delivery probability but in doing that, the algorithm may select few outliers that might keep the messages for far longer time than usual and as number of nodes, the outlier count increases which increases the message delivery latency. However, this is compensated by the lesser hop count delivery of the non-outlier nodes that help to keep the latency times in control for majority of the messages. Similar impact of outliers can be seen when the TTL and message generation interval is varied. But the majority of non-outlier nodes overcome such impact with their timely delivery of the messages to their destinations.

Fig. 7 compares the Hop Count of various algorithms when the message generation interval is varied from 25-35 to 65-75. When Prophet and iProphet are compared, the hop count is slightly lesser for iProphet algorithm. This is attributed to the fact that the iProphet algorithm tries to maximize the delivery probability by even considering forwarders with lesser Prophet probability which could possibly have other better contextual attributes. As a result, messages move through lesser number of hops than Prophet.

Fig. 11 compares the overhead ratio of various algorithms when the message generation interval is varied from 25-35 to 65-75. It is observed that the overhead ratio is better in iProphet because of the increased delivery probability resulting from better forwarders chosen as part of random forest classifier.

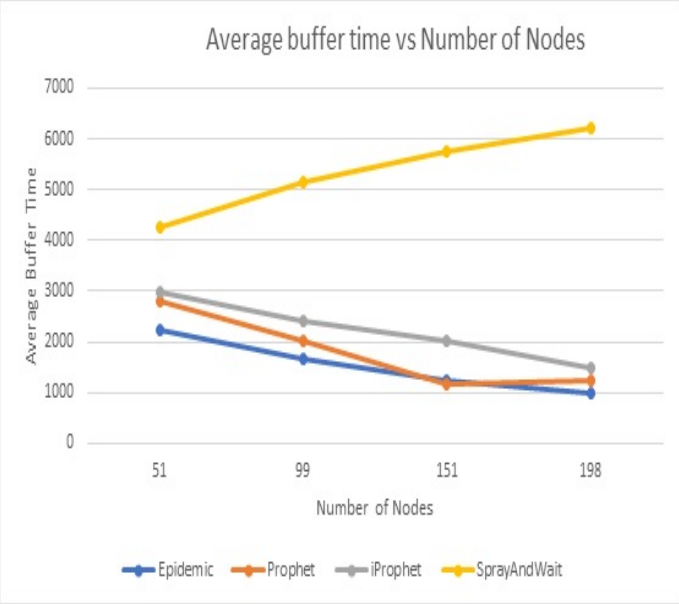

Fig. 12: Comparison of the average buffer time vs number of nodes.

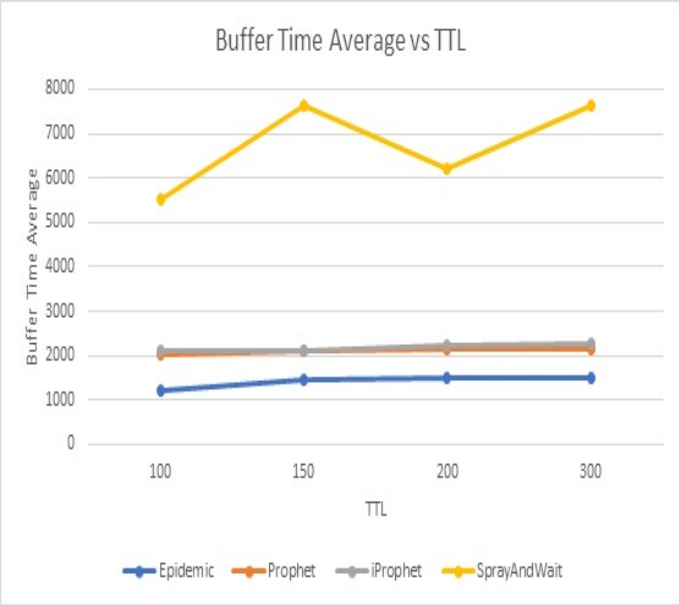

Fig. 13: Comparison of the buffer time average vs ttl.

Fig 12 and Fig 13 compares the average buffer time of various algorithms when the number of nodes and TTL is varied in the manner described above.. iPRoPHET algorithm has slightly greater average buffer time because of the selection of the nodes with lesser PRoPHET delivery probability which is required to maximize the overall delivery probability but in doing that, the algorithm may select few outliers that might keep the messages for far longer time than usual and as number of nodes increases, the outlier count increases keeping the average buffer time slightly larger and outlier prone. When the TTL is increased, the similar impact of outliers can be seen. The messages stay in the outlier nodes buffer for longer time than usual without getting forwarded and therefore add to the increased average buffer time.

VI. Conclusions and future work

In this paper multi-copy routing algorithm for opportunistic networks named iProphet or improved Prophet is proposed. This method makes use of machine learning classifier called random forest and delivery probability of existing Prophet to establish routing of messages. With the simulation results it can be concluded that: (i) While making each routing decision, random forest classified the nodes as reliable or unreliable which led to better selection of intermediate nodes for carrying out the routing task leading to efficient routing. (ii) Proposed method outperforms in terms of delivery probability, hop count, overhead ratio and latency in comparison with similar existing algorithms. (iii) The classifier trained with large amount of data extracted using simulation leads to precise classification of the nodes as reliable or unreliable for carrying out the routing task. However the proposed method will have poor performance in terms of average buffer time in par with similar multi-copy routing algorithm.

The average buffer time has experienced a performance issue due to the presence of the outlier nodes, which needs to be addressed as part of future enhancement. Furthermore, energy consumption of the nodes needs to be optimized for working with compute-intensive machine learning algorithms.

VII. Acknowledgment

This research work has been funded by the Science and Engineering Research Board (SERB-DST) Project File No: EEQ/2017/000681. Authors sincerely thank SERB-DST for intellectual generosity and research support provided.

References

[1] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Network*, pages 2787–2805, Oct. 2010.

[2] NN Srinidhi, SM Dilip Kumar, and KR Venugopal. Network optimizations in the internet of things: A review. *Engineering Science and Technology, an International Journal*, 2018.

[3] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine, vol. 44, Issue 11*, pages 134–141, Nov. 2006.

[4] B. Guo, D. Zhang, Z. Wang, and X. Zhou Z. Yu. Opportunistic iot: Exploring the harmonious interaction between human and the internet of things. *Journal of Network and Computer Applications*, pages 1531–1539, 2013.

[5] A.Vahdat and D.Becker. Epidemic routing for partially connected ad hocnetworks. *TechnicalReportCS-2000-06*, 2000.

[6] S. K. Dhurandher, Deepak Kr. Sharma, I. Woungang, and S. Bhati. Hbpr: History based prediction for routing in infrastructure-less opportunistic networks. *IEEE AINA 2013, Barcelona, Spain*, March 25-28, 2013.

[7] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE, Mobile Computing and Communications Review, vol. 7, Issue 3*, pages 19–20, July 2003.

[8] S. K. Dhurandher, S. J. Borah, M. S. Obaidat, D. K. Sharma, and B. Baruah S. Gupta. Probability-based controlled flooding in opportunistic networks. *Proc. of Intl. Conference on Wireless Information Networks and System (WINSYS), Colmar, France*, pages 3–8, July 20-22, 2015.

[9] C. Boldrini, M. Conti, I. Iacopini, and A. Passarella. Hibop: A history based routing protocol for opportunistic networks. *Proc. of IEEE Intl. Symposium on World of Wireless, Mobile and Multimedia Networks (WOWMOM 2007)*, pages 1–12, 2007.

[10] S. K. Dhurandher, S. Borah, I. Woungang, D. K. Sharma, K. Arora, and D. Agarwal. Edr: An encounter and distance based routing protocol for opportunistic networks. *Proc. of IEEE AINA 2016, Crans-Montana, Switzerland*, pages 297–302, March 23-25, 2016.

[11] I k Jeon and K w. Lee. A dynamic markov chain prediction model for delay-tolerant networks. *International Journal of Distributed Sensor Networks, Vol 12*, pages 2–7, Sept. 2016.

[12] D. K. Sharma, S. K. Dhurandher, I. Woungang, R. K. Srivastava, A. Mohananey, and Joel J.P.C. Rodrigues. A machine learning based protocol for effcient routing in opportunistic networks. *IEEESystemsJournal, DOI: 10.1109/JSYST.2016.2630923*, 2016.

[13] Ari Keranen. Opportunistic network environment simulator. *Special Assignment report, Helsinki University of Technology, Department of Communications and Networking*, 2008.

[14] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005.

Algorithm 2: Improved Prophet algorithm

```
Begin;
// Original Prophet
foreach node N_i as i do
    dp[i] <- random()
end
for every 5 seconds do
    foreach node N_i as i do
        dp[i] <- GetProphetProbabilty(i);
    end
end
// End of original Prophet
foreach message m in current node n buffer do
    foreach node N_i as n in neighborhood set G do
        P(n) <- dp[n];
        if dp[n] ≥ P(N_i) then
            if N_i == X then
                update X;
                update G;
                Label = RandomForest1(X, G, P(X), P(n));
            end
            if Label == "deliver" then
                Relay the message to node X;
            else
                if Label == "don't deliver" then
                    Don't relay the message to the node X;
                end
            end
        end
        if dp[n] < P(N_i) then
            if N_i == X then
                update X;
                update G;
                Label = RandomForest2(X, G, P(X), P(n));
            end
            if Label == "deliver" then
                Relay the message to node X;
            else
                if Label == "don't deliver" then
                    Don't relay the message to the node X;
                end
            end
        end
    end
end
```