International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 10, No.1 (Apr-2021)
http://dx.doi.org/10.12785/ijcds/100143

HMMR: Hidden Markov Model Prediction-Based Routing in Opportunistic IoT Scenario

Srinidhi N N[1], Sharath J[2] and Dilip Kumar S M[3]

[1,2,3]Dept. of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore, India

Received 18 Apr.2020, Revised 31 Jan. 2021, Accepted 27 Feb. 2021, Published 01 Apr. 2021

Abstract: Opportunistic networks are the networks having dynamic connections which are ad hoc in nature and are not fixed because of their dynamic movement. Predicting connection between two nodes is difficult in such scenarios and it depends on various parameters such as speed of the connection, number of nodes transmitting simultaneously, number of messages and time to live (TTL) of the message generated by a node for a given time interval. Opportunistic network are being used widely due to the development in IoT and 5G and messages generated has to be delivered effectively in such environment becomes very critical. So, in this paper, Hidden Markov Model Routing (HMMR) algorithm is proposed which uses Hidden Markov technique to find the most efficient route for the transmission of the message. The proposed algorithm uses emission probability and transition probability to predict the success rate of the route while predicting routing path. The proposed HMMR algorithm results have been compared with similar algorithms in order to evaluate its efficiency. The proposed algorithm can be used in health care application in order to monitor patients suffering from Cardio-Vascular disease.

Keywords: Cardio-Vascular Disease, Emission Probability, Hidden Markov, IoT, Opportunistic Networks, Routing, Transition Probability.

1. Introduction

IoT consists of interconnected devices wherein devices with processing capability interact with each other through communication medium which can be either electromagnetic waves or unfixed point-to-point connection [1]. An Opportunistic IoT is a class of IoT in which connection are not fixed and connections are established in an ad-hoc manner [2]. In this paper Hidden Markov Model Routing (HMMR) algorithm is proposed which uses Hidden Markov technique to recognize the most efficient route for the transmission of the message. Hidden Markov technique is a prediction technique to identify hidden states in finite set of transitions between states represented as a finite state machine[3]. Hidden Markov Model based routing algorithm which will be further referred as HMM routing algorithm uses a minimalistic Baum-Welch algorithm to predict the best possible next hop to transmit the packet for it to reach the end of the route. The algorithm in general terms determines the path of transmitted message that has taken till that point of time and together with the location of the node in the network. The algorithm performs the same steps for all the nodes it has encountered during transmission except for the destination node. The algorithm trains and builds the model for this region. For non-destination node it generates message only if the delivery probability for the route is high. The HMMR algorithm predicts the best route to deliver the packet by making using of a Hidden Markov Model to determine the observable hidden states [4]. The prediction using hidden states minimizes the parameters used by the algorithm and also reduces overall data required to predict the best possible route. This algorithm conserves energy by sending data to determined route rather than many different routes hence it also maximizes delivery probability. The overhead in processing large number of data is reduced through this algorithm hence network overhead is minimized.
HMMR algorithm utilizes less energy and network resources due to the following reasons:

- The algorithm is self-learning and predicts the best path to transmit the message.

E-mail:srinidhi.n@campusuvce.in, sharath.j@outlook.com, dilipkumarsm@gmail.com

- The algorithm uses minimalist version of HMM Baum Welch algorithm, which is efficient in prediction.
- Learning is performed only on the regions which the algorithm divides.

The HMMR algorithm works by training itself, the delivery of the message provides an opportunity for the algorithm to train itself so the algorithm can become more efficient. The algorithm efficiency is similar to base routing algorithm at the beginning of the message transmission, after the algorithm trains itself on number of successful deliveries of message the algorithm becomes effective and can transmit message with good delivery probability [5]. The HMM routing algorithm uses less parameters leading to less data utilization and as a result predicts path faster. In addition, the algorithm dividing the network into region limits the delivery probability storage requirement in the region, each node need to store data related to that region and storage of other region data is not required [6]. The HMM model is represented as model with finite set of states, so it performs transitions between states depending on the probability rule [7]. In such a finite state system upcoming transitions are fixed, and the bygone transition does not determine the next transitions. The present state and duration that has passed by solely determines the next transition [8]. This finite number of states of markov chain system is used in most of its applications and also it can be easily represented using a finite state machine.
Motivation of this work are, (1) Dynamic routing nature of opportunistic networks lead to difficulty in predicting next IoT node for routing of data. (2) Applications like healthcare require efficient delivery of critical data about patient in faster and efficient way. (3) Overhead in predicting next node as forwarder node during training phase of HMM.
Contributions of this work are:
1. Predicting reliable next node as forwarder node to forward data to destination.
2. Predicting efficient and reliable path for the delivery of the data without much delay.
3. Minimizing overhead in training and building of HMM to predict next forwarder node.
4. Providing test case for healthcare application particularly to monitor patient suffering from cardio-vascular disease.
The remaining portion of the paper is organized as follows. Section 2 discusses an overview of related work to identify major research work being done in this area. Then in section 3 problem statement of proposed work is presented. Later in section 4 model has been discussed and followed by proposed algorithm in Section 5. Section 6 discusses performance analysis of the proposed work and case study has been discussed in section 7.Finally, conclusion has been presented in section 8.

2. LITERATURE SURVEY

EERPFAnt proposed in [9], uses the theory of ant colony technique along with fuzzy logic to select the optimum relay by considering the energy level of the nodes and the routes that has successfully received replica of the message. Various parameters that are considered while calculating the distance are history of encounter, residual energy level, buffer memory message count, information from the successful routes to check whether it can take part in the transmission, rate at which packet is delivered, network resource usage, message delivery probability and number of copies generated. However, this process requires more memory to store all this information, which results in overhead in terms of memory usage. In [10], authors have proposed model to predict the route which the driver is traveling to. The proposed work performs well in terms of predicting the path which driver will likely to take in next time interval. But this method fails to work efficiently when vehicles are shared by the community such as those shared by a firm or by set of people. Also, this method has major setbacks when it has been used in IoT system where IoT system consists of group of users and accuracy of the model is about 50% for both trained model and data set.

In [11], a trust framework for authentication in opportunistic network is proposed. The trust framework authenticates other node using a trust vector, when a node comes into contact with super node then it transfer its trust vector to super node. The node then checks whether limit for maximum allowed number of semi super node has reached, if it has not reached this value then checks trust value of the node with the threshold value, if obtained trust value is higher than that of threshold value then this node is considered as semi super node. This framework allows registration of the node dynamically in an opportunistic network. Major drawback of this framework is that it requires excessive transmission of message to transmit trust vector which results in higher network overhead. Borah et al., in [12], have proposed infrastructure Opportunistic Internet of Things (OIoT) known as game theoretic approach for context-based routing (GT-ACR). GT-ACR is an optimized routing protocol to find next node in order to transfer the message. If the number of nodes is configured differently than the GT-ACR algorithm results in poor performance particularly when delivery probability is compared with TTL and with different message generation interval.

In [13], the algorithm replicates messages and transmits them depending on the network condition after finding the efficient intermediate node to transmit the message. But the recovery technique used in this method requires large amount of energy to receive and to delete requesting packets resulting in high energy consumption, which makes it not suited for energy constrained IoT applications.

Dhurandher et al., in [14], proposed novel algorithm for opportunist network called History-Based Prediction Routing (HBPR) to determine the next best node based on the behavioral information of the node. But in this method nodes will not acknowledge after receiving the message, which makes sender not be aware whether the node has received the message or not and also if parameters like nodes speed and TTL are varied then algorithm results in inconsistent performance.

In [15], authors have proposed a simple Hidden Markov Model represented as a finite state machine, the state in which the machine is represented by the status of the system. In this method IoT system heart beat is analyzed and transmitted to monitor its performance. Network anomaly can be identified very efficiently using this algorithm. Being very accurate while predicting failures at the same time the algorithm doesn't keep information for any process after processing it to reduce memory utilization but these data are required while making decision. Control replication scheme in [16], transmits message in multiple parallels and carried using control replication scheme. To predict the delivery probability, it considers the history of encounter information and contact duration. Construction of binary tree which has been carried as part of computation by this algorithm results in computational overhead. Buffer is used such that packet having higher delivery probability stay in the temporary memory and packet having lower delivery probability will not stay in the temporary memory, this results in packets with higher delivery probability staying in the temporary memory for prolonged duration and packet which take less time in the temporary memory will be ignored which will have average probability of getting delivered.

3. PROBLEM STATEMENT

The problem is to predict efficient routing path in order to conserve energy and to optimize latency, delivery probability, hop count, residual energy and overhead.

4. SYSTEM MODEL

In this work, a predictive routing algorithm based on the Baum-Welch estimation technique is proposed. This Beum-Welch is a Forward-Backward algorithm of HMM. The algorithm starts its processing by building a model which is then used to estimating the expected results for each parameter of the algorithm. During this process, it changes the value of the path which it has processed, thus the path with maximum value will be most visited. This proposed algorithm minimizes the number of parameters required while predicting the routing path and the number of parameters used are least compared to other algorithms. As a result, proposed algorithm is faster and uses less memory comparedto other algorithms.

Distance is calculated using coordinates of two nodes and it is given by following formula

$$d(a,b)^2 = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2} \quad (1)$$

First coordinate is represented by (a_1,b_1) and second node coordinate is represented by (a_2,b_2), hence these two coordinates are used to calculate the distance.

Various notations used in the algorithm are given below:

$Q = \{q_1,q_2,\ldots q_N\}$	Represents the states
$O = \{o_1,o_2,\ldots o_N\}$	Represents an observations sequence
$A = \{a_{11}\ldots a_{ij\ldots}a_{NN}\}$	A represents the transition probability given as the probability of when state changes from one state to next is represented as $a_{ij.}$
$B = b_i(o_t)$	It represents the likelihood of observation sequence representing probabilities, each expressing the probability of getting observation
$\pi = \{\pi_1,\pi_2,\ldots,\pi_N\}$	It is the emission probabilities. It represents the initial probability distribution over states, if the Markov chain begins in in the state i it is represented as p_i. $p_j = 0$ denotes state j is not its initial state.

As this algorithm uses HMM prediction method to predict delivery probability. HMM prediction problem can be classified into following classes.

1. Likelihood problem
2. Decoding problem
3. Learning-based problem

Given the set of observation and HMM parameters states predicting the successful route in terms of probability is called a likelihood problem. Given the HMM parameters and the observation sequence, finding the best-hidden state sequence is called decoding problem. This is the most widely used algorithm. Given the observation sequence and the set of states the algorithm tries learning the parameters of the HMM model is called the learning problem and this paper uses this problem-solving technique for predicting the most efficient route to deliver the message efficiently. Thus, this algorithm uses an HMM prediction model to predict route efficiently. The Baum-Welch algorithm uses both forward and backward probability which the algorithm follows while predicting next hop, which makes this method different from other algorithms. The probability of finding next hop is the resultant of both forward and backward probability and is given by

$$P\,(y_s = l, O|\mu) = P\,(o_1, o_2, \ldots. o_s, y_s = l|\mu)$$

$$P\,(o_{s+1}, os_{+2,}\ldots o_S|y_s = l, \mu) \quad (2)$$

Here, $y_s = l$ means s^{th} state in states sequence is the state l, μ is automaton, s is the time and P is the probability function and O is the set of observations.

The backward probability α can be stated as the mirror of forward probability, which is the second probability on the equation and defined as the observation probability from time $s+1$ to last time when observation is made at state l with time s.

$$\alpha_s(l) = P\ (o_{s+1}, o_{s+2},.....o_S|y_s = l,\mu).(3)$$

The transition matrix represented as T is row normalized which contains the probability by which the HMM model will change from one state to another. The matrix is represented as $|X|\times|X|$ having $|X|$ as its number of states, $T_{i,j}$ denotes state transition from i to j. The only subset of stochastic processes is represented due to history dependency of the model and due to this HMM represents other stochastic processes. The transition matrix T is given as follows

$$T = \begin{vmatrix} p_{1,1} & p_{1,2} & . & . & . & p_{1,n} \\ p_{2,1} & p_{2,2} & . & . & . & p_{2,n} \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ p_{n,1} & p_{n,2} & . & . & . & p_{n,n} \end{vmatrix} \quad (4)$$

In the transition matrix $p_{i,j}$ represents the probability of success for transition between state i to state j. Let r be the region, where all the divided regions consists of N nodes and $\{l_1, l_2...l_s\}, s = \{1,2,....,q\}$ represents separated regions of the network. n_k,i represents region r in the network k with iterations i, where $k = \{1,2,...,l\}$and $i = \{1,2,...,m\}$. $n_{1,i}$, is the representation of a node in the region 1 and $n_{2,i}$ is the representation of the node in the region 2. The processing overhead is distributed over the network by dividing nodes into regions and nodes in other regions not required to be processed thus resulting in less processing overhead.

5. PROPOSED HMMR ALGORITHM

HMMR algorithm is probability-based algorithm. This algorithm identifies frequent position pattern in the mobile connected set of nodes with their frequent pattern. The algorithm tries to identify the best set of intermediate nodes to forward the transmission so that the probability of getting delivered can be improved. This is based on the fact that future meet of the nodes is based on the previous and frequent meets. The parameters the algorithm considers are the priority of the message, resource availability and mean delivery ratio of current messages. The algorithm collects data during transmission this is the data sent by neighbors in the network. This is done to reduce the processing overhead that will be required to recompute the information by itself, thus it uses information processed by other nodes in order to reduce the overhead. This transmitted information can be used by other nodes; therefore, the node re-transmits the information with the message in an environment where nodes can connect to any node in random and the connections are not persistent. Fig. 1 provides overall flow of the algorithm, in this flow chart source node checks whether the packet is nearby its target node and if it has reached its destination then the algorithm trains itself based on this successful delivery. If the neighboring node does not able to deliver the packet to destination node then it checks whether the algorithm is in the most probable path to reach it by making use of Algorithm 1. The algorithm keeps improving based on successful routes and after a certain time interval the algorithm stops learning after finding frequent successful routes. To overcome from training model with the same routes to improve efficiency, threshold is introduced and represented as θ on the learning process. This threshold stops the algorithm from learning after successful learning attempts. This threshold function helps in increasing residual energy and processing power. The threshold θ depends on the area of the network as in implementation the area is fixed size, so threshold remains constant. The algorithm considers location of the node and buffer space as parameters. The algorithm keeps track of paths used to deliver the message to its destination, these paths are of the whole transition from the source to its destination. If the route has reached the destination then algorithm starts training the model with the corresponding list of location and buffer size pairs. The training using this algorithm generates the output in the form of matrix for the given inputs which are observation sequence represented as O, initial state probability represented is π, transition probability represented as a and output (emission) probability represented as b. Here O is the set of observation in the list of buffer sizes with their corresponding locations of the nodes in the network. This data is used to train the model on each successful route to the destination.

As the proposed work uses HMM Model to predict next routing path, there are three major techniques in HMM which includes Brute Force techniques, Viterbi technique and Baum-Welch technique. In this implementation Baum-Welch algorithm technique for HMM model implementation is used. Baum-Welch algorithm which is a forward-backward probability estimating technique can precisely predict the probability of reaching a destination and is therefore used in this implementation to find the

Algorithm 1: HMM Routing Algorithm

```
1  Input: Location, BufferSize, Count of Nodes
2  Output: Set of best nodes N to forward message
3  if no_of_attempts ≤ T then
4      if next_hop = destination then
5          location = getLocation ()
6          buffer_size = getBufferSize ()
7          HMMModel.train(location, buffer_size)
8            // calls the training algorithm
9      End
10 End
11 for ∀n ∈ path do
12     location = getLocation ()
13     buffer_size = getBufferSize ()
14     HMMModel.verify(location, buffer_size)
15     // verifies with a and b matrices
16     if result = "pass" then
17         N = n∪N // add node to set of N
18     End
19 End
```

most efficient route to reach the destination. The observation sequence length determined is assigned to *T* and *σ_size* are passed as an input parameter to the algorithm. Forward probability is calculated and represented in the matrix form using the variable forward_prob and backward probability with *backward_prob*. *π* is initialized by recursively calling γ function with forward-probability and backward-probability as its parameters in Algorithm 2. After these initialization steps the algorithm uses following formula recursively to calculate the transition probability.

$$a_{ij} = \frac{\sum_{t=1}^{T-1} p_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (5)$$

Similarly, the calculation of emission probabilities is carried by following formula.

$$b_{ij} = \frac{\sum_{t}^{O(t)=O_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (6)$$

When the function *P* is called the function does the calculation of the probability matrix. So, the probability matrix is given by

$$p_{ij} = \frac{forward\_prob_t(i) a_{ij} b_j(O(t+1)) backward\_prob_t(j)}{P(O|HMM\_Model)} \quad (7)$$

Verification function which is given in the algorithm considers emission probability and transition probability to predict the success rate of the route. The algorithm predicts the ongoing route through these matrices which are matrix of transition probability represented as *a* and matrix of emission probability represented as *b*. These outputs are generated as a part of its training process. The verification of probability for next hop is done using transition probability matrix that is *a*. The value of delivery probability of next hop in the matrix is at index corresponding to next-hop with current node as first index and next-hop as second index position, similarly emission probability must be verified using the emission probability matrix represented as *b* with the corresponding index of the next-hop like in the previous is the current node as first index and next-node as second index position. In the matrix the probabilities are stored in fractions.

6. PERFORMANCE ANALYSIS

This part of the proposed work discusses about the numerical outcome being accomplished after conducting the simulation for the formulated modules. The simulation of the proposed work has been carried out using Opportunistic Network Environment (ONE) [17] simulator. Simulation has been carried out in an area of 4500 x 3400 square meter with nodes varying from 66 to 186 arranged in a group of 4 to 10. Various parameters considered for simulation has been shown in TABLE 1.
The work simulates HMMR where, the performance validation has been considered with respect to set of performance metrics such as delivery probability, latency, average energy consumption, hop count and finally residual energy and overhead. The proposed work has been compared with the existing algorithmssuch as Spray and-Wait [18], Prophet [19] and EERPF [8].

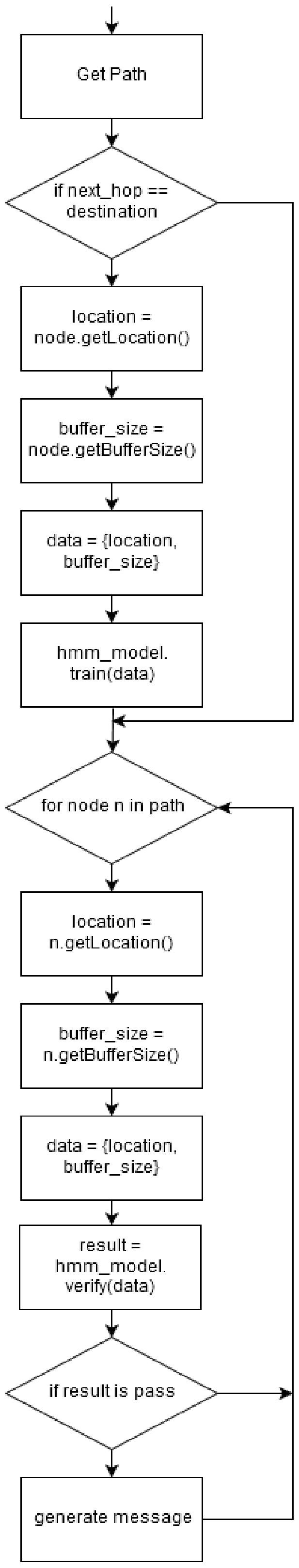
Get Path
if next_hop == destination
location = node.getLocation()
buffer_size = node.getBufferSize()
data = {location, buffer_size}
hmm_model. train(data)
for node n in path
location = n.getLocation()
buffer_size = n.getBufferSize()
data = {location, buffer_size}
result = hmm_model. verify(data)
if result is pass
generate message

Figure.1. Flow Chart of HMM routing.

Algorithm 2: Training HMM

```
Input: Observation sequence (O), σ_size
Output: Matrix of transition probabilities, Matrix of emission probabilities
T = O.getlength()
for s in 0 –> T do
    for k in 0 –> number_of_states do
        for j in 0 –> number_of_states do
            forward_prob[s+1] = forward_prob[s] * a[j][k]
        end
        forward_prob[s+1] = forward_prob [s+1] * b[j][O[s+1]]
    end
end
for s in T - 2 –> 0 do
    for k in 0 –> number_of_states do
        for j in 0 –> number_of_states do
            backward_prob[s] = backward_prob[s+1] * a[j][k] * b[j][O[s+1]]
        end
    end
end
for i in 0 –> number_of_states do
    π[i] = γ(i, 0, O, forward_prob, backward_prob)
end
for j in 0 –> number_of_states do
    for k in 0 –> number_of_states do
        for s in 0 –> T - 1 do
            number = number+p(s, j, k, O,forward_prob, backward_prob)
            denominator = denominator + γ (j, 0, O, forward_prob,backward_prob)
        end
        a[i][j] = number / denominator
    end
end
for j in 0 –> number_of_states do
    for l in 0 –> σ_size do
        for s in 0 –> T - 1 do
            g = γ (j, s, O, forward_prob, backward_prob)
            if k = O[s] then
                number = number + g
            end
            denominator = denominator + g
        end
        b[j][l] = number / denominator
    end
end
```

TABLE 1 Simulation Parameters

Parameter name	Assigned value
Simulation Area Considered	4500x3400 sqmt.
Groups of nodes	4 to 10
Total node count	66 to 186
Tram groups only	3
Total nodes for every tram group	2
Speed of the tram	6.5 kilometer per hour
Size of the buffer for the tram	50 megabytes
Pedestrian group only	3
Total nodes in each pedestrian group	30
Average pedestrian speed	0.5 to 1.5 kilometer per hour
Size of buffer for pedestrian	15 megabytes
High-Speed interface range of transmission	1500 meters
High-Speed interface transmission speed	10 megabytes
Bluetooth range of transmission	250 kilometers
Bluetooth transmission speed	20 meters
TTL for group messages	100 minutes
Transmitted message size	500 kilobytes to 1 megabyte
Message generation interval	25 to 35 seconds
Node movement	Shortest Path Map Based

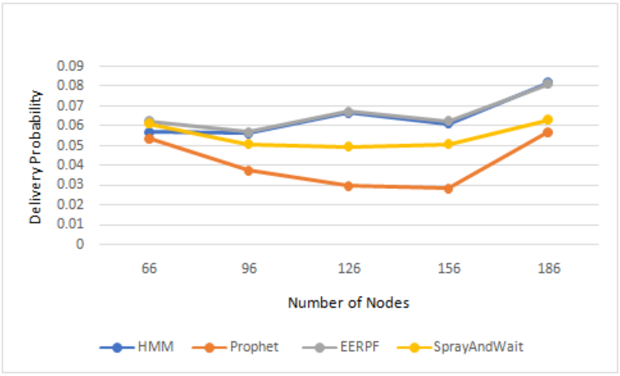

Figure.2. Number of nodes vs Delivery probability

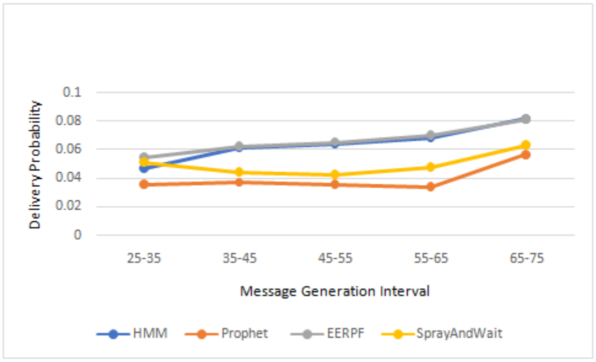

Figure.3. Message interval vs Delivery probability

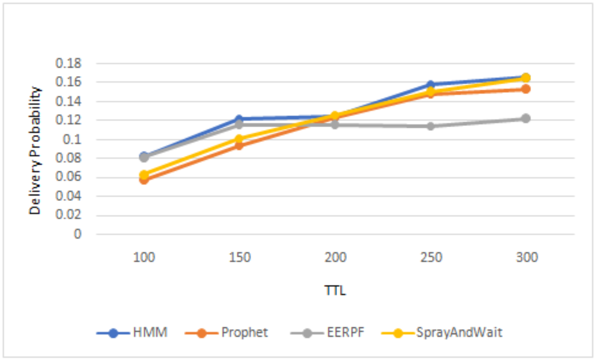

Figure. 4. Delivery probability vs TTL

Fig. 2-4 provides delivery probability of various algorithms such as Spray-and-Wait, Prophet, HMMR and EERPF when metrics such as Time to Live (TTL), Number of nodes and Message interval are varied and HMMR results better due to consideration of emission probability and transition probability to predict the success rate of the route. In Fig. 2 delivery probability v/s varying number of nodes is carried out. The HMMR algorithm has an overall delivery probability of 0.06468 which is highest among all the compared algorithms. It is immediately followed by EERPF which has the second-highest level of average delivery probability at 0.06122. The third highest delivery probability is from Spray and-Wait algorithm at 0.219525139 and least average delivery probability is Prophet at 0.0413. In Fig. 3 delivery probability of various algorithms with varying message generation interval is compared. The average delivery probability in case of HMMR algorithm is at 0.13052 which is highest among all the algorithms compared. It is followed by Spray-and-Wait algorithm with 0.12142. Then Prophet algorithm at 0.1154 and least average delivery probability is from EERPF at 0.10986. In Fig. 4 delivery probability of various algorithms with varying TTL is carried out. The average probability of delivery in case of the proposed routing algorithm is 0.06448 which is highest among all algorithms compared. It is followed by EERPF with 0.06282. The third highest delivery probability is from Spray-and-Wait algorithm at 0.04992 and least average delivery probability is from Prophet at 0.0399. Fig. 5-7 provides the hop count of various algorithms when TTL, Number of nodes and Message interval are varied and HMMR algorithm has better results due to effective prediction of the routes.

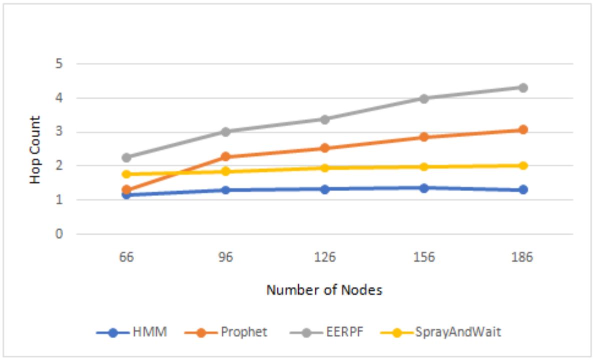

Figure.5. Number of nodes vs Hop count

In Fig. 5 hop count of various algorithms are carried out with varying number of nodes. The average hop count for HMMR algorithm is at 1.28962 which is lowest among various algorithms compared. It is immediately followed by Spray-and-Wait with 1.92098. Next lowest hop count is from Prophet algorithm at 2.41918 and highest average hop count is from EERPF at 3.40506. In Fig. 6 hop count v/s message generation interval are varied. The overall hop count for proposed algorithm is 1.28962. The Spray-and-Wait algorithm has the next-lowest level of average hop count at 1.92098. The third lowest hop count is from Prophet algorithm at 2.41918 and highest average hop count is from EERPF at 3.40506. In Fig. 7 hop count comparison of various algorithms with varying TTL of the message is shown. The overall hop count for the proposed algorithm is at 1.28962 which is lowest. The Spray-and-Wait algorithm has the next lowest level of average hop count at 1.92098. The third lowest hop count is from Prophet algorithm at 2.41918 and highest average hop count is from EERPF at 3.40506.

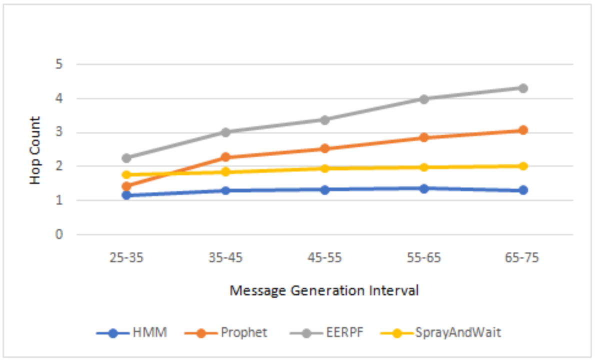

Figure. 6. Message interval vs Hop count

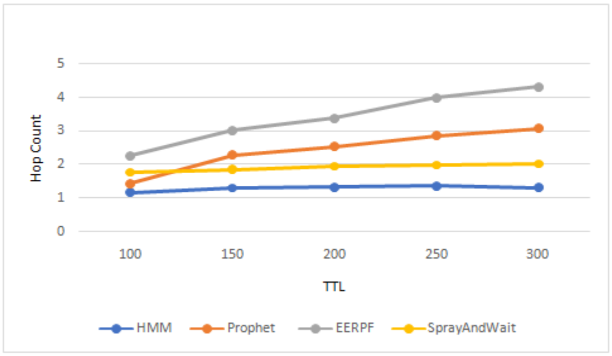

Figure. 7. Hop count vs TTL

In Fig. 8-10 provides the average latency of various algorithms when TTL, Number of nodes and Message interval are varied and HMMR algorithm has lesser latency due to efficient delivery of packets as resultant of less delay during route selection. In Fig. 8 average latency comparison against number of nodes in the network is simulated. The overall average latency for proposed algorithm is at 3415.00312 which is the lowest. The Spray-and-Wait algorithm has the next-lowest level of average latency at 3595.32848. The third-lowest average latency is from Prophet algorithm at 3795.90186 and highest average latency is of EERPF at 4048.9306. Fig. 9 provides average latency comparison of various algorithms when Message interval is varied. The overall average

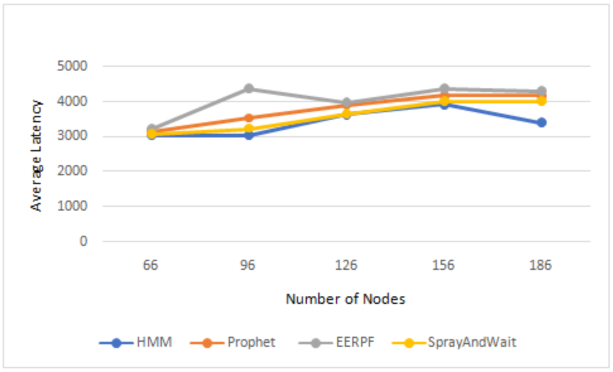

Figure. 8. Number of nodes vs Average latency

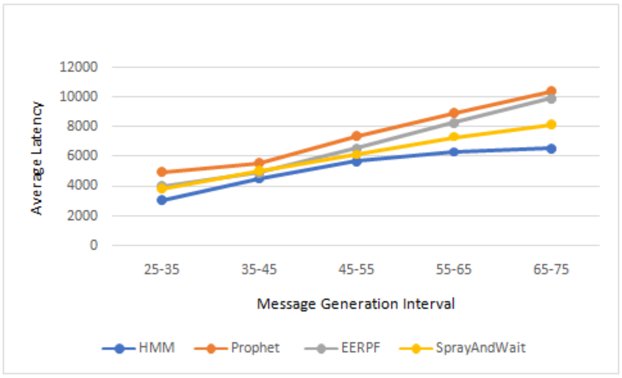

Figure. 9. Message interval vs Average latency

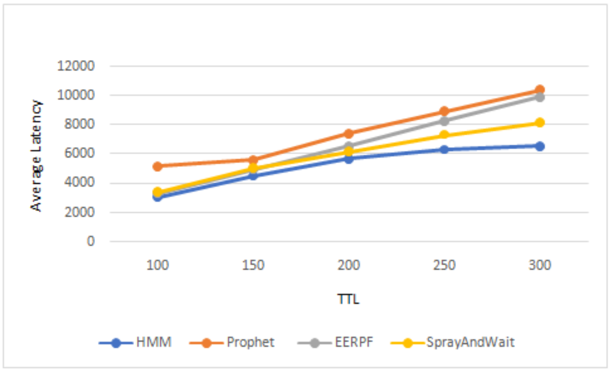

Figure. 10. Average latency vs TTL

latency for HMMR algorithm is at 5221.87908, Spray-and-Wait algorithm has 6034.63512, EERPF algorithm at 6599.52348 and highest average latency is from Prophet at 7464.1094. In Fig. 10 average latency comparison of various algorithms with varying TTL of shown. The average latency for HMMR algorithm is 5221.87908, then the Spray-and-Wait algorithm has the next-lowest level of average latency at 6123.95514 followed by EERPF algorithm at 6741.84864 and highest average latency is from Prophet at 7464.10994 average energy consumption of different algorithms are compared when parameters such as TTL, Number of nodes and Message interval are varied and HMMR algorithm gives better results due to the efficient prediction of path to route the message which reduces energy utilization by not forwarding message to path which is unlikely to deliver messages successfully. Energy consumption average for HMMR algorithm is at 0.770772039 after varying all the three different parameters, then followed by Prophet at 0.780474861, Spray and Wait at 0.779248666 and EERPF at 0.800959994.

In Fig. 11-13 average energy consumption of different algorithms are compared when parameters such as TTL, Number of nodes and Message interval are varied and HMMR algorithm gives better results due to the efficient prediction of path to route the message which reduces energy utilization by not forwarding message to path which is unlikely to deliver messages successfully. Energy consumption average for HMMR algorithm is at 0.770772039 after varying all the three different parameters, then followed by Prophet at 0.780474861, Spray and Wait at 0.779248666 and EERPF at 0.800959994.

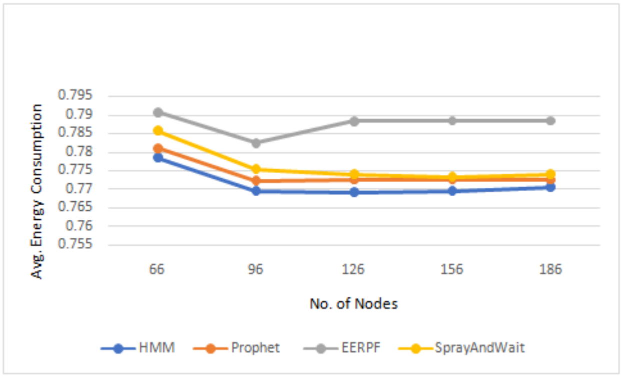

Figure. 11. Average energy consumption vs Number of nodes

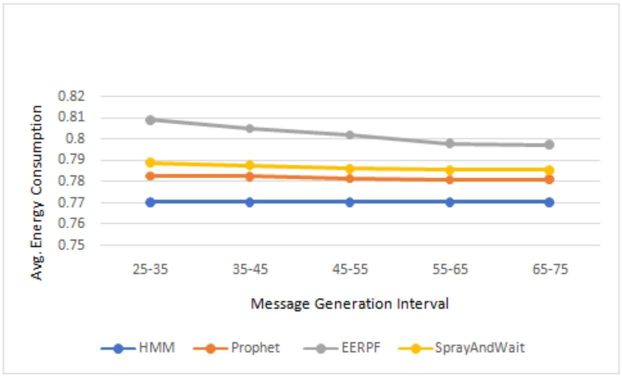

Figure. 12. Average energy consumption vs Message interval

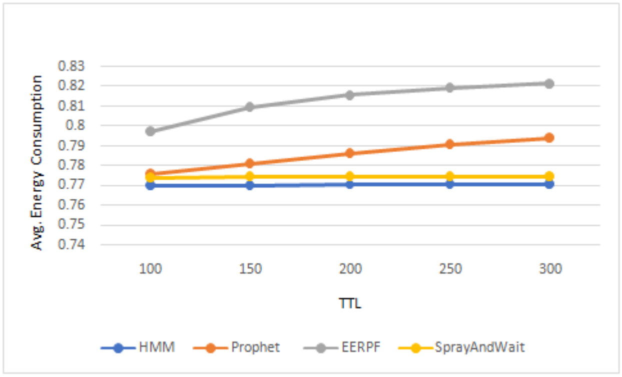

Figure. 13. Average energy consumption vs TTL

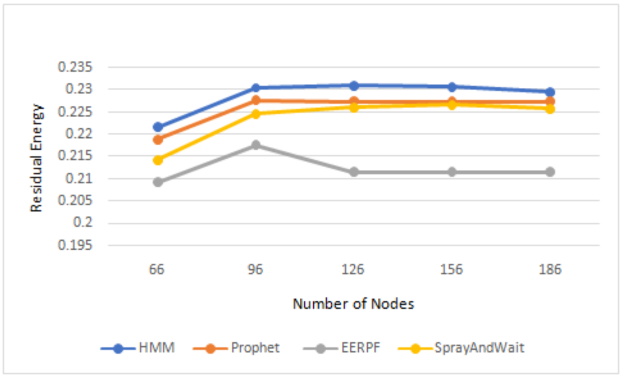

Figure. 14. Number of nodes vs Residual energy

Fig. 14-16 gives residual energy of different algorithms. The residual energy for HMMR algorithm is 0.229227961 which is followed by Prophet which has the next higher level of residual energy of 0.219525139, average residualenergyforSprayandWaitis0.220751334andaverageresidual energy for EERPF is 0.199040006.

Fig. 17-19 provides overhead of different algorithm when parameters like TTL, Number of nodes and Message interval are varied. The overall average overhead for HMMR is at 8.666806667, Prophet routing algorithm has an overall average overhead at 56.99136, significantly more compared to HMMR algorithm, likewise EERPF has an overall average overhead at 158.5108867 while overall average overhead for spray and wait is at 36.84770667, this reduction in overall overhead in case of HMMR can be attributed to the reduction in dropped packets results from unsuccessful routes.

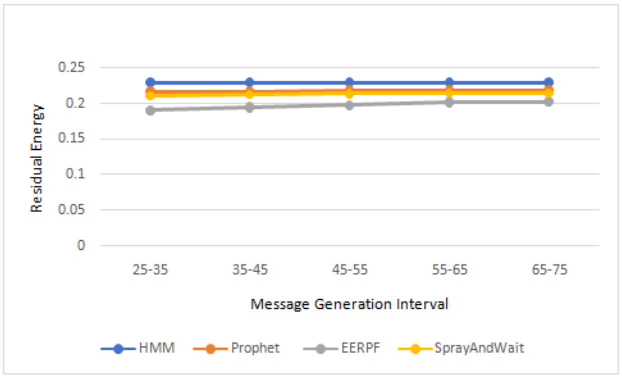

Figure. 15. Message Interval vs Residual energy

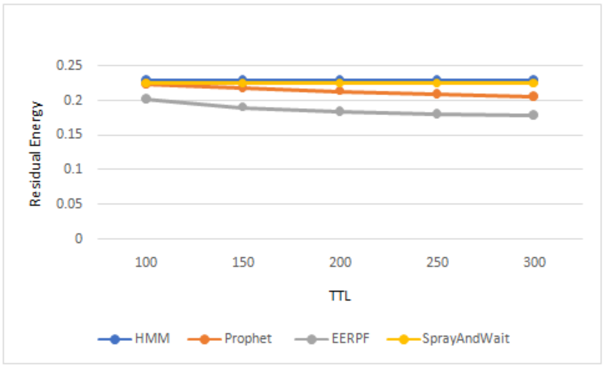

Figure. 16. Residual energy vs TTL

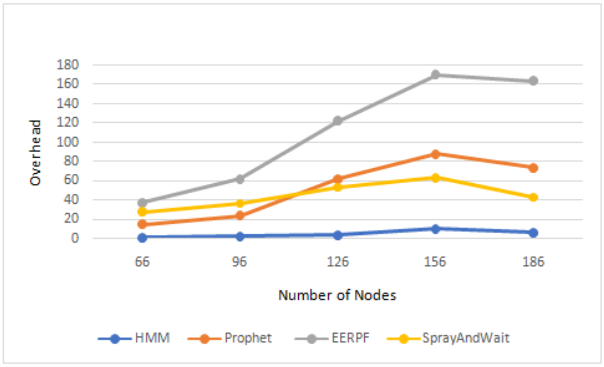

Figure. 17. Overhead vs Number of nodes

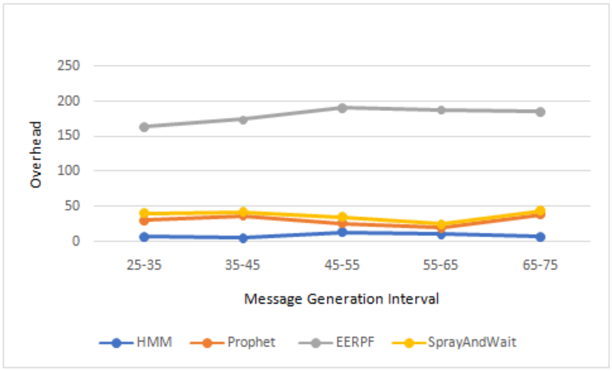

Figure. 18. Message interval vs Overhead

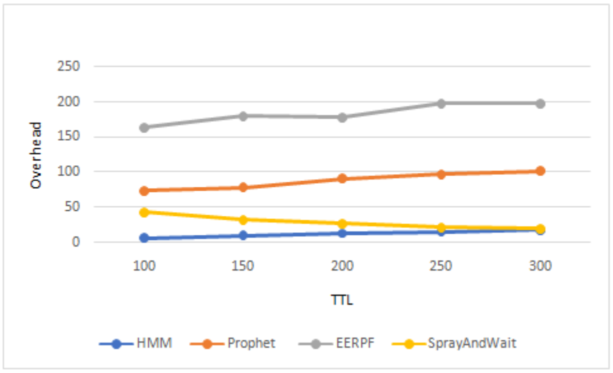

Figure. 19. Overhead vs TTL

7. CASESTUDY: HEALTH CARE APPLICATION

The proposed method to predict the routing path for successful delivery of messages will help in reducing delay while connecting to the next node to deliver data. In this case study proposed approach can be related to health care application using IoT especially to track as well as accurately deliver critical data of a patient suffering from Cardio-Vascular disease. Data about the patient's location will be continuously monitored to provide services in case of emergency. This work also helps to conserve energy by allowing idle sensors to be in sleep mode in order to conserve the energy. Only the sensors that take part in the routing and communication will be active. So, by that way energy consumption of sensor nodes can be reduced. Predicting next routing path of the node results in seamless connectivity which allows the mobile nodes to be connected to a sensor node continuously. This avoids data packet loss and losing critical data in case of emergency. This is a crucial aspect in health care applications as the loss of critical data can have a very significant downside. Seamless connectivity also ensures that switching between access point in an efficient way, decreasing delay and reducing signaling cost. Thus, seamless connectivity ensures the mobile node data is not lost and the data is routed in the most efficient way as possible. The collected data in the network will be sent to a gateway or a server. In case a sensor node relaying the data fails, there must be a solution to work around this. Self-healing, aspect of a network helps to work around the problem of node failure as the next best path is calculated and used in case of node failure. All the above-mentioned approaches to solving crucial issues in the IoT health care applications makes this work useful and applicable in real life IoT health care applications.

8. CONCLUSION

A Hidden Markov Model based predictive routing algorithm was proposed with an objective of improving routing efficiency. With the results it can be concluded that the HMMR algorithm reduces overhead and makes better utilization of network resources. Prediction of the next routing path for successful delivery of messages will help in reducing delay while connecting to the next node to deliver data. Considering emission probability and transition probability to predict the success rate of the route allows nodes to be connected to a sensor node continuously which result in efficient data delivery. Simulation results obtained by HMMR algorithm outperforms in terms of delivery probability, latency, average energy consumption, hop count, residual energy and overhead when compared to similar routing algorithms. The proposed work can be extended to non-opportunistic network scenarios and it can be further improved using real time data to prove its efficiency in real time scenario.

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] NN Srinidhi, Sunitha GP, Raghavendra S, SM Dilip Kumar, and Victor Chang. Hybrid energy-efficient and qos-aware algorithm for intelligent transportation system in internet of things. In International Journal of Grid and Utility Computing. Inderscience, (in press).

[2] Srinidhi, N. N., C. S. Sagar, S. Deepak Chethan, J. Shreyas, and SM Dilip Kumar. Machine Learning Based Efficient Multi-copy Routing for OppIoT Networks. In International Conference on Computational Intelligence, Security and Internet of Things, pp. 288-302. Springer, Singapore, 2019.

[3] Srinidhi, N. N., E. Nagarjun, and SM Dilip Kumar. "HMCRA: Hybrid Multi-Copy Routing Algorithm for Opportunistic IoT Network." In 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 370-375. IEEE, 2019.

[4] Jurafsky Daniel and James H Martin. Hidden markov models. Speech and Language Processing, pp. 464–479, 2019.

[5] Ullah, Israr, Rashid Ahmad, and DoHyeun Kim. A prediction mechanism of energy consumption in residential buildings using hidden markov model. Energies, vol. 11, no.2, pp. 358, 2018.

[6] Jung, J. Y., & Min, O. Spatial Region Estimation for Autonomous CoT Clustering Using Hidden Markov Model. ETRI Journal, vol. 40, no. 1, pp. 122-132, 2018.

[7] Kim, J., Jeon, Y., & Kim, H. The intelligent IoT common service platform architecture and service implementation. The Journal of Supercomputing, vol. 74, no. 9, pp. 4242-4260, 2018.

[8] Jiapeng Xu, Daniel WC Ho, Fangfei Li, Wen Yang, and Yang Tang. Event-triggered risk-sensitive state estimation for hidden markov models. IEEE Transactions on Automatic Control, 2019.

[9] Mohamed Ababou, Mostafa Bellafkih, et al. Energy efficient routing protocol for delay tolerant network based on fuzzy logic and ant colony. International Journal of Intelligent Systems and Applications, vol. 11, no. 1, pp. 69, 2018.

[10] Yassine Lassoued, Julien Monteil, Yingqi Gu, Giovanni Russo, Robert Shorten, and Martin Mevissen. A hidden markov model for route and destination prediction. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6. IEEE, 2017.

[11] Umesh Pal Singh and Naveen Chauhan. Authentication using trust framework in opportunistic networks. In 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–7. IEEE, 2017.

[12] Satya J Borah, Sanjay Kumar Dhurandher, Isaac Woungang, and Vinesh Kumar. A game theoretic context-based routing protocol for opportunistic networks in an iot scenario. Computer Networks, vol. 129, pp.572–584, 2017.

[13] Zhang Yu-Tong. Research of computer network data transmission routing method. In 2017 International Conference on Smart City and Systems Engineering (ICSCSE), pp. 167–171. IEEE, 2017.

[14] Sanjay K Dhurandher, Deepak Kumar Sharma, Isaac Woungang, andShruti Bhati. Hbpr: history based prediction for routing in infrastructureless opportunistic networks. In 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 931–936. IEEE, 2013.

[15] Palani Kumar and Meenakshi D'Souza. Design a power aware methodology in iot based on hidden markov model. In 2017 9th International Conference on Communication Systems and Networks (COMSNETS), pp. 580–581. IEEE, 2017.

[16] Chen Yu, Zhongqiu Tu, Dezhong Yao, Feng Lu, and Hai Jin. Probabilistic routing algorithm based on contact duration and message redundancy in delay tolerant network. International Journal of Communication Systems, vol. 29, no. 16, pp. 2416–2426, 2016.

[17] Ari Keranen, Jorg Ott, and Teemu Karkkainen. The one simulator for dtn protocol evaluation. In Proceedings of the 2nd international conference on simulation tools and techniques, pp. 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2009.

[18] P Maitreyi and M Sreenivas Rao. Design of binary spray and wait protocol for intermittently connected mobile networks. In 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–3. IEEE, 2017.

[19] Phearin Sok, Seryvuth Tan, and Keecheon Kim. Prophet routing protocol based on neighbor node distance using a community mobility model in delay tolerant networks. In 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, pp. 1233–1240. IEEE, 2013.

Srinidhi N N pursuing the Ph. D degree from University Visvesvaraya College of Engineering (UVCE) and received B. E degree and M. Tech degree in Vishweswaraiah Technological University and JNTU, Hyderabad respectively. He has published many papers in International Journals including Elsevier, Inderscience, other International reputed journals and Conferences. He has worked as reviewer for various reputed journals including Springer, IEEE IoT, Elsevier and many abroad research projects. He has given various talks on WSN, IoT and Robotics in various colleges including IIT, NIT and many premier institutions. His current research lies in the areas of sensor networks, cloud computing and IoT.

Sharath J received his B.Tech. in Computer Science and Engineering from Manipal Institute of Technology. Presently he is pursuing his M.E in Information Technology at University Visvesvaraya College of Engineering (UVCE), Bangalore University, Bangalore. His current research interests include Internet of things, Social network analysis, Machine learning, and Artificial Intelligence.

S M Dilip Kumar is a Professor in Department of CS&E, University Visvesvaraya College of Engineering. He has more than 21 years of teaching experience and published more than 60 papers in International Journals including Elsevier, Springer and Inderscience and Conferences. He was the Principal Investigator for a research project sponsored by Department of Science and Technology, Govt. of India in the area of grid computing and another research project sponsored by Department of Science and Technology in the area of Internet of Things.