An Energy-Efficient Location Prediction-based Forwarding Scheme for Opportunistic Networks

Satya J. Borah*, Sanjay K. Dhurandher*, Isaac Woungang†, Nisha Kandhoul* and Joel J. P. C. Rodrigues‡
* CAITFS, Division of Information Technology, NSIT, University of Delhi, Delhi, India
† Department of Computer Science, Ryerson University, Toronto, ON, Canada
‡ National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí, MG, Brazil;
Instituto de Telecomunicações, Portugal; University of Fortaleza (UNIFOR), Fortaleza-CE, Brazil
Email: satyaborah@yahoo.co.in; dhurandher@gmail.com; iwoungan@scs.ryerson.ca; nishakandhoul1990@gmail.com; joeljr@ieee.org

***Abstract*—Opportunistic networks (OppNets) is a subclass of delay-tolerant networks characterized by unstable topology, intermittent connectivity, and no guarantee of the existence of an end-to-end path between the source and destination nodes. In such networks, data forwarding from a source node to a destination node is a challenge. In this paper, an energy-aware routing protocol for OppNets (so-called Energy-efficient Location Prediction-based Forwarding for Routing using Markov Chain (ELPFR-MC)) is proposed, in which the next best hop selection of a message relies on the use of the node's residual energy and its location based on delivery probability. Simulation results show that ELPFR-MC is superior to E-Prophet, E-PRoWait, E-EDR and the Distance and Encounter based Energy-efficient Protocol for Opportunistic Networks (DEEP), where E-Prophet, E-PRoWait, E-EDR are respectively the energy-aware implemented versions of the Prophet, PRoWait, and Encounter and Distance-based Routing (EDR) protocols. The proposed ELPFR-MC outperforms DEEP in terms of node's residual energy by 6.34%, number of dead nodes by 6.58%, message delivery probability by 16.83% and average latency by 8% respectively when number of nodes are varied.** E-Prophet, E_PRoWait,E-EDR???

Index terms: Opportunistic networks, Delay-tolerant-networks, Opportunistic Network Environment (ONE) simulator, E-Prophet, E-PRoWait, E-EDR, DEEP.

I. Introduction

OppNets [1] are considered as an enhancement of mobile ad hoc networks, where the transmission of a data packet from source to destination is achieved in a store-carry-and-forward fashion by exploiting the opportunistic connectivity. Since such networks consist of hand held devices such as cell phones, laptops, etc, which operate based on their battery power, the successful delivery of a message is also dependent on the node's energy since most of the node's energy get drained during the message transmission and node discovery phases.

The recently proposed LPFR-MC protocol [2] considers the nodes present location and the angle formed by it and the corresponding source with respect to the destination to predict the nodes next location or region using a Markov chain and to determine the probability of a node moving towards the destination. The protocol does not consider the energy consumption of nodes causing the nodes to die earlier and thus reducing the network performance. Thus, considering the energy level of nodes as routing parameter, this paper proposes an energy-efficient version of LPFR-MC routing protocol where the next best hop is selected based on the node's residual energy and its location based delivery probability. Simulation results show that ELPFR-MC is superior to E-Prophet, E-PRoWait, E-EDR and the Distance and Encounter based Energy-efficient Protocol for Opportunistic Networks (DEEP), chosen as benchmark protocols, in terms of node's residual energy, number of dead nodes, message delivery probability, and message latency.

The rest of the paper is organized as follows. In Section II, some routing protocols for OppNets are described. In Section III, the proposed ELPFR-MC protocol is described. In Section IV, simulation results are presented. Section V concludes the paper.

II. Related Work

Many recent energy-aware routing protocols for OppNets have been proposed in the literature. In [3], Chilipirea et al. proposed an energy-aware extension of the Binary Spray-and-Wait protocol, where the node's remaining energy and vibrancy information are used for the selection of the next forwarder to carry the message to its destination. When a node meets another node, the vibrancy information, i.e. the node's activity during a given period of time, is exchanged among them and updated accordingly. This information is utilized in the routing decision to construct the routing path. In [4], Yao et al. designed an energy-aware routing protocol for sparse OppNets (called ERASA), which is based on the asynchronous sleep approach. In their scheme, those nodes that can participate in the routing process are prevented from entering the low power consumption dormancy state by becoming timely awaken when other nodes enter their communication range. In [5], Wennerstromy et al. proposed an energy-aware routing protocol for OppNets by investigating the heterogeneous link quality of connections. In their scheme, the link quality features are combined with the inter-contact opportunities to improve the node's energy saving and to reduce the number of relays of a message towards its destination. In [6], Khuram proposed an energy-efficient routing protocol for OppNets (called AEHBPR), in which the best forwarder of a message is selected based on a one-hop acknowledgment mechanism and the residual node's energy. In their scheme, a data packet is forwarded only through those relay nodes which have enough available remaining energy. In [7], Gao et al. proposed an energy-efficient routing protocol for OppNets in which the node's speed and residual energy are used as routing criteria to determine the best forwarder to carry a message to its destination. These criteria are used in a utility function to ensure that low efficiency in data accumulation and uncontrolled spraying can be prevented.

III. Designs of E-Prophet, E-PRoWait, E-EDR Protocols

The E-PRoWait, E-EDR, and E-Prophet protocols are respectively the energy-aware implemented versions of the PRoWait, EDR, and Prophet protocols. In the PRoWait protocol [8], initially, the packets are sprayed to the neighbouring nodes according to the Spray-and-Wait strategy [9]. Next, the selection of the best next hop to carry the message to destination is based on the forwarding strategy used in the Prophet [10] protocol. Each node is assigned a delivery predictability, and whenever a node meets with another node, the one that has the lower delivery predictability passes its packets to the one that has the higher delivery predictability of packets towards destination. This process continues until a node reaches the destination or has only one copy of the message, guaranteeing a successful delivery of the message to destination with a lesser delay and amount of flooding. In the EDR protocol [11], the selection of the best next forwarder of the message towards its destination is based on three parameters: (a) the number of times a node meets other nodes in the past (called encounter value), (b) the sum of encounters of a node, and (c) the Euclidean distance between every pair of nodes. Based on these values, a utility value (called γ) is calculated dynamically. The routing path is then formed by selecting the best forwarders of a message among a set of those nodes whose γ values are greater than or equal to a predefined threshold. In [12], the energy-efficient versions of the Prophet [10], PRoWait [8] and EDR [11] protocols are designed respectively, where the selection of the next best forwarder for a message relies on the energy of the nodes. For each of these schemes, *CurrentEnergy(node)*, the residual energy of each neighbouring node of the source/intermediate node is initially recorded; then, $FromEnergy_{node}$ the residual energy of the source/intermediate node and $ToEnergy_{node}$ the residual energy of the neighbouring node are calculated. Using these parameters, those set of nodes from the neighbours of the source for which $ToEnergy_{node}$ is greater than or equal to $FromEnergy_{node}$ are identified and stored in a Hashmap.

E-Prophet (resp. E-PRoWait) calculates the message delivery probability of all nodes available in the Hashmap as per the Prophet protocol [10] (resp. ProWait protocol [8]). Then, those nodes in the Hashmap whose message delivery probabilities are greater than that of the source/intermediate nodes are identified and stored in $Hashmap_{Proph}$ (resp. $Hashmap_{ProWait}$) [12]). Finally, the message is forwarded from the source/intermediate nodes to all nodes in $Hashmap_{Proph}$ (resp. $Hashmap_{ProWait}$). Similarly, E-EDR calculates the parameters λ, μ and η as described in the EDR protocol [11]. Then, it selects only those nodes from the Hashmap [12] whose η value is greater than or equal to a prescribed threshold *T* and store them in $Hashmap_{EDR}$ [12]. Finally, the message is forwarded from the source/intermediate nodes to all nodes in $Hashmap_{EDR}$.

IV. Proposed ELPFR-MC Protocol

The Proposed ELPFR-MC protocol considers an opportunistic network environment composed of *N* mobile nodes, which are cooperative and do not behave maliciously during the message forwarding process. It is also assumed that the nodes have sufficient buffer capacity to store their context information, and for each node, there are *n* neighbouring nodes which take part in the message transmission. In the ELPFR-MC scheme, the energy level of a node is considered in the routing decision. If the neighbour of a node has insufficient energy level to support the message transfer, the probability for that node to be a successful carrier of the message will decrease. The residual energy of the source node and each of its neighbor nodes is calculated as:

$$SourceEnergy_{node} = getCurrentEnergy(_{src/carrie,node}) \tag{1}$$

$$NeighbourEnergy_{node} = getCurrentEnergy(_{neighbour}) \tag{2}$$

where *getCurrentEnergy* returns the residual energy of the corresponding node. The energy (resp. probability) threshold parameters which are considered when making the routing decision, are obtained as:

$$EnergyThreshold(_{node}) = \Sigma_{n=1}^{n} NeighbourEnergy(_{node})/count \tag{3}$$

$$ProbThreshold(_{node}) = \Sigma_{n=1}^{n} Prob(_{node})/count \tag{4}$$

For the selection of the next best hop of a message, the following three additional parameters EP_{val}, EP_{param} and EP_{total} are also considered:

$$EP_{val} = (\alpha \times \frac{(getCurrentEnergy(_{neighbour})}{EnergyThreshold_{(node)}}) + (\beta \times \frac{Prob_{neighbour})}{ProbThreshold_{neighbour}}) \tag{5}$$

$$EP_{Toatal} = \Sigma_{n=1}^{n} EP_{val} \tag{6}$$

$$EP_{para} = \frac{EP_{total}}{count}, \tag{7}$$

where α and β are the weights assigned to *node's current residual energy* and *location based probability* respectively. These weights are varied to give priority to the energy metric or the location-based probability metric while performing routing. Using the aforementioned parameters, the ELPFR-MC scheme selects those nodes from the neighbour of the source/carrier node whose EP_{val} value is greater than or equal to EP_{para} and stores them in a Hashmap. Finally, ELPFR-MC transfers the message from the source/ carrier node to all the nodes in the constructed Hashmap. The pseudo-code of the proposed ELPFR-MC algorithm is shown in Algorithm 1.

V. Performance Evaluation

In this section, the performance of ELPFR-MC is evaluated and compared against that of E-Prophet, E-PRoWait, E-EDR and DEEP protocol, under varying number of nodes, TTL, and message generation interval, using the ONE simulator [13].

Algorithm 1 ELPFR-MC

```
1: Begin
2: Initialize the transition probability matrices for source ($TP_s$) and neighbour($TP_n$)
3: for every messages in the node buffer do
4:     for each n do
5:         Find the region and state of each n (i.e. $S_0$, $S_1$, $S_2$, $S_3$ and $S_4$)
6:         Calculate the probability that n moves from state 1 to another state (i.e. for the node in $S_0$ to move to another state with probability $P_{S00}$, $P_{S01}$, $P_{S02}$, $P_{S03}$ and $P_{S04}$)
7:         Update the transition probability matrix for n
           $TP_n$ = ($TP_{previous}$ + $P_{i,j}$) / 2
8:         Calculate the final transition probability matrix for n with the help of source/carrier node. $TP_{new}$ = ($TP_n$ × $TP_s$)
9:         Predict the next region/location of node n using a Markov chain $TP_{n,next}$ = [$nv_0$, $nv_1$, $nv_2$, $nv_3$, $nv_4$] × $TP_{new}$
10:    end for
11:    Predict the next region/location of the source/carrier node using a Markov chain $TP_{S,next}$ = [$Sv_0$, $Sv_1$, $Sv_2$, $Sv_3$, $Sv_4$] × $TP_{new}$
12:    for each n do
13:        Calculate the message delivery probability (DP) of the node n from the predicted values. i.e. $TP_{n,next}$ using the weighted matrix
           $Prob_n = TP_{n,next} \times [W_0, W_1, W_2, W_3, W_4]_{(5,1)}$
14:    end for
15:    Calculate the message delivery probability (DP) of the source/carrier node S from the predicted values. i.e. $TP_{S,next}$ using the weighted matrix
       $Prob_S = TP_{S,next} \times [W_0, W_1, W_2, W_3, W_4]_{(5,1)}$
16: end for
17: count = 0, $prob_{node}$ = 0
18: for each neighbor n of the source S do
19:    $prob_{node}$ = $prob_{node}$ + $prob_n$
20:    count = count + 1
21: end for
22: $ProbThreshold_(node)$ = $Prob_(node)$/count
23: $SourceEnergy_(node) = getCurrentEnergy_{(source/carrier,node)}$
24: $NeighbourEnergy_{node}$ = 0, count = 0
25: for each neighbor n of the source/carrier node do
26:    $NeighbourEnergy_{node}$ = $NeighbourEnergy_{node}$ + $getCurrentEnergy_{(neighbour)}$
27:    count = count + 1
28: end for
29: $EnergyTreshold_{node}$ = $NeighbourEnergy_{node}$ / count
30: $EP_{para}$ = 0, $EP_{total}$ = 0, count = 0
31: for each neighbor n of the source/carrier node do
32:    $EP_{val} = (\alpha \times (getCurrentEnergy_{neighbour}) / EnergyThreshold_{(node)} + (\beta \times Prob_{neighbour}) / ProbThreshold_{neighbour})$
33:    count = count +1
34: end for
35: $EP_{Total} = \Sigma_{n=1}^{n} EP_{val}$
36: $EP_{para}$ = $EP_{Total}$ / count
37: for each neighbour n of the source/carrier node do
38:    if $EP_{val} \geq EP_{para}$ then
39:        Insert node in Hashmap
40:    end if
41: end for
42: for each node in Hashmap do
43:    Transfer message from source/carrier node to node in Hashmap
44: end for
45: End
```

For simulation purpose, the considered energy parameters are: initial energy (ranging from 700 to 800 Joules), scan energy: 0.06 Joules, scan response energy: 0.08 Joules, transmit energy: 0.5 Joules, charging coefficient: 20 Joules, and Threshold on energy level: 100 Joules. In addition, the initial energy is randomly allocated to each node, the same initial battery level is allocated to each node at the start of the simulation; and once the battery of a node is exhausted at some point in time, that node is considered *dead* and it never gets recharged. The network dead time is defined as the time at which 80% of the nodes in the network are completely discharged. The *Shortest path map based movement model* is used as node movement model [13], under a simulation area of 4500 meters × 3400 meters. In our simulations, three groups of trams are considered, each with 2 nodes; the buffer size of each node is 15 Mb and each node's speed is in the range 2.7 m/s - 13.9 m/s. Also, three groups of pedestrians are considered, each with 30 nodes; the buffer size of each node is 15 Mb with each node's walking speed is in the range 0.5 m/s -1.5 m/s. The Bluetooth transmission rate is set to 250 Kbps, with a transmission range of 20 meters. The high speed interface transmission speed is set to 10 Mbps, with a coverage of 1500 meters. For messages generation purpose, a TTL of 100 minutes is assigned to each group of nodes. Each simulation runs for 15000 seconds, and a new message of size 500 Kb to 1 Mb is generated after every 25-35 seconds.

First, the number of nodes, TTL, and message generation interval are varied separately, and the impact of these variations on the node's residual energy is investigated. The results are captured in Fig. 1 to Fig. 3 respectively. In Fig. 1, it can

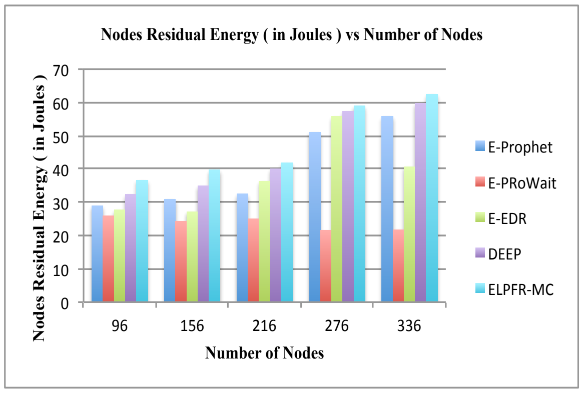

Fig. 1. Node's residual energy (in Joules) vs Number of nodes

be observed that for all studied protocols, the node's residual energy increases when the number of nodes is increased. This is attributed to the fact that in ELPFR-MC, the maximum number of nodes remain in the network, hence the node's residual energy is kept intact. The average value of the node's residual energy is 47.97 Joules for ELPFR-MC, 39.8779 Joules for E-Prophet, 23.7162 Joules for E-PRoWait 37.5888 Joules for E-EDR, and 44.9286 Joules for DEEP, showing the superiority of ELPFR-MC compared to other protocols. Indeed, in terms of node's residual energy, the performance of ELPFR-MC is 16.89% better than that of E-Prophet, 50.56% better than that of E-PRoWait, 21.64% better than that of E-EDR and 6.34% better than that of DEEP, when the number of nodes is varied.

In Fig. 2, it can be observed that when the TTL is increased,

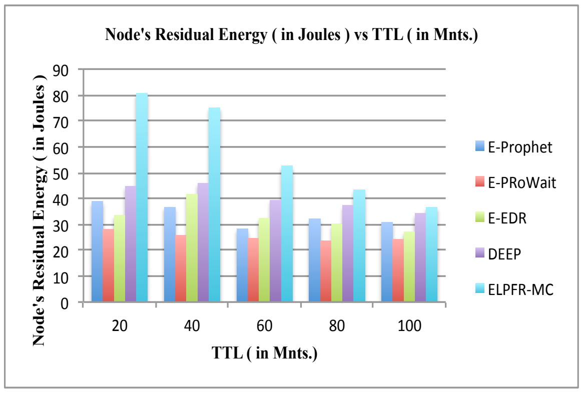

Fig. 2. Node's residual energy (in Joules) vs TTL (in minutes)

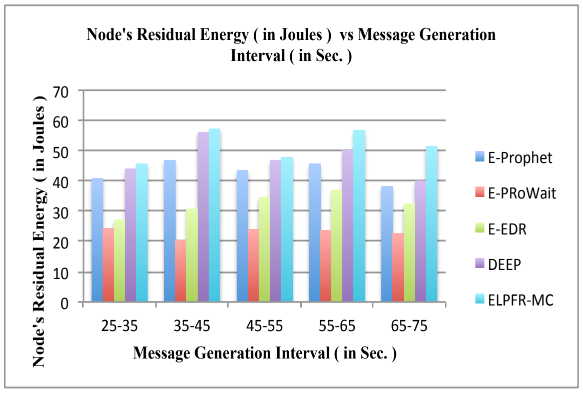

Fig. 3. Node's residual energy (in Joules) vs. Message generation interval (in seconds)

the node's residual energy decreases for all protocols. This is due to the fact that an increase in the TTL yields an increase in the number of message copies in the network, and this contributes in draining the node's residual energy. The average node's residual energy for ELPFR-MC is 57.7090 Joules, which is better than that of E-Prophet (33.4279 Joules), E-PRoWait (25,3285 Joules), E-EDR (32.9583 Joules) and DEEP (40.3219 Joules). Indeed, the performance of ELPFR-MC is 42.05% better than that of E-Prophet, 56.10% better than that of E-PRoWait, 42.88% better than that of E-EDR and 30.12% better than DEEP, when the TTL is varied.

In Fig. 3, it can be observed that when the message generation interval is increased, the average residual energy for ELPFR-MC is 51.7769 Joules, which is the higher than that of E-Prophet (35.4011 Joules), E-PRoWait (18.51022 Joules), E-EDR (25.9346 Joules), and DEEP (39.4132 Joules). Indeed, in terms of node's residual energy, the performance of ELPFR-MC is 31.62% better than that of E-Prophet, 64.25% better than that of E-PRoWait and 49.91% better than that of E-EDR, and 22.63% better than that of DEEP, when the message generation interval is varied.

Second, the number of nodes, TTL, and message generation interval are separately varied, and the impact of these variations on the number of dead nodes is investigated. The results are captured in Fig. 4, Fig. 5, and Fig. 6 respectively.

In Fig. 4, it can be observed that for all the studied protocols, when the number of nodes is increased, the number

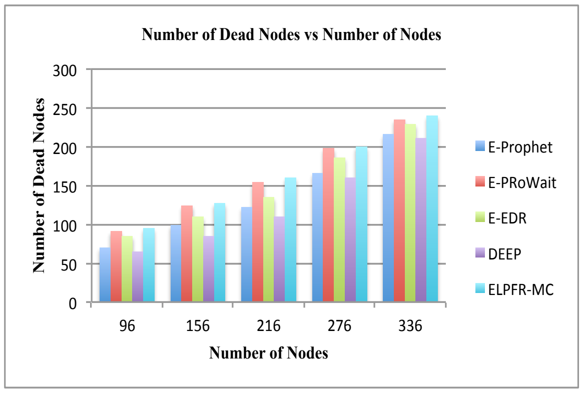

Fig. 4. Number of dead nodes vs. Number of nodes

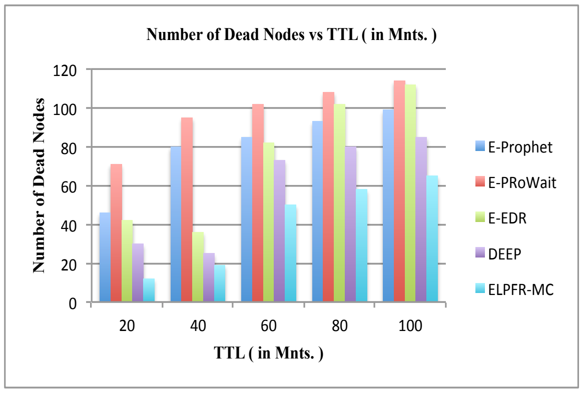

Fig. 5. Number of dead nodes vs. TTL (in minutes)

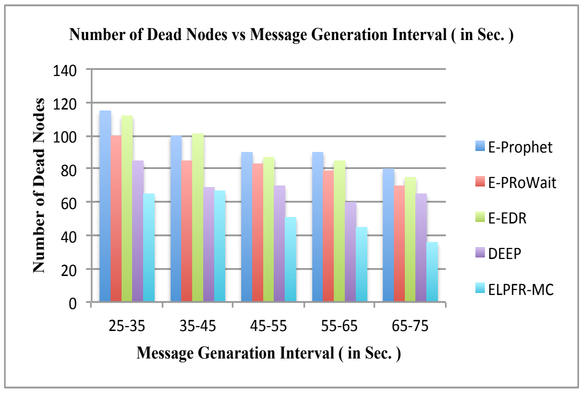

Fig. 6. Number of dead nodes vs. Message generation interval (in seconds)

of dead nodes also increases. The average number of dead nodes generated by ELPFR-MC is 118, which is lesser than that obtained using E-Prophet (134), E-PRoWait (160), E-EDR (149) and DEEP (126). Furthermore, the performance of ELPFR-MC in terms of number of dead nodes is 13.68% better than that of E-Prophet, 35.47% better than that of E-PRoWait, 25.84% better than that of E-EDR, and 6.58% better than that of DEEP, when the number of nodes is increased.

In Fig. 5, it can be observed that when the TTL is increased, the number of dead nodes also increases for all the protocols. Also, the average number of dead nodes generated by ELPFR-MC is 40, which is lesser than that generated by E-Prophet (80), E-PRoWait (98), E-EDR (74) and DEEP (58). Indeed, the performance of ELPFR-MC in terms of number of dead nodes is 49.37% better than that of E-Prophet, 58.36% better than that of E-PRoWait, 45.45% better than that of E-EDR, and 30.37% better than that of DEEP, when the TTL is increased.

In Fig. 6, it can be observed that when the message generation interval is increased, the average number of dead nodes for ELPFR-MC is 52, which is lesser than that generated by E-Prophet (95), E-PRoWait (83), E-EDR (92) and DEEP (69). Also, the performance of ELPFR-MC in terms of number of dead nodes is 44.42% better than that of E-Prophet, 36.69% better than that of E-PRoWait, 42.6% better than that of E-EDR, and 24.35% better than that of DEEP when the message generation interval is varied.

Third, the number of nodes, TTL, and message generation interval are varied respectively, and the impact of these variations on the message delivery probability is investigated. The results are captured in Fig. 7 to Fig. 9.

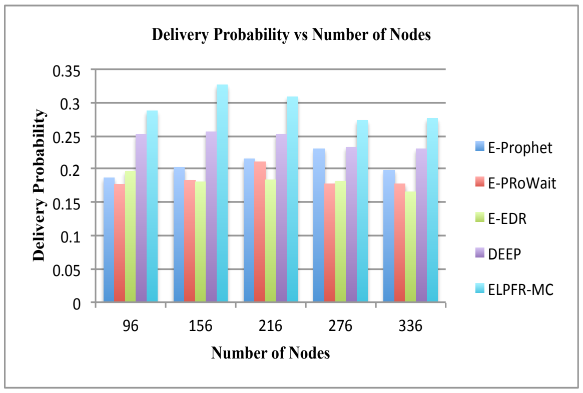

Fig. 7. Delivery probability vs. Number of nodes

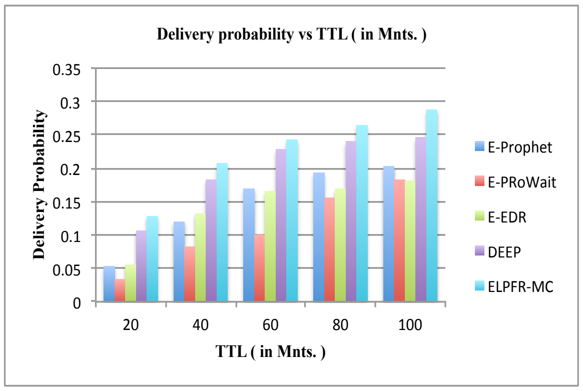

Fig. 8. Delivery probability vs. TTL (in minutes)

In Fig. 7, it can be observed that when the number of nodes is increased, the message delivery probability also increases for all the protocols. This is attributed to the increase in the number of message copies in the network, resulting to a large number of messages that get delivered to the destination. The average message delivery probability for ELPFR-MC is 0.2943, which is higher than that obtained for E-Prophet (0.2070), E-PRoWait (0.1855), E-EDR (0.1821) and DEEP (0.2448). Indeed, the performance of ELPFR-MC is 29.67% better than that of E-Prophet, 36.98% better than that of E-PRoWait and 38.13% better than that of E-EDR, and 16.83% better than that of DEEP when the number of nodes is increased.

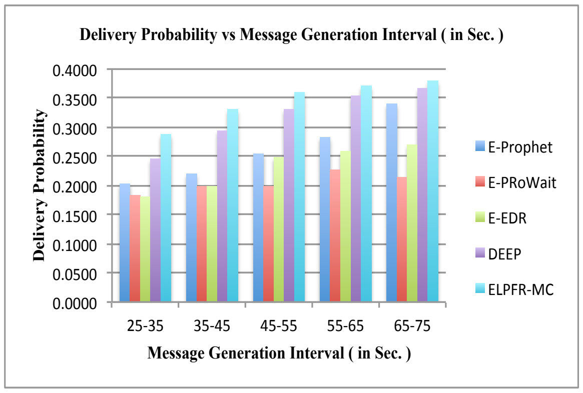

Fig. 9. Delivery probability vs. Message generation interval (in seconds)

In Fig. 8, it can be observed that when the TTL is increased, the message delivery probability for all the protocols also increases. The average message delivery probability is 0.2259 for ELPFR-MC, 0.1476 for E-Prophet, 0.1110 for E-PRoWait, 0.1405 for E-EDR and 0.2008 for DEEP. Indeed, the performance of ELPFR-MC is 34.661% better than that of E-Prophet, 50.86% better than that of E-PRoWait 37.79% better than that of E-EDR, and 11.13% better than that of DEEP, when the TTL is varied.

In Fig. 9, it can be observed that when the message generation interval is increased, the message delivery probability for all the studied protocols increases. Also, the performance of ELPFR-MC is 18.34% better than that of E-Prophet, 40.91% better than that of E-PRoWait, 33.09% better than that of E-EDR, and 7.85% better than DEEP,when message generation interval is varied.

Fourth, the number of nodes, TTL, and message generation interval are varied respectively, and the impact of these variations on the average latency is investigated. The results are captured in Fig. 10, to Fig. 12.

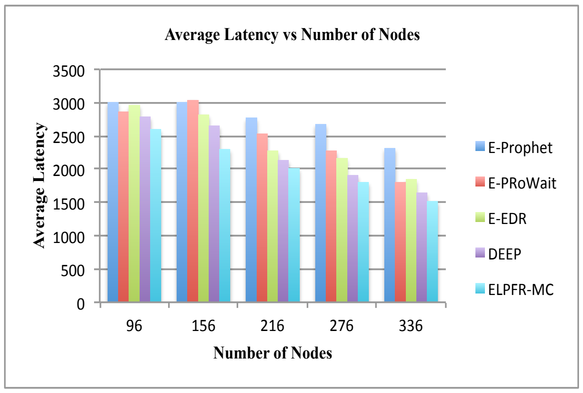

Fig. 10. Average latency vs. Number of nodes

In Fig. 10, it can be observed that when the number of nodes is increased, the mean average latency for ELPFR-MC is 2046.4562 seconds, which is lower than that of E-Prophet (2754.8783 seconds), E-PRoWait (2502.5159 seconds), E-EDR (2413.7280 seconds) and DEEP (2224.500 seconds). In-

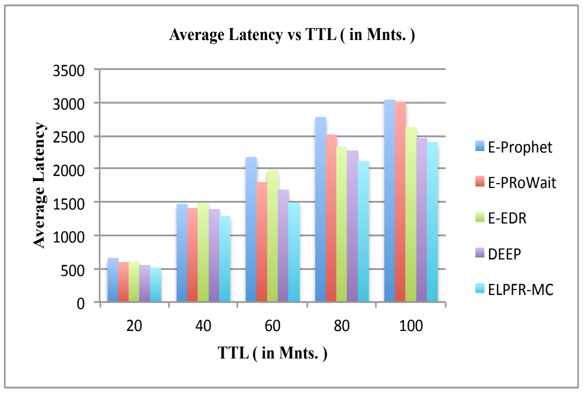

Fig. 11. Average latency vs. TTL (in minutes)

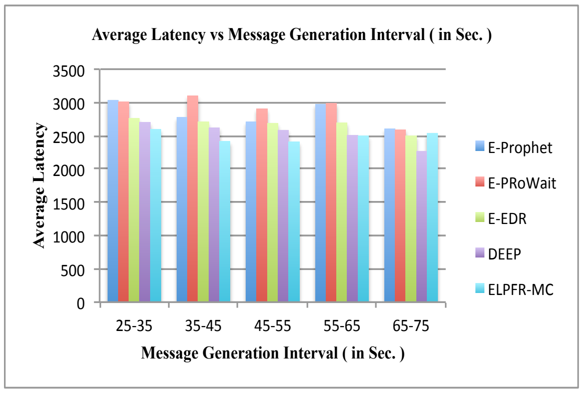

Fig. 12. Average latency vs. Message generation interval (in seconds)

deed, in terms of average latency, the performance of ELPFR-MC is 25.71% better than that of E-Prophet, 18.22% better than that of E-PRoWait 15.21% better than that of E-EDR, and 8% better than that of DEEP, when the number of nodes is increased.

In Fig. 11, it can be observed that when the TTL is increased, the obtained mean average latency for ELPFR-MC is lower than that obtained for the other protocols. Indeed, the performance of ELPFR-MC is 22.69% better than that of E-Prophet, 16.21% better than that of E-PRoWait, 13.38% better than that of E-EDR and 6.57% better than DEEP when the TTL is increased.

In Fig. 12, it can be observed that when the message generation interval is increased, the obtained mean average latency is 2491.5193 seconds for ELPFR-MC, 2820.65566 seconds for E-Prophet, 2918.8736 seconds for E-PRoWait, 2672.11292 seconds for E-EDR, and 2536.2408 seconds for DEEP. Indeed, in terms of average latency, the performance of ELPFR-MC is 11.66% better than that of E-Prophet, 14.64% better than that of E-PRoWait, 6.75% better than that of E-EDR, and 1.76% better than that of DEEP when the message generation interval is increased.

VI. Conclusion

In this paper, an energy-efficient routing protocol for OppNets (called ELPFR-MC) is proposed, which relies on the node's residual energy and location based delivery probability for message forwarding. Simulations results have shown that ELPFR-MC outperforms E-Prophet, E-PRoWait, E-EDR and DEEP in terms of node's residual energy, number of dead nodes, message delivery probability, and average latency. As future work, we plan to investigate the performance of ELPFR-MC using real-trace mobility models.

Acknowledgments

This work is partially supported by the NSERC DG grant, held by the 3rd author, under reference number: RGPIN-2017-04423, and a National Funding held by the 5th author, from the FCT- Fundação para a Ciência e a Tecnologia through the UID/EEA/50008/2013 Project, and by Finep, with resources from Funttel, Grant No. 01.14.0231.00, under the Centro de Referência em Radiocomunicações - CRR project of the Instituto Nacional de Telecomunicações (Inatel), Brazil.

References

[1] C.-M. Huang, K.-C. Lan, C.-Z Tsai, "A survey of opportunistic networks", Proc. of AINA Workshops, Okinawa, Japan, Mar. 25-28, 2008, pp. 1672-1677.

[2] S. K. Dhurandher, S. J. Borah, I. Woungang, A. Bansal, A. Gupta, "A Location Prediction-based Routing Scheme for Opportunistic Networks in an IoT Scenario", Proc. of Elsevier Journal of Parallel and Distributed Computing, 2017.

[3] C. Chilipirea, A-C Petre, C. Dobre, "Energy-Aware Social-based Routing in Opportunistic Networks", Proc. of WAINA Workshop, Barcelona, Spain, Mar. 25-28, 2013, pp. 791-796.

[4] Y. K. Yao, W. H. Liu, W. X. Zheng, Z. Ren, "An energy-saving routing algorithm for opportunistic networks based on asynchronous sleep approach", Applied Mechanics and Materials, Vol. 441, 2014, pp. 1001-1004.

[5] H. Wennerstromy, C. Rohnery, D.B. Smith, "Considering multi-contact encounters in opportunistic networks", Proc. of the 10th ACM MobiCom Workshop on Challenged Networks (CHANTS), Paris, France, Sept. 7-11, 2015, pp. 13-18.

[6] Khuram Khalid, "A History-Based Energy-Efficient Routing Protocol For Opportunistic Networks", Master of Science Thesis, Department of Computer Science, Ryerson University, Jan. 1, 2016.

[7] S. Gao, L. Zhang, H. Zhang, "Energy-aware spray and wait routing in mobile opportunistic sensor networks", Proc. of 3rd IEEE Intl. Conference on Broadband Network and Multimedia Technology (IC-BNMT), Beijing, China, Oct. 26-28, 2010, pp. 1058-1063.

[8] S. K. Dhurandher, S. J. Borah, M. S. Obaidat, D. K. Sharma, S. Gupta, B. Baruah, "Probability-based Controlled Flooding in Opportunistic Networks?, Proc. of Intl. Conference on Wireless Information Networks and System (WINSYS), Colmar, France, July 20-22, 2015, pp. 3-8.

[9] T. Spyropoulos, K. Psounis, C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks", Proc. of ACM SIGCOMM Workshop on Delay-tolerant Networking, Philadelphia, PA, USA, Aug. 26, 2005, pp. 252-259.

[10] A. Lindgren, A. Doria, D. Schelen, "Probabilistic Routing in Intermittently Connected Networks", Proc. of ACM SIGMOBILE Mobile Comp. Commun,2003, pp. 19-20.

[11] S. K. Dhurandher, S. J. Borah, D. K. Sharma, I. Woungang, K. Arora, D. Agarwal, "EDR: An Encounter and Distance based Routing Protocol for Opportunistic Networks", in Proc. of AINA 2016, Crans-Montana, Switzerland, March 23-25, 2016, pp. 297-302.

[12] S. K. Dhurandher, S. J. Borah, S. Tibarewala, I. Woungang, M. S. Obaidat, "Energy-Efficient Prophet-PRoWait-EDR Protocols for Opportunistic Networks", in Proc. of IEEE GLOBECOM 2017, Singapore, Dec. 4-8, 2017, pp. 1-6.

[13] Keränen A., Ott J., Kärkkäinen T., "The ONE simulator for DTN protocol evaluation", Proc. of 2nd Intl. Conference on Simulation Tools and Techniques (SIMUTools' 09), Rome, Italy, Mar. 2-6, 2009, pp. 1-9.