A Trajectory Inference-based Technique for Energy Efficient Store-and-Forward Technology

Antônio Rodrigo D. De Vit
UFSM
Fred.West., Brazil
rodrigodevit@inf.ufsm.br

César Marcon
PUCRS
Porto Alegre, Brazil
cesar.marcon@pucrs.br

Sidnei Renato Silveira
UFSM
Fred.West., Brazil
sidnei.silveira@ufsm.br

Ricardo Tombesi Macedo
UFSM
Fred.West., Brazil
rmacedo@inf.ufsm.br

Raul Ceretta Nunes
UFSM
Santa Maria, Brazil
ceretta@inf.ufsm.br

2021 Wireless Days (WD) | 978-1-6654-2559-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/WD52248.2021.9508277

***Abstract*— The wireless network infrastructure is critical for many applications, often operating over energy-sensitive networks with large communication delays. The store-and-forward technology used in Delay/Disruption Tolerant Networking (DTN) can minimize this problem. However, DTN requires efficient energy consumption technology for increasing the mobile node lifetime, especially when the connection opportunities among nodes change over time. This paper proposes an energy-saving technique on store-and-forward technology that saves energy by employing a node trajectory inference model based on machine learning for communication control. Experimental results indicate more than 47% of energy-saving on data communication applying the proposed trajectory inference model.**

Keywords— DTN; energy saving; trajectory inference model.

I. INTRODUCTION

Countless applications require communicating through wireless network infrastructures; applications like disaster response coordination [1] and vehicle monitoring systems for smart cities [2] often operate over sparse, low-quality, and energy-sensitive networks getting disconnections/disruptions and significant communication delays. Consequently, the network can compromise the application services if the messages are not properly delivered or the system spends a lot of energy working. The store-and-forward technology used in Delay/Disruption Tolerant Networking (DTN) [3] can minimize the communication problem by grouping and storing the application data on each node and forwarding it to the next node when access is available. This technology allows data transmission where there may be no end-to-end connection between source and destination. As a result, DTN, also known as opportunistic networking, can deal with unusual communication conditions, such as minutes or hours of transmission delays, intermittent connectivity, and low reliability [4], and can be explored on mobile ad hoc networks.

Although DTN solves message delivery requirements, energy management still is a challenge. The node movement in a DTN alters the network topology, constantly changing the connection opportunities among nodes [5]. Due to transmission range limits and intermittent connectivity transmissions, the network suffers long delays and energy losses. One fundamental problem in this context is routing messages from source to destination with a high delivery rate, low latency, low overhead, and low energy consumption since end-to-end paths might be absent for the entire message lifetime. Since the store-and-forward technology helps but does not avoid high energy consumption, for propagating messages over the network, nearby nodes must exchange information within their transmission range, which is limited by the hardware characteristics, reaching short distances. Each node consumes energy scanning neighborhood connections to receive and transmit data from/to the nearby nodes.

978-1-6654-2559-9 / 21 / $ 31,00 © 2021 IEEE

Dealing with nodes mobility is a promising way to increase the chance of knowing the DTN's nodes encounter moments, and this knowledge can be used to save the network energy. This work assumes nodes move over predictable trajectories and can infer its and neighborhood mobility and infer current and future trajectory movements. Thereby, the main problem to be solved is how to make each node judge the sleeping condition and the wake-up time according to the historical mobility of DTN's nodes. We propose a technique for saving energy on DTNs by using machine learning for inferring the trajectory of mobile nodes. The technique saves energy by controlling the activation and deactivation of the communication mechanisms.

This work is organized as follows. Section II presents the related work. Section III comprises the proposed approach with a detailed description of the models used for node trajectory inference. Section IV details the experimental setup and obtained results. Finally, the main conclusions are clarified in Section V.

II. RELATED WORK

Borah et al. [6] suggest an energy-aware routing protocol for OppNets; the next best hop selection of a message relies on the node residual energy and its location based on delivery probability. Chunyue et al. [7] propose the EASE routing algorithm, combined with an asynchronous sleeping mechanism, making each node judge the sleeping condition and the wake-up time according to the historical encounter information and the node movement status. De Vit et al. [8] proposed a method of classifying trajectories that, together with the fixed schedules of bus tables, seeks to determine the times of meetings between mobile nodes. Our work differs from these works demonstrating that energy-saving approaches applied to DTN regarding (i) the protocol used to route messages through the network nodes and (ii) the knowledge of the position of nodes for efficient message delivery. This section shows how our work relates to other scientific approaches considering these two aspects.

A. Message Routing Protocols

The routing protocols used in DTNs are mainly based on the mobility of the nodes aware and store-carry-forward mechanisms. The Epidemic protocol operates similarly to the

spread of an infectious disease. Nodes that adopt this protocol disregard the content of the message they carry, transmitting the message blindly to all other nodes that do not yet have a copy of this message until the message reaches its destination. Spray and Wait is a two-phase epidemic protocol with a limited number of copies of each message sent. The authors of the PRoPHET protocol propose a probabilistic routing protocol for DTN networks, improving routing performance by doing probabilistic routing [9]. Derakhshanfard and Soltani [10] present a bitmap-based approach for designing a tree to send packets without any additional and pointless routes upon receiving a request to send a message to a specified node; because each packet has its unique and not time-consuming path. Wu et al. [11] introduced a way of building social circles based on the node clustering phenomena in DTN. Considering that the forwarding capability of nodes is significantly different, they proposed a spray strategy based on social circles to improve the spray-and-wait routing algorithm.

Since people or vehicles can transport a node, it is crucial to managing the energy costs for sending/receiving messages and probe connections from other nodes within its transmission range. Unlike the related research presented here, our contribution is to demonstrate that exploring mobility is a promising way to save energy in a DTN. The probing connections close to the nodes within their transmission range are major factors of energy consumption among nodes [12]. A node that can predict its mobility and neighbors can infer current and future trajectories, enabling to explore techniques to reduce energy consumption. Besides, different routing algorithms can implement our proposal without changing its operating characteristics.

B. Node Trajectory Inference

Several criteria must be considered to understand the principles for inference of node trajectory [13]. The Manhattan mobility, Random Waypoint, and Random Walk models introduce a "memoryless" pattern with no dependence between current and future movements; thus, they are not suitable for modeling DTNs mobility problems. In literature, we found different traces based on the mobility of humans, animals, and vehicles to model real scenarios. Our work uses traces from the real-world bus scenario [14].

III. Proposed Approach

For saving energy on the store-and-forward mechanism of DTNs, we propose a technique that blends node trajectory prediction and the node meeting knowledge. With the meeting knowledge, the solution controls the activation and deactivation of communication mechanisms avoiding unsuccessful scanning phases. The prediction method uses a machine learning technique to build decision trees capable of classifying node trajectories and predicting node positions for a meeting map, referred to here as *ConnectivityMap*.

The machine learning process requires high computational power, which is performed by a central computer not over network nodes. Assuming we are dealing with a fleet of buses, each vehicle, at the beginning of its daily work, leaves the garage with the first version of the *ConnectivityMap*. This map is updated throughout the day as the traffic conditions escape from the previously learned behavior - an accident on a high-traffic vehicle that interrupts traffic. Also, at each period, when passing through a specific station, a node can receive a new and updated *ConnectivityMap*, keeping the map updated.

To explain how our approach works, the following sections detail the energy-saving and inference models.

A. Energy Saving Model

A node in a DTN operates at least three states for saving energy during its lifetime. Fig. 1 shows a Finite State Machine (FSM) for a DTN node operation regarding communication and energy management. From the initial **Scanning** state, two other states can be reached. If the node found another one in its range of connectivity, it goes to the **Communicating** state, remaining there while having data to transmit and/or receive. When the communication finishes, the node goes to the **Sleeping** state, a low energy consumption state, until reaching a timeout defined by the *sleep time*. When the node wakes up, it goes back to the Scanning state and restarts scanning for neighboring nodes during a predefined period defined as *scan time*. If the scanning procedure fails during the *scan time*, the node goes to sleep again to reduce its energy consumption.

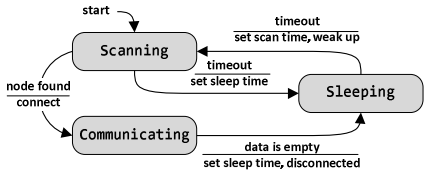

Fig. 1. FSM of energy consumption for a node in a DTN.

The FSM shows the energy management strongly depends on *sleep time* and *scan time* settings generating a tradeoff. With a high *sleep time,* the sleeping node may lose several opportunities to find other nodes. Nevertheless, with a short *sleep time*, the node remains much time spending an enormous amount of energy scanning for possible communications, which is a typical characteristic of node behavior in a DTN; i.e., a node spent more time disconnected and looking for connections than connected and transferring information. The Sleeping state must be avoided to guarantee connections in all meeting opportunities, resulting in substantial energy consumption. The challenge is to set the *sleep time* and the *scan time* in a way the node wakes up in time to meet other nodes, maximizing the communication opportunities. Our approach explores the node trajectory inference to maximize communication opportunities saving energy, represented by the **Inferring** state on the FSM of Fig. 2.

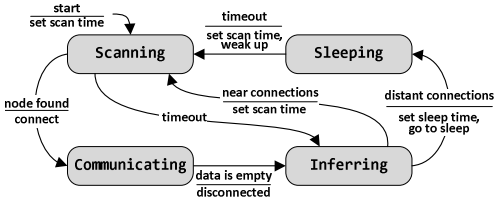

Fig. 2. FSM of energy-saving by node trajectory inference for a node operating in a DTN.

Our solution inserts the Inferring state before going to the Sleeping state, responsible for estimating future connections based on the *ConnectivityMap*. Once the previous communication finishes, the Inferring state allows a node to estimate if the next possible connection is timely distant or near. With this solution, each time the node goes to Inferring state and needs to wait for a distant connection, the *sleep time* is set to a value close to the meeting inferred time. If the connection is inferred to be near, the *scan time* is set, and the node goes to the Scanning state. This inference-based

technique can control the time adjust tradeoff, making the DTN store-and-forward technology energy efficient.

Sleep time knowledge is the main reason for reducing energy consumption without losing connections. While the classical approach does not have mechanisms to estimate the *sleep time*, this one can produce estimation for keeping communication circuits off. The success of the proposed method depends on the quality of the *connectivity inference algorithm* that relies on aspects like the number of nodes, node movement pattern, time spent learning the node movement. A loss of a meeting or an unsuccessful scanning happens when the connectivity inference algorithm sets a greater or lesser *sleep time* than the time required, respectively.

B. Inference Model

To design the connectivity inference algorithm, we assume the network of nodes corresponds to an asynchronous and distributed system $\prod = \{\boldsymbol{p_1}, ..., \boldsymbol{p_N} \mid 1 < N\}$ with N mobile nodes, without a global clock, and the nodes communicate through message exchange using radio broadcast for a finite communication range. The hardware architecture we propose for a mobile node contains the following components: (i) processing unit; (ii) memory; (iii) input/output devices responsible for data traffic; (iv) radio for sending/receiving messages; (v) sensors and actuators for communicating with the external environment; and a Positioning System (PS). The PS module allows the node to obtain and notify information about its mobility, represented by Cartesian positions added with time tags. The software architecture considers the DTN protocol stack, which is a layered model. Each layer is responsible for a group of tasks, providing a well-defined set of services for the upper-layer protocol.

The communication model regards radio messages with a limited transmission range and no signal loss. The bundle layer, implemented by the bundle protocol, handles intermittent connectivity (storing a message and carrying this message until a contact takes place). A message m, sent by a node p_i, is only received by a node p_k that is within its transmission range when p_k can identify the signal power of m. The Trajectory Inference Module (TIM – part of our software architecture) is responsible for the functionality of the connectivity inference algorithm. Fig. 3 illustrates this algorithm that receives the coordinates with their respective time tag of the nodes (readMobilityTrace) and calculates the connections among the nodes, using this information to manage the node energy efficiently. We highlight the time reference of a node p_i used to compute connectivity from the node p_i perspective and not for a global time notion.

Each instance of the algorithm executes two tasks: (i) makingConnectivityMap (**Task0**), which is a static task performed at the beginning of the simulation, and (ii) inferringStateOperation (**Task1**), which is a dynamic task performed during the simulation.

Task0 generates decision trees (*ConnectivityMap*), often used when high precision and low complexity execution are required [15], and produces the connectivity map of a given node. The task starts reading the mobility trace of the node (line 1). Then, the algorithm creates a map (connectivityMap) containing all intervals of time a given node is in a connectivity range of other nodes (lines 2-10). This procedure considers the 2D positioning of all nodes for each time (simulation step) and signal strength. To build Task0, we used data mining techniques for discovering patterns from the massive amounts of data; and used these patterns for predicting the values of dependent variables by identifying regularities and building generalizations in attributes of the dataset [16]. Decision trees were obtained through training using the J48 algorithm, an open-source implementation of the C4.5 algorithm available on *Waikato Environment for Knowledge Analysis* (WEKA) [16] – a machine learning tool.

```
Task0: makingConnectivityMap of node_i ∈ ∏
1.  readMobilityTraces(node_i)
2.  for all 1 ≤ k ≤ N | node_k ∈ ∏ {
3.      if k ≠ i {
4.          readMobilityTraces(node_k)
5.          for each time slice {
6.              if node_i(X,Y) in connectivity range of node_k(X,Y)
7.                  connectivityMap_i(time)
8.          }
9.      }
10. }

Task1: inferringStateOperation of node_i ∈ ∏
1.  at each transition to the Inferring state {
2.      timeToNextConnection ← connectivityMap_i(time)
3.      if timeToNextConnection < predefinedThresholdTime {
4.          scan time ← predefinedScanningTime
5.          go to Scanning state
6.      }
7.      else {
8.          sleep time ← timeToNextConnection – predefinedScanningTime
9.          reduce energy consumption of node_i
10.         go to Sleeping state
11. }
```

Fig. 3. Connectivity inference algorithm.

To construct our model and training set, we use the following information: X and Y axes coordinates; length, calculated from the sum of the distance formed between all pairs of consecutive trajectories points p_n and p_{n-1}; X and Y axes lower values; X and Y axes greater values; X and Y axes difference; travel duration time; displacement, calculated from the distance between the first and last trajectory points; average speed between all trajectory pairs points; average acceleration between two speeds, calculated for all trajectory points; mean slope between all points; and medium change direction between all points.

Task1 is performed each time a node goes to the Inferring state. The algorithm computes the time inferred for the meeting, stored in the variable timeToNextConnection (line 2), using the *connectivityMap*. The algorithm compares this value with a predefined threshold (predefinedThresholdTime), whose definition depends on the quality of the prediction method (lines 3-11). Hence, for small timeToNextConnection values, the node goes to the Scanning state to search for a new connection; else, the node goes to the Sleeping state, allowing saving energy.

IV. Experimental Results

This section is twofold organized. We start presenting how the experiments were organized. Next, we present the results.

A. Experiment Settings

We used the ONE simulator [18] to evaluate the impact of saving energy through the node trajectory inference in a DTN. Envision a smart city scenario; we evaluated the efficacy of the proposed algorithm in saving energy of nodes placed in vehicles with predefined routes. For this purpose, we use the CRAWDAD dataset originated from the bus fleet movement of Seattle city (Washington, USA) [14]. Although the scenario is controlled, as the bus routes are tightly defined and schedules are slightly defined, the traffic of the city is unknown, generating a high variability degree in the communication range among buses. This variability is what the connectivity inference algorithm needs to capture.

The *ConnectivityMap* is previously built and loaded on each node at the beginning of the simulation. Despite the *ConnectivetyMap* can change and be reloaded during the node lifetime, over the simulation running, no update is done.

The raw data of the bus dataset refers to date and time, bus and route identifications, and the XY coordinates of the buses. Following the instructions in [18], to use the simulator, date and time were converted to seconds, and time events sort all lines. The sampling interval (time difference between two-time events) was one second for the entire file, which was obtained by interpolating the original values of the dataset. Additionally, all trajectories were divided into trajectory segments multiple of 100 seconds (i.e., 100, 200, 300, ...), up to the total value of the trajectory itself.

The connectivity maps, containing all intervals of possible inter-node communications, are built regarding the node connectivity range. The dataset collection range is between Oct. 30 and Dec. 2 of 2001 - 33 days. The bus system consists of more than 1200 vehicles, covering an area of 5100 square kilometers. We used the interval until Nov. 23 and the remaining interval for experimentation and simulation.

This work uses the energy model consumption developed by [19], adapted for our case, that classifies the energy consumption of the wireless interface in the four states shown in Fig. 2: (i) *Communicating* - the node consumes energy while sending or receiving a message; (ii) *Scanning* - the node consumes energy while the network interface searches for neighbor nodes; (iii and iv) *Inferring* and *Sleeping* - there is minimum energy consumption in the communication interface during this two states, which is ignored by the energy model. Additionally, for simulation purposes, the initial energy of the nodes is unlimited. TABLE I summarizes the energy consumption in each state.

TABLE I. ENERGY CONSUMPTION VALUES.

State	Energy consumption (mJ)
Initial	unlimited
Scanning	4.6
Communicating	0.4
Inferring / sleeping	0

TABLE II summarizes the parameters of the simulated environment. The simulation area, number of nodes, and simulation time are based on real data extracted from [14]. The range values and transmission speed are compatible with IEEE 802.11 interfaces used outdoor, considering it can transmit up to 50 meters away at a speed of 54 Mbps. Additionally, we implemented Epidemic, Spray, and Wait, and Prophet routing protocols [9], implying different energy consumptions due to the send and receive operations.

TABLE II. SIMULATION PARAMETERS.

Parameter	Data
Simulation area [width, height]	[54818, 82860] meters
Number of nodes	1,163
Simulation time	74,897 seconds
Node movement	Jetcheva et al. [14]
Transmission range	50 meters
Transmission speed	6,912 kBps
Routing protocol	Epidemic, Spray and Wait, Prophet [9]

B. Experiment Results

When alternating the states of a connection between Communicating (receiving or transmitting) and Scanning states, and a new connection is created for the node, the operation of device discovery performed in the "scan response" unit spends the same energy of a transmission. This allows computing the cost of connecting and disconnecting the network interface and the transition between the two states. Besides, the sum of the energies consumed by all operations (TABLE I) allows computing the total energy consumption for each node and the entire network.

TABLE III shows the impact of routing algorithms on the number of sending and receiving operations for all network nodes during the entire simulation. The Epidemic algorithm is the most energy consumer since it floods the network with messages. The controlled flood of the Spray and Wait algorithm saves considerable energy. Meanwhile, the Prophet algorithm is listed as intermediate consumption among all. In this regard, the energy consumption of the inference model obtains the same results as the traditional model because it does not affect the send and receive operations.

TABLE III. NUMBER OF SEND AND RECEIVE OPERATIONS ACCORDING TO THE ROUTING ALGORITHM.

Routing algorithm	Send and receive operations
Epidemic	6,263,994
Prophet	3,358,023
Spray and Wait	1,022,653

In send and receive operations, the three evaluated algorithms consume the same energy, using or not the inferring algorithm because TIM does not interfere in this kind of operation. Similarly, our approach does not affect the metrics (i) loss of meetings and (ii) number of unsuccessful scanning because the connectivity inference algorithm, which uses decision trees, always sets an ideal *sleep time*.

In the traditional model, a node is continually scanning for new connections. It does not happen with the TIM model; this is the main difference of our work. TABLE IV highlights that considering the total scanning operations for all network nodes, the TIM model accounted for 47.51% fewer operations than the traditional one.

TABLE IV. ENERGY CONSUMPTION ON SCANNING OPERATIONS.

Routing algorithm	Energy consumption of a single node (mJ)	
	Total	(%)
Traditional	352,291.00	100.0
TIM	184,917.17	52.5
Difference	167,373.83	47.5

Fig. 4 summarizes these results showing the total energy consumed during the network simulation period for the three routing algorithms regarding traditional and TIC models. As the energy consumption displayed in Fig. 4 considers communicating and scanning operations, it is clear that the scanning operations consume the highest network energy and that the impact of communicating operations and routing algorithms in the network energy consumption is small. Additionally, the advantage of the TIC model is highlighted when compared to the energy consumed by the traditional model. As a result, this experiment shows that the proposed trajectory inference-based technique offers a significant improvement in energy management efficiency.

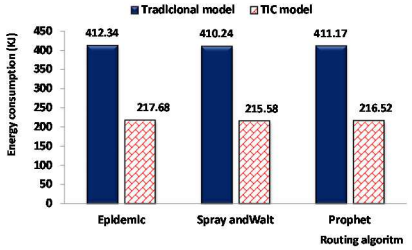

Fig. 4. Energy consumed during the network simulation regarding communicating and scanning operations.

We draw a route map based on the actual movement of the buses, allowing us to inspect intersections occurring for different tracings within the same set. Some sections in the traces have only one bus running all the way, while others have up to 52 vehicles; all routes have vehicles within these limits. Some buses use several different routes throughout their shifts, saving resources, including more buses in a specific route during intense user interval. This variability affects the network behavior because routes with fewer buses have more sparse communication opportunities among nodes.

TIM may provide promising results under sparse communication conditions. We choose two of the sparsest routes as case studies to evidence this statement: routes 187 and 773, which are fulfilled by two and four buses, respectively. TABLE V shows that the TIM approach becomes even more economical in a very sparse network, with few nodes and few encounters. While the traditional model performs 151,332 scanning operations, TIM performs only 0.08% of these operations in a network with 2 nodes and 0.76% of 192,392 scanning operations in a network with 4 nodes. Therefore, under these conditions, energy savings for the entire network are several orders of magnitude greater.

TABLE V. ENERGY CONSUMPTION FOR A SPARSE NETWORK.

Number of nodes	Routing algorithm	Scanning operations	
		Total	(%)
2	Traditional	151332	100.00
	TIM	122	0.08
4	Traditional	192392	100.00
	TIM	1468	0.76

Likewise, Fig. 5 illustrates that as the dispersion of the network nodes decreases, TIM shows higher values of scanning operations, reaching a maximum of 62.87% of the operations performed by the traditional model.

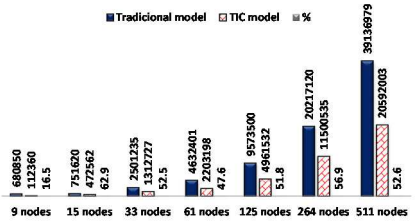

Fig. 5. Scanning operations for networks with 9 to 511 nodes.

Networks containing 2, 4, 9, 15, and 33 nodes represent individual routes, and traces of more than one route are combined. The network containing 61 nodes, for example, is the combination of route number 7 (with 52 buses) with route number 240 (with nine buses). There is no correlation between the number of nodes in the network and the values obtained by TIM because the greater or lesser number of connections among nodes affects the scanning operations, and it depends on their mobility and dispersion. Consequently, in sparse networks, some scanning operations are low; the opposite occurs in dense networks. In general, the network nodes must maintain continuous communication with the number of scanning operations to be the same in both traditional and TIM models, which is an unexpected situation in a DTN.

V. CONCLUSIONS

This paper represented a method for saving energy by inferring the trajectory of nodes in a DTN. Three separate algorithms were investigated, all of which yielded promising results. When opposed to conventional approaches, we could save more than 47 percent of energy in some activities. The results show that, regardless of the algorithm used, the node trajectory inference can be used to save energy in a DTN, thus achieving optimal results. We demonstrated that the node behavior was simulated by activating its energy to transmit or scan only when needed, such as when another node was nearby. In every other circumstance, the node disables these activity functions. This technique is analogous to a basic principle: an "energy-conscious person" who only triggers his device network interface when he discovers a link point. If the user cannot find many link points, the system spends a lot of time with the interface switched off, saving a lot of energy. Finally, this research showed that our strategy saves a large amount of energy for a DTN.

REFERENCES

[1] B. Goldberg, J. Hall, P. Pham, C. Cho, "Text messages by wireless mesh network vs voice by two-way radio in disaster simulations: A crossover randomized-controlled trial", *The American Journal of Emergency Medicine*, v. 48, pp. 148-155, Oct. 2021.

[2] A. Liu, L. Cao, Y. Han, S. Gao, X. Li, W. Zhao, "Design of a low-power road monitoring system for smart cities based on Wireless Sensor Network", *Advances in Transportation Studies*, v. 53, pp. 183-196, Apr., 2021.

[3] S. Das, K. Sinha, N. Mukherjee, B. Sinha, "Delay and Disruption Tolerant Networks: A Brief Survey", *Intelligent and Cloud Computing*, v. 194, pp. 297-305, Oct. 2020.

[4] T. Abdelkader, K. Naik, A. Nayak, N. Goel, V. Srivastava, "A Performance Comparison of Delay-tolerant Network Routing Protocols", *IEEE Network*, v. 30, n. 2, pp. 46-53, 2016.

[5] J. Rodrigues, "Advances in Delay-tolerant Networks (DTNs): Architecture and Enhanced Performance", *Woodhead Publishing Series in Electronic and Optical Materials: n. 67, Elsevier Science*, 298p., 2014.

[6] S. Borah, S. Dhurandher, I. Woungang, N. Kandhoul, J. Rodrigues, "An Energy-Efficient Location Prediction-Based Forwarding Scheme for Opportunistic Networks", *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1-6, 2018.

[7] Z. Chunyue, H. Tian, Y. Dong, B. Zhong, "An energy-saving routing algorithm for opportunity networks based on asynchronous sleeping mode", *Computers & Electrical Engineering*, v. 92, pp. 1-11, Jun. 2021.

[8] A. De Vit, C. Marcon, R. Nunes, T. Webber, G. Sanchez, R, Rolim, "Energy Saving on DTN Using Trajectory Inference Model", *Proceedings of the Annual ACM Symposium on Applied Computing (SAC)*, pp. 1-4, 2018.

[9] N. V. R. Reddy, C. Sivasankar, N. Divya Jyothi, "A comprehensive survey on replication based protocols of delay tolerant networks," *International Conference on Communication and Signal Processing (ICCSP)*, pp. 0014-0018, 2016.

[10] N. Derakhshanfard, R. Soltani, "Opportunistic routing in wireless networks using bitmap-based weighted tree", *Computer Networks*, v. 188, pp. 1-8, Apr. 2021.

[11] L. Wu, S. Cao, Y. Chen, J. Cui, Y. Chang, "An adaptive multiple spray-and-wait routing algorithm based on social circles in delay tolerant networks", *Computer Networks*, v. 189, pp. 1-15, Apr. 2021.

[12] N. Banerjee, M. Corner, B. Levine, "Design and Field Experimentation of an Energy-Efficient Architecture for DTN Throwboxes", *IEEE/ACM Transactions on Networking*, v. 18, n. 2, pp. 554-567, Apr. 2010.

[13] A. Rudenko, L. Palmieri, M. Herman, KM. Kitani, DM. Gavrila, KO. Arras. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*. v. 39(8), pp. 895-935, 2020.

[14] J. Jetcheva, Y. Hu, S. PalChaudhuri, A. Saha, D. Johnson, "CRAWDAD dataset rice/ad_hoc_city (v. 2003-09-11)", available at crawdad.org/rice/ad_hoc_city/20030911/, Mar. 2021.

[15] Wu, Xindong, et al., "Top 10 algorithms in data mining." *Knowledge and information systems* 14.1, pp. 1-37, 2008.

[16] J. Han, M. Kamber, J. Pei, "Data mining: concepts and techniques", 3rd ed., Elsevier, 703p., 2012.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update". *SIGKDD Explorations*, Volume 11, Issue 1, pp. 10-18, 2009.

[18] A. Keränen, T. Kärkkäinen, M. Pitkänen, F. Ekman, J. Karvo, J. Ott, "The One Simulator Javadoc", available at www.netlab.tkk.fi/tutkimus/dtn/theone/javadoc_v141/, Mar. 2021.

[19] D. Silva, A. Costa, J. Macedo, "Energy Impact Analysis on DTN Routing Protocols", in Proceedings of the *ACM Extreme Conference on Communication* (ExtremeCom), 6p., 2012.