Identifying User Communities Using Deep Learning and Its Application to Opportunistic Networking

Danielle L. Ferreira [*], Cláudio de Souza [†], Katia Obraczka [§], and Carlos Alberto V. Campos [*]

[*]Federal University of the State of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
[†]Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
[§]University of California, Santa Cruz, CA, USA
Email: danielle.ferreira@uniriotec.br, disouza@cos.ufrj.br, katia@soe.ucsc.edu, beto@uniriotec.br

***Abstract*—Opportunistic networking has been proposed to address episodic connectivity, common in so-called "challenged" or "extreme" networking environments where arbitrarily frequent and long-lived connectivity disruptions are the norm, instead of the exception. Examples of extreme networking environments and applications include interplanetary communication, disaster relief, and emergency response, (semi-)autonomous driving, to name a few. This paper proposes a novel community-based opportunistic routing approach that identifies user communities based on mobility features extracted from real traces of user mobility. The proposed Deep AutoenCoder Community-based Opportunistic Routing protocol, or DACCOR, employs deep learning to identify user communities based on data extracted from user mobility traces and uses user community information to make forwarding decisions in opportunistic networking scenarios. Through extensive simulations, we evaluate DACCOR's performance and show that it outperforms well-known opportunistic forwarding protocols in terms of delivery probability, latency, and communication overhead. We also show that DACCOR's lower communication overhead yields considerable energy efficiency, increasing mobile devices' battery lifetime.**

I. Introduction

Despite the ever increasing ubiquity and availability of wireless communication, there are still applications and scenarios where network connectivity is intermittent and may suffer from arbitrarily frequent and long-lived disruptions. Notable examples include environments where fixed infrastructure is sparse (e.g., rural- and under-developed regions) and areas where the wireless channel is prone to propagation impairments (e.g., due to obstacles in the case of dense urban regions, interference, etc).

Opportunistic networking has emerged as a way to address connectivity-challenged environments by considering node mobility an opportunity instead of a challenge [6]. In opportunistic networks, temporary and occasional contacts between users through their mobile devices present themselves as data transmission opportunities, and user mobility serves as a way to carry data from its source to its destination directly or through intermediate nodes. As such, opportunistic forwarding schemes should be able to send messages to a select set of intermediate relays so that chances of successful data delivery at the destination are maximized. Another important goal is to keep network overhead as low as possible, avoiding unnecessary transmissions. The importance of reducing overhead in opportunistic networking environments is due to the fact that mobile devices have limited battery power. Reducing unnecessary radio transmissions can substantially decrease energy consumption and increase battery life. Thus, one of the main challenges in opportunistic networks is to forward data to relay nodes that have the best chance to encounter the destination, while limiting the number of copies being relayed in the network.

Opportunistic routing/forwarding protocols seek to discover information about users to assist in deciding when and to whom messages should be forwarded [27], [1], [13], [5]. It has been demonstrated that user community based schemes improve forwarding messages in specific scenarios [27]. BUBBLE [9] was one of the first community-based opportunistic forwarding protocols. It uses the well-known centrality metric and a community structure to forward data. More recent works, such as [25], [26], [18] also use community-based protocols to forward data.

Most community detection schemes to-date extract user information via inter-contact time between users [7] (i.e. peer contact) or use information obtained through social networks or through network service providers [19], [17] (e.g., examining logs that record call information made by mobile phones). In our work, we identify communities exclusively through user mobility behavior, rather than depending on information obtained from external sources. Additionally, to our knowledge, most efforts to-date do not account for user geographical preference when identifying user communities.

In this paper, we introduce a community-based opportunistic routing approach that identifies user communities based on mobility features extracted from traces that record user mobility in real and diverse environments. The proposed Deep AutoenCoder Community-based Opportunistic Routing protocol, or DACCOR employs a deep learning approach to identify user communities based on data extracted from user mobility traces and uses user community information to make forwarding decisions in opportunistic networking scenarios.

The main contributions of DACCOR can be summarized as follows:

- It introduces a user mobility feature extraction technique based on information obtained from traces of real user mobility. The resulting user mobility features will then be used by DACCOR's novel deep learning based algorithm to identify user communities.

- It proposes a metric to measure affinity between users and user communities.
- It develops a deep learning based approach to identify user communities based on features extracted from mobility traces.
- It proposes an opportunistic routing protocol that uses user community information to decide how to forward data. Through extensive experimentation using synthetic- and real mobility records representing diverse urban mobility scenarios, we show that DACCOR is able to outperform other well-known opportunistic routing protocols by delivering more data, faster, and using less networking resources.

The remainder of this paper is organized as follows. Section II presents DACCOR's user mobility feature extraction technique as well as DACCOR's deep learning based community identification mechanism. Section III describes DACCOR's routing protocol which uses two data forwarding strategies, namely: inter- and intra-community forwarding. In Section IV, we describe the experimental methodology we use to evaluate DACCOR's performance, including the user mobility datasets and performance metrics. Section V presents results from our performance evaluation study. In Section VI, we review related work and in Section VII, we conclude the paper with some directions of future work.

II. Deep AutoenCoder Community Detection

This section provides a detailed description of the proposed Deep AutoenCoder Community-based Opportunistic Routing protocol (DACCOR) for data forwarding in opportunistic networks. We start by describing how user features and preferences are extracted from raw mobility traces. Then, we present a novel, deep-learning-based approach that identifies user communities based on user mobility features extracted from the mobility traces. Once user communities are identified, we then propose a similarity metric to measure levels of similarity and dissimilarities between members within a community and between distinct communities.

1 presents an overview of DACCOR. DACCOR's data forwarding component makes inter- and intra-community routing decisions based on the user community structure it builds. Inter-community routing uses members of the destination community as relays, while intra-community routing forwards messages to nodes that have higher geographical preference similarity with the destination node when compared to the geographical preference of the current node. For instance, in Figure 1, node A (from the orange community) has a message to deliver to node D (from the blue community) and encounters node B (from the blue community). In this case, the inter-community forwarding decision will be made based on community labels assigned by DACCOR's deep learning based community identification component. Next, node B encounters node C (also from the blue community). Since they both belong to the same community as node D, the forwarding decision will be made by comparing how similar B's geographical preferences are to D's versus how C's geographical preferences are to D's.

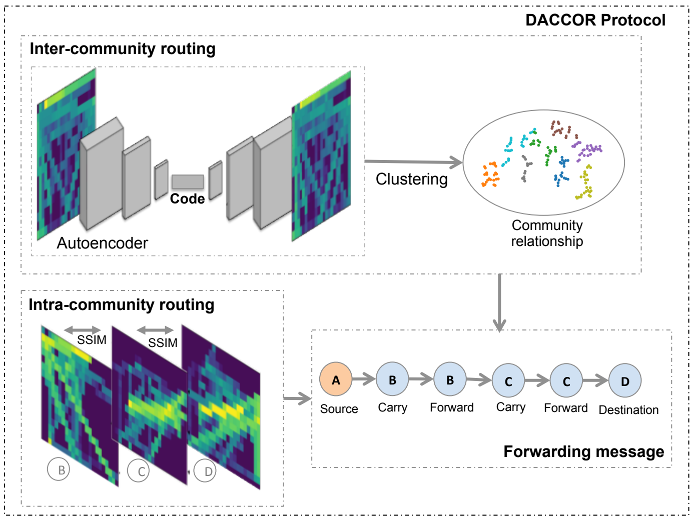


Fig. 1. The DACCOR protocol

A. *Extracting user features from mobility traces*

Mobility traces provide mobile device location information over time. By analyzing such information it is possible to obtain user mobility characteristics, including distance traveled by the user, distance to other devices, time they are within communication range of one another (a.k.a., contact time), etc. Since people move with a certain purpose (e.g. go to work, go home after work), we assume that their locations and mobility characteristics also reflect their interests and preferences. Conversely, if users do not share features and places in common, we assume they have different interests and thus are less likely to have social relationships. As such, we use mobility traces to identify *communities* of users with similar mobility behavior and geographical preferences/interests. We contend that extracting and using features from real mobility traces allows us to design more realistic mobility models, and consequently more efficient communication networks, their protocols and applications.

DACCOR extracts user mobility features from traces (e.g., GPS- and WiFi records) following a set of steps, namely: (1) It extracts the two-dimensional maximum and minimum limits of the area defined in the trace and divides this area into equal sized squared cells, constructing a spatial-temporal feature matrix. Such matrix holds user geographical and temporal characteristics. The i-th row of the matrix represents the i-th user $\forall i \in [1, I]$, where I is the maximum number of users. The c-th column of the matrix represents the c-th cell $\forall c \in [1, C]$, where C is the maximum number of cells. Each (i, c) matrix position holds the time spent by user i in cell c. It is important to note that this feature matrix must be normalized so that all attributes are uniformly accounted for; (2) Once the normalized feature matrix is constructed, a nonlinear transformation is applied using the logarithmic likelihood function *l*ogit. When applied to the normalized data, this transformation modifies the feature matrix values so that data between $(0, 1)$ takes real values, between $(-\infty, \infty)$, and is symmetric at 0.5. This transformation, besides evidencing the differences and similarities between the observations for each variable, also improves pattern identification and learning, as it will become clear in the next section; (3) Finally, the i-th image for each node i, by reshaping the i-th row of the mobility feature matrix into a 2D-image, that reflects the dimensions of the area of the trace in cells. To construct this image, we consider each c-th position of the i-th row of the feature matrix as a pixel, where the value of each position corresponds to the intensity of that pixel. These images reflect actual user displacement and thus indicate user movement patterns and geographical preferences. They serve as input to DACCOR's deep neural network module, as discussed in the next section.

B. *Identifying user community structures using deep learning*

Deep learning (DL) models have been widely employed in recent years by researchers and practitioners to solve a plethora of different problems in many areas [12], [2], [15]. Identifying user community structures from raw mobility data requires unsupervised learning approaches since, most of the time, there is no previous knowledge from these raw records about the nature of the relationship between users, whether they belong to certain communities, etc. An autoencoder is a neural network architecture designed to learn data encodings in an unsupervised fashion. It is typically used for dimensionality reduction, where the complexity and variability of the data is reduced into an encoded, more compact representation [3]. Along with data reduction, there is also a reconstruction step that tries to reconstruct a representation as close as possible to the original input. In other words, the autoencoder takes a set of unlabeled data $x \in R^n$ and tries to learn an approximation to the identity function to force the output to be as similar as possible to the input.

Autoencoders consist of three basic general components: (1) the encoder, that is the portion before the most compressed layer (or code) of the architecture. It compresses the input vector x into a latent representation h using a weight matrix ω; (2) the code, h, or latent space representation, is a lower-dimensionality representation of the input. This reduced representation allows interesting data features to be uncovered; and (3) the decoder which maps h back to the input, reconstructing it to obtain x' with another weight matrix ω'. Parameter optimizations are used to minimize the average reconstruction error between x and x'. Usually, the input and output layers have the same dimensionality. One category of neural network that is widely used for image processing tasks is the convolutional autoencoder (CAE). CAEs are designed to process data inputs in the form of multidimensional arrays, e.g. images composed of 2D arrays containing pixel intensities in color channels. CAEs use the same principle as traditional autoencoders discussed above, but instead of fully-connected layers, it contains convolutional layers in the encoder and deconvolutional layers in the decoder. The vast majority of applications of convolutional neural networks focus on image data, which is also the case in our work.

1) Convolutional autoencoder design: Training the neural network means learning the weight matrix ω' associated with all the neurons in the network. The basic unit of computation in a neural network is the neuron, often called a node or unit. During the training, each unit located in any layer in between input and output layers, also called hidden layers, receives several inputs from the preceding layer. Analogously, CAE architectures are structured in several stages of convolutional and pooling layers [21]. The units in a CAE are organized in features maps, also known as convolutional filters or even convolutional kernels, that are connected through a set of weights between the layers.

The unit computes the weighted sum of these inputs and eventually applies an activation function, to produce the output. The output of all, except the last of our convolutional layers are activated by a *Rectified Linear Unit* (ReLU) activation function, where the output is $f(x) = max(0, x)$. Only the last layer, the output layer, is activated by a linear function. The non-linear behavior of neural networks comes from the choice of these activation functions. Other popular ones are *Linear, Logistic, ReLU, SELU*, and *Tanh*. In this way, the convolutional autoencoder is able to detect local groups of values in an array of images that are often highly correlated, and also detect spatial invariance patterns. In other words, if a pattern is identified in a part of an image, it could appear also in other parts. Hence, the convolutional layer is responsible for detecting patterns from the previous layer, and the pooling layer for merging semantically similar features to one. In CAE, the pooling layer is responsible for reducing the dimension of the representation and creating an invariance to small shifts and distortions on the images.

After these steps, the output x' (reconstructed node's trajectory image) is compared to the input x (original node's trajectory image), and the error will be propagated to every individual unit using the back-propagation algorithm [12]. Finally, each weight's contribution to the error is calculated and the gradient descendent algorithm is adopted to adjust the parameters at each layer (i.e., update the weights). We trained our autoencoder to minimize the mean square error and the optimizer used was *Adam*.

Usually, CAEs contain two or three stages of convolutional layers, non-linear activations and pooling layers, followed by more convolutional and/or fully-connected layers. Our proposed architecture contains a convolutional network with three 2D convolutional layers on the encoder, followed by a fully-connected layer in the latent space, and three symmetric 2D convolutional layers to reconstruct the input. This network has the following structure: the three convolutional layers on the encoder side contain 128, 64 and 32 filters with sizes that varies from (3x3) to (5x5) depending on the shape of the input image, and strides over 2x2-pixels. The latent layer contains a flatten layer with 8 units. The reshape image has size N = 1 x 2 and 128 filters. This leads to feature representations of dimensionality D = 8, which were used as input into the clustering algorithm. The deconvolutional layer is symmetric to the convolutional one with similar parameters.

2) Clustering: DACCOR employs the Model-Based Clustering (MBC) algorithm to detect community structures from the lower-dimensionality representation of the input obtained from training the autoencoder (see Figure 1). MBC [16] is a well-established method for cluster analysis and unsupervised learning. It assumes a probabilistic model (e.g., mixture model) for the data and then estimates the model parameters by optimizing an objective function (e.g., model likelihood). To use the MBC method for clustering data as well as automatically selecting the number of clusters, K, it is necessary to generate a set of candidate models. The Expectation-Maximization (EM) algorithm is often used for estimating the parameters of the model, where clusters are centered at the mean value, and the geometric features (shape, volume, and orientation) are given by the covariance matrix.

MBC-based clustering algorithms have used the Gaussian distribution to model clusters using three parameters: mean vector, covariance matrix and an associated probability in the mixture, where each point has a probability of belonging to each cluster. The algorithm consists of the following steps: (1) During initialization, it is necessary to specify the number of clusters and randomly initialize the distribution parameters for each group. The agglomerative hierarchical clustering is used to obtain the initial partitions of the data; (2) Then, the probability that each data point belongs to a particular cluster is computed; (3) The EM algorithm is applied, which is based on a maximum likelihood estimate used to estimate the likelihood of the mixture parameters; (4) Finally, in the case that the covariance matrix of the components lead to different models, the BIC (Bayesian Information Criterion) is applied to choose the best model.

C. Measuring community- and user afinity

Image quality comparison metrics work in our case as similarity indexes for spatial displacement since the images we use as input for our autoencoder architecture can be seen as heat-maps of each user's geographical preferences (i.e., time spent at a given location). Analogously, we seek to establish, rather than a spatial, also a temporal relationship metric. We argue that users belonging to the same group would spend more time together, as they would share similar interests, routes and geographical preferences, even though we only used data about users' individual geographical preferences in order to form groups.

In this way, we use the Structural SIMilarity (SSIM) Index to compute the similarity in the mobility behaviour of two nodes by comparing the similarities between the two user trajectory images. The SSIM index identifies the information structures found in the images and therefore is used to compute the similarity between a pair of nodes. The SSIM algorithm compares point by point two images aligned and scaled. Three similarity functions are computed on the image data: (1) luminance similarity, (2) contrast similarity, and (3) structural similarity. Note that the more the value of SSIM approaches 1, the more similar are the two images, and also, the more similar are the attributes of the two nodes.

The similarity for two images X and Y can be calculated as follows:

$$SSIM(x,y) = [l(x,y)]^{\alpha} \ldots [c(x,y)]^{\beta}[s(x,y)]^{\gamma} \quad (1)$$

where the luminance comparison function $l(x,y)$ is a functions of the mean intensity of image x and y and is given by

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C1} \quad (2)$$

The contrast comparison function $c(x,y)$ is the comparison of the standard deviation intensity of image x and y and is given by

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C2} \quad (3)$$

And, the structure comparison function $s(x,y)$ is defined as

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x + \sigma_y + C3} \quad (4)$$

where σ_{xy} can be estimated as

$$\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \quad (5)$$

The constants C1, C2 and C3 are small constants that provide stability when the denominator approaches zero. More details about the SSIM algorithm can be obtained in [24].

In summary, the encoding of features towards extracting user mobility features using deep-autoencoder consists of the following steps: (1) generate the mobility image based on the feature matrix extracted from the real trace; (2) construct the deep autoencoder architecture for the trace. It is not possible to train a single architecture for general use, since the models depend on the size of the input, and it varies with the application scenario (i.e., the size of the area, number of cells and feature matrix changes from scenario to scenario); (3) train the deep autoencoder by using the input image representation obtained from the mobility features described above; (4) once the network is trained, extract the reduced features (code) from the autoencoder latent representation space. These features can be used to make predictions, and comparing the original input with the reconstructed image; (5) use the reduced latent feature representation as input for the MBC clustering algorithm; (6) extract community labels from the clustering algorithm; (7) extract node similarity from SSIM metric.

We now have extracted mobility patterns and are in possession of community structure and relationship indicators between each pair of nodes. The community structure is indicated by the community labels given by the clustering algorithm. The relationship between every node is given by the SSIM index values computed for every pair of nodes, which indicates their geographical preferences similarity. We can take advantage of these information to make more intelligent and educated decisions on when and to whom forward a message to in the context of opportunistic and delay tolerant networks. The following sections introduce DACCOR, a new forwarding protocol that takes advantage of the community structure information.

III. DACCOR Data Forwarding Protocol

The proposed DACCOR forwarding scheme uses community information to make forwarding decisions between members of different communities and user geographical preferences (based on the SSIM metric presented in Section II-C) to make forwarding decisions within the same community. This approach improves DACCOR's scalability since users do not have to carry geographical preferences from all other users in the network; they just need to have information about members of their own community.

A. *Relay Node Selection*

We consider the following features to determine the suitability of a relaying node to carry a message to the destination:

1. Node community ID (cID) - The community label carried by each node and extracted by the proposed deep learning based community identification technique as detailed in Section II-B.

2. Community affinity (CA) - The affinity between the communities of the encountered node and destination node. It is computed by averaging the SSIM for pairs of nodes belonging to the two communities (i.e., the community of the encountered node and the community of the destination node), as further discussed in the following Section III-B.

3. Node similarity - The similarity between the encountered node and the destination node in the case that both nodes belong to the same community. It is computed using the SSIM index for each pair of nodes in the community as described in Section II-C.

We assume that all nodes know the information about which community it belongs to, the community affinity between all communities in the network, and the SSIM metric of all members of their own community.

Data forwarding will be carried out by combining two strategies: (1) Inter-community routing based on community afinity and (2) Intra-community based on node afinity.

B. *Inter-Community Routing*

User communities are identified based on the user mobility patterns as described in Section II-B. Every node knows which community it belongs to. The inter-community forwarding scheme uses community affinity information to make forwarding decisions. In DACCOR, each node maintains an N x N community affinity matrix, which is defined as:

$$CA = \begin{bmatrix} C_{11} & C_{12} & ... & C_{1N} \\ C_{21} & C_{22} & ... & C_{2N} \\ ... & ... & ... & .... \\ C_{N1} & C_{N2} & ... & C_{NN} \end{bmatrix},$$

where $C_{n,m}$ denotes the community affinity between community n and community m, for all n and $m \in \{1..N\}$, where N is the total number of communities. $C_{n,m}$ is obtained by

$$C_{nm} = \begin{cases} 1, & \text{if } n = m \\ E(\text{SSIM}_{n,m}), & \text{if } n \neq m. \end{cases}$$

$E(\text{SSIM}_{n,m})$ is the average of SSIM between pairs of nodes belonging to communities n and m. Thus, the community affinity matrix CA determines how message forwarding between nodes belonging to different communities, i.e. nodes that have different communities ID (cID) will be carried out.

Algorithm 1 Inter-community Routing

```
Input: $node_d$, $node_i$, $node_j$, community ID (cID), $M$, CA
while $node_d$ without M do
    current $node_i$ encounters $node_j$ without $M$
    if $node_j$ is $node_d$ then
        $node_j$ accepts $M$ from $node_i$ and the forwarding process ends
    else
        if $node_j$ and $node_d$ in C then
            $node_i$ forwards $m$ to $node_j$
        else
            if CA($cID_j$,$cID_d$) > CA($cID_i$,$cID_d$) then
                $node_i$ forwards $m$ to $node_j$
            end
        end
    end
end
```

For example, assume that a node i has a message to destination node d and encounters with node j. Also, assume that node i and node d belong to different communities, n and m, respectively, i.e. $C_{n,m} \neq 1$. If j is the destination (i.e., $j = d$), i will forward the message M to j. If node j belongs to the same community of node d, then node i forwards the message to node j. If that is not the case, node i will consult the community affinity matrix to decide if it is going to forward M to node j or not, i.e. if $C_{j,d}$ > $C_{i,d}$ then i is going to forward M to j. Otherwise, i will not forward M and continue to carry message M. When the message has reached the destination's community, DACCOR looks for relay nodes that are more similar to the destination node, by using the SSIM metric as discussed in Section II-C. Algorithm 1 shows the detailed forwarding algorithm for the first phase.

C. *Intra-Community Routing*

For intra-community routing, the proposed protocol uses the $SSIM$ metric defined in Section II-C to determine each node's message forwarding "fitness" with respect to the message's destination. Thus, the $SSIM$ metric dictates the message forwarding between members of the same community.

For instance, assume there is a node i carrying a message, an encountered node j and a destination node d, all belonging to the same community. DACCOR will search for the higher geographic similarity, by comparing the SSIM metric between the two nodes and the destination node. In other words, if SSIM($node_j$, $node_d$) > SSIM($node_i$, $node_d$) the message is forwarded from node i to node j. In addition, to increase the probability of message delivery, even if node j and node d are not part of the same community, the message can be

Algorithm 2 Intra-community Routing

```
Input: $node_d$, $node_i$, $node_j$, community ID (cID), $M$, CA, SSIM, E[CA]
while $node_d$ without m do
    current $node_i$ encouters $node_j$ without $M$
    if $node_j$ is $node_d$ then
        $node_j$ accepts $M$ from $node_i$ and the forwarding process ends
    else
        if $node_j$ is in $C$ and SSIM($node_j$,$node_d$) > SSIM($node_i$,$node_d$) then
            $node_i$ forwards $M$ to $node_j$
        else
            if CA($cID_j$,$cID_d$) > E(CA) then
                $node_i$ forwards $M$ to $node_j$
            end
        end
    end
end
```

forwarded using the community affinity metric. We understand that a node may have a social relationship with more than one community on the network and therefore if node j has community affinity that is greater than the average community affinity for all communities in the network, the message will be forwarded. Algorithm 2 shows the detailed forwarding algorithm for the second phase.

IV. Performance Evaluation

We evaluate DACCOR's data forwarding mechanism against two well-known opportunistic routing approaches, namely PRoPHET [14] and Epidemic [22]. We use delivery ratio, average latency, hop count and overhead as performance metrics and drive our experiments using both synthetic and real mobility records. Our rationale for choosing Epidemic and PRoPHET in our comparative performance study of DACCOR is as follows. Epidemic, despite its limitations, has been widely used to evaluate opportunistic networks and serves as the upper bound for reliability as well as cost. It stores messages locally and uses opportunistic encounters to relay messages with the goal that they will eventually reach their destination.

PRoPHET uses the delivery predictability metric based on historical contact frequency between nodes to choose the next relay nodes. The difference between DACCOR and PRoPHET is that DACCOR selects the next relay nodes based on the geographical similarity, whereas PRoPHET relies on node encounter history to estimate which node has the highest "likelihood" of being able to deliver a message to the final destination. PRoPHET is a non-oblivious benchmark that has been evaluated against several previous works, including social-based protocols. For example, BubbleRap [9], the first social-based protocol, compares its performance against PRoPHET. As a result, the authors found a similar delivery ratio to PRoPHET with half of the PRoPHET cost.

The following section presents the evaluation scenarios and experimental setups used in our evaluation.

A. *Experimental Datasets*

To illustrate our approach, the datasets used in this study were selected to cover a range of scenarios considering vehicular and human mobile networks: GeoLife [28], San Francisco (SF) cabs [20] and Helsink [11].

The GeoLife trace, refers to mobility in various scenarios in the city of Beijing, including different modes of transportation (e.g. walking, cycling and driving). The vehicular mobility record is related to the movement of taxis in the city of San Francisco/USA. The GeoLife and SF traces represent GPS users trajectories. GeoLife was collected over a period of three years and sampled every 5 seconds, and SF was collected for 24 days with samples ranging from 1 to 3 minutes.

Trace	# users	Type	Speed (km/h)
GeoLife [28]	169	-	-
SF Taxis [20]	483	-	-
Helsink [11]	80	Pedestrians	1.8 to 5.4
	40	Cars	10 to 50
	6	Trams	25 to 36

TABLE I
SUMMARY OF USER MOBILITY TRACES CONSIDERED IN OUR STUDY.

Helsink is a synthetic mobility trace available in the ONE simulator. Nodes move on the simulation area according to a mobility trace generator, where 80 are pedestrians, 40 are cars and 6 are trams. In this scenario, trams follow predefined routes defined by the simulator, while pedestrians and cars choose random destinations in their reach on the map and move towards theirs next destination by following a shortest path algorithm, such as Dijkstra algorithm.

A summary of the traces are shown in Table I.

B. *Experimental setup*

We designed simulation experiments with the Opportunistic Network Environment (ONE) simulator. Random source nodes generate messages to a randomly chosen destination on average once every interval of time. The length of such interval was varied in order to change network load conditions, i.e., smaller inter-message periods allow a greater load, while larger intervals decrease the load in the network. Inter-message periods are randomly chosen over the following average intervals: 6, 8, 12, 18, 60 and 600 seconds (uniformly distributed). Message sizes are uniformly distributed between 500 KB and 1 MB. Given average message sizes, inter-message transmission intervals and channel capacity, these intervals were chosen to represent the network load ranging from 0.1 to 1 proportion of the channel capacity. A summary with these parameters are shown in Table II.

Parameters	values
Transmission rate	2 Mbps
Radio range	150m
TTL	12h
Buffer size	1GB
Simulation time	12h
Message sizes	500KB to 1000KB

TABLE II
SIMULATION PARAMETERS.

In all experiments, we compare each protocol using the following routing metrics.

- *Delivery probability:* computed as the total number of successfully delivered messages in the networks, divided by the total number of messages created.
- *Overhead:* the total number of relayed messages in the network, divided by the total number of successfully delivered message, i.e. the amount of transfers required to perform one successful delivery.
- *Latency:* the average elapsed time (seconds) from the instant a message is generated to its successful delivery at the destination.
- *Hop count:* the average number of hops for each successful delivery.
- *Buffer time:* the average duration of time (seconds) a message spend in a buffer.
- *Dropped Message:* number of messages dropped due to buffer overflow.

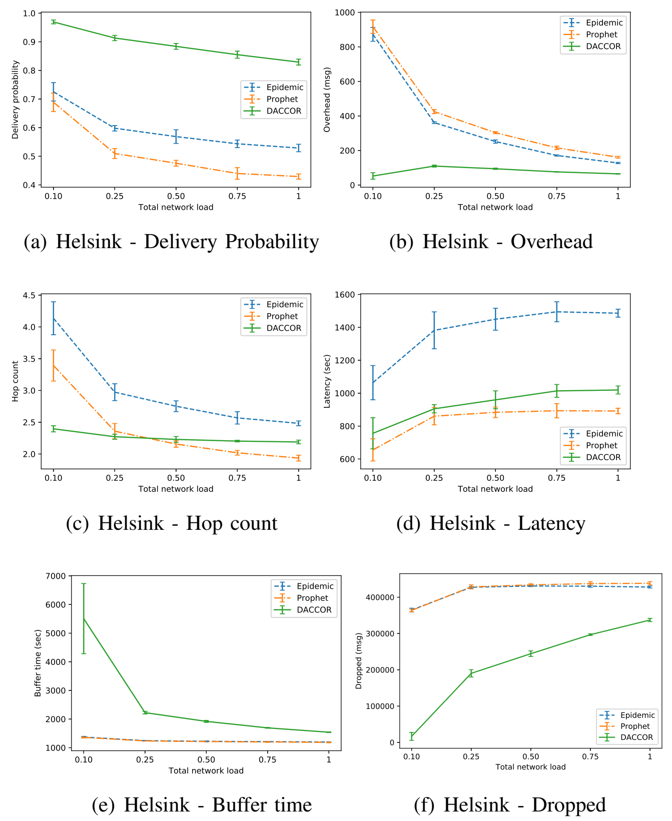
V. Results

Results are reported here for the Helsink, San Francisco and GeoLife mobility traces with a 95% confidence interval over 10 runs for each network load. We randomize the traffic scenarios by varying the source and destination pairs of the flows in each of the 10 runs.

Figures 2, 4 and 3 show the performance of the network metrics varying the total network load for Helsink, GeoLife and San Francisco mobility scenarios. We can see from those figures that DACCOR achieves the highest delivery ratio with the lowest overhead and Hop count when compared to other protocols. The only exception is for Geolife scenario considering a network load of 0.1. We argue that this reduced delivery ratio can be explained by the scenario's sparsity associated with the selective nature of DACCOR, which does not forward messages until it encounters a node with the same geographical preferences as the destination node. In fact for this scenario, where there are few messages to be forwarded and a low contact number between nodes, the best performing protocol is Epidemic. However, even though Epidemic achieves best delivery probabilities for low load, it costs more unnecessary transmissions and hops, dramatically increasing battery consumption of mobile devices and at the expense of network bandwidth, as shown in Figures 4(b) and 4(c), respectively.

It should be noted that the overhead metric considers only the messages delivered to calculate the number of messages replicated on the network. Therefore, we can observe that as the number of messages dropped on the network increases in Figures 2(f), 3(f), and 4(f), the overhead on the network decreases in 2(b), 3(b), and 4(b). In other words, since there are fewer messages delivered on the network, there is also a smaller value of relayed messages counted in the overhead metric as the load increases.

Figures 2(e), 3(e), and 4(e) show the buffer time for messages that were not delivered to the network due to buffer overflow. The message discard policy is FIFO. We note that DACCOR has the longest buffer time, especially for low load, as it is more selective and therefore tends to buffer



(a) Helsink - Delivery Probability
(b) Helsink - Overhead
(c) Helsink - Hop count
(d) Helsink - Latency
(e) Helsink - Buffer time
(f) Helsink - Dropped

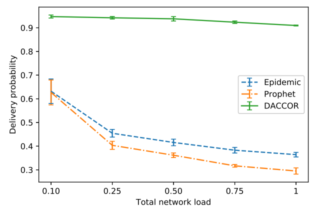
Fig. 2. Performance evaluation for buffer size of 1GB and no TTL for DACCOR, Prophet, and Epidemic protocols under Helsink scenario.

messages longer. We can argue that in the case of the Epidemic protocol, which forwards messages at each encounter, when the node buffer fills the buffer time approaches the meeting time between nodes. It is worth noting that Figures 3(e), and 4(e) do not have values for buffer time up to a load of 0.25 for DACCOR protocol. This is because DACCOR protocol does not reach the buffer limit, so there is no message loss until this message generation rate. A possible downside for DACCOR due to its selective behavior when forwarding messages is the average latency for message delivery. However, the protocol has clear advantages over other metrics.

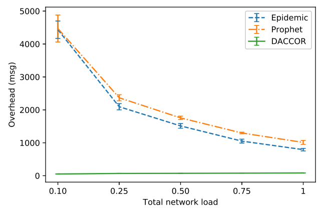
Results presented in this section confirm the efficiency of introducing geographical preference in the design of community based routing schemes. It is also worth to emphasize the extremely reduced energy fingerprint of DACCOR, as observed by the network overhead. With an overhead that is orders of magnitude lower than the other protocols, DACCOR activates the devices radio much less, saving energy and increasing battery life.

VI. Related Work

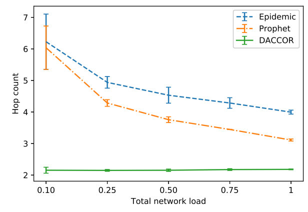
In opportunistic networks, best forwarding nodes are chosen based on chances (utility of node) they have to delivery a message to their destination. Next, a strategy to forward message to relay nodes with high utility and lower cost has to be taken. Several forwarding messages protocols on



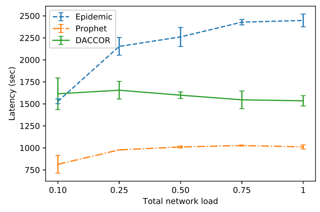
(a) SF - Delivery Probability



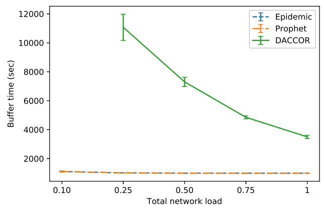
(b) SF - Overhead



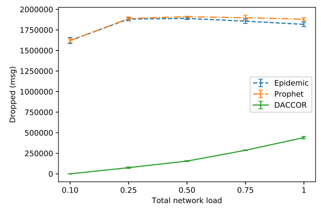
(c) SF - Hop count



(d) SF - Latency

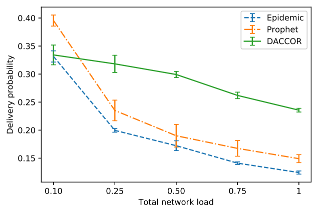


(e) SF - Buffer time

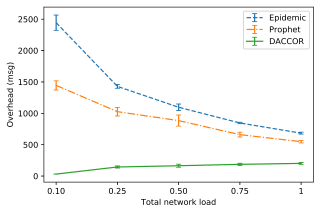


(f) SF - Dropped

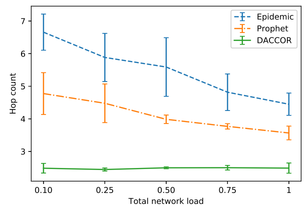
Fig. 3. Performance evaluation for DACCOR, Prophet, and Epidemic protocols under San Francisco scenario.



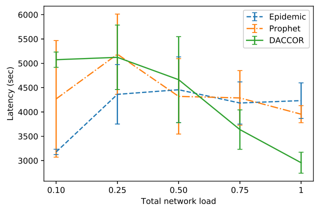
(a) GL - Delivery Probability



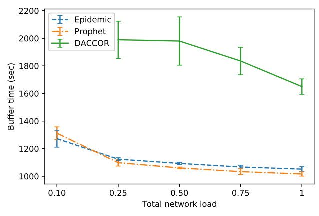
(b) GL - Overhead



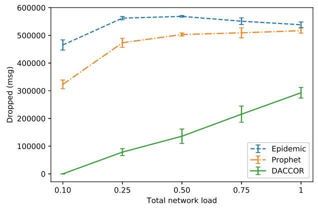
(c) GL - Hop count



(d) GL - Latency



(e) GL - Buffer time



(f) GL - Dropped

Fig. 4. Performance evaluation for DACCOR, Prophet, and Epidemic protocols under GeoLife scenario.

opportunistic networks, including DACCOR, use the concept of communities as a strategy for choosing the next hop.

The use social relationship information for selecting the best relay node was used in [27], [1], [13], [5]. The first community-based proposed was BUBBLE [9], that uses the well known centrality metric and community structure to forward data. More recent works, such as [25], [26] to name a few, still proposing community based protocols to forwarding data. For example, [18] uses relationship information containing node's profile (such as name, address, workplace, hobbies, etc.) to calculate the probability with destination.

In [23] the relay node is selected in the neighbourhood based on the highest probability to reach the destination. The probability is calculated, using information that the sender knows about the destination, based on the behavior of repeating patterns at different times during day, week, and month. Authors in [8] proposed effective schemes that consider the existence of other relays carrying replicas of the same message in the network. The schemes eliminate this redundancy with some global network information. The authors found an interesting result, they observed that some messages replicas contribute little on improving the delivery ratio. The impact of less popular nodes on the diffusion of messages on the network was studied in [26]. More specifically, the authors removed nodes that were less "important" (low centrality) and found that message delivery performance was degraded.

Hui and others [10] proposed a distributed detection scheme for Pocket Switched Networks, where each device senses and detects its own community by analyzing the mobile device history it encountered. Just encounter events are used to build social relationship between them. This work use data obtained from opportunistic networks traces, which only contains information of the meetings between the mobile devices. More recently, [4] proposed an expected encounter based routing protocol that makes the routing decision by comparing the minimum expected meeting delay to the destination. Besides, they proposed a community aware routing protocol using the expected number of encountering communities. The paper studies how the failures of some nodes in opportunistic networks can affect the performance of social-based forwarding strategies. These nodes can fail due to energy exhaustion, or intermittent connectivity where they can be out of communication range. It was shown that the non participation of only some important nodes can significantly degrade the performance of the entire network. The authors concludes that the community-based forwarding and routing methods in DTNs are really sensitive to the change of network communities.

In this paper, we have also shown how to detect social structures from real mobility traces. However, our approach uses the mobility behavior and the user geographical preference for community identification and to decide how to forward the data.

VII. Conclusions

In this paper we introduced DACCOR protocol. Thus, we hypothesize that users that have similar geographical preferences have also similar interests and as such we used a deep autoencoder to pre-process raw mobility datasets. This autoencoder approach was able to more accurately uncover community structures which identifies groups of users sharing common geographical interests and temporal relationships.

The proposed protocol used the community information and user relationship to make an efficient next hop selection decisions. Through extensive experimentation using one synthetic and two real mobility records representing diverse urban mobility scenarios we show the effectiveness of the proposed opportunistic protocol.

Our results show that the proposed deep autoencoder community based routing protocol lead to an improvement of the performance of the studied network metrics, i.e. delivery probability, overhead ratio, hop count, latency and dropped messages when compared with Epidemic and Prophet routing protocols. Finally, we show that DACCOR is able to outperform other opportunistic forwarding protocols, not only on the networks metrics, but also in the fact that, by using less bandwidth and less radio, DACCOR is able to dramatically decrease energy consumption, optimizing battery life.

VIII. Acknowledgments

This work was partially supported by CAPES (Ministry of Education, Brazil) and by NSF under project CNS 1321151.

References

[1] M. Alajeely, R. Doss, and A. Ahmad, “Routing protocols in opportunistic networks: A survey,” *IETE Technical Review*, vol. 0, pp. 1–19, 2017.

[2] Y. Bengio, A. C. Courville, and P. Vincent, “Unsupervised feature learning and deep learning: A review and new perspectives,” *CoRR, abs/1206.5538*, vol. 1, p. 2012, 2012.

[3] D. Charte, F. Charte, S. GarcÃa, M. J. del Jesus, and F. Herrera, “A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines,” *Information Fusion*, vol. 44, pp. 78 – 96, 2018.

[4] H. Chen and W. Lou, “Contact expectation based routing for delay tolerant networks,” *Ad Hoc Networks*, vol. 36, pp. 244–257, 2016.

[5] M. Chuah and A. Coman, “Identifying connectors and communities: Understanding their impacts on the performance of a dtn publish/subscribe system,” in *IEEE CSE*, vol. 4, 2009, pp. 1093–1098.

[6] M. Conti and S. Giordano, “Mobile ad hoc networking: milestones, challenges, and new research directions,” *IEEE Communications Magazine*, vol. 52, no. 1, pp. 85–96, 2014.

[7] N. Eagle and A. (Sandy) Pentland, “Reality mining: Sensing complex social systems,” *Personal Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, Mar. 2006.

[8] W. Gao, Q. Li, and G. Cao, “Forwarding redundancy in opportunistic mobile networks: Investigation and elimination,” in *IEEE INFOCOM*, 2014, pp. 2301–2309.

[9] P. Hui, J. Crowcroft, and E. Yoneki, “Bubble rap: Social-based forwarding in delay tolerant networks,” in *Proceedings of the 9th ACM MobiHoc '08*, Hong Kong, China, 2008, pp. 241–250.

[10] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, “Distributed community detection in delay tolerant networks,” in *2Nd ACM/IEEE MobiArch'07*, Kyoto, Japan, 2007, pp. 7:1–7:8.

[11] A. Keränen, J. Ott, and T. Kärkkäinen, “The one simulator for dtn protocol evaluation,” in *2nd Simutools'09*, 2009, pp. 55:1–55:10.

[12] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436 EP –, 2015.

[13] F. Li and J. Wu, “Localcom: A community-based epidemic forwarding scheme in disruption-tolerant networks,” in *6th IEEE SECON'09.*, June 2009, pp. 1–9.

[14] A. Lindgren, A. Doria, and O. Schelén, “Probabilistic routing in intermittently connected networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[15] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, 12 2016.

[16] P. D. McNicholas, “Model-based clustering,” *Journal of Classification*, vol. 33, no. 3, pp. 331–373, 2016.

[17] M. Motani, V. Srinivasan, and P. S. Nuggehalli, “Peoplenet: Engineering a wireless virtual social network,” in *Proc. of the 11th MobiCom'05*, Cologne, Germany, 2005, pp. 243–257.

[18] H. A. Nguyen and S. Giordano, “Context information prediction for social-based routing in opportunistic networks,” *Ad Hoc Networks*, vol. 10, no. 8, pp. 1557 – 1569, 2012.

[19] S. Phithakkitnukoon, Z. Smoreda, and P. Olivier, “Socio-geography of human mobility: a study using longitudinal mobile phone data,” *PLoS ONE*, vol. 7, no. 6, p. e39253, June 2012.

[20] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, “CRAWDAD data set epfl/mobility (v. 2009-02-24),” Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, Feb. 2009.

[21] V. Sze, Y. Chen, T. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec 2017.

[22] A. Vahdat, D. Becker, *et al.*, “Epidemic routing for partially connected ad hoc networks,” 2000.

[23] N. G. Vangelis Angelakis and D. Yuan, “Probabilistic routing in opportunistic ad hoc networks,” *Wireless Ad-Hoc Networks*, 2012.

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[25] F. Xia, Q. Yang, J. Li, J. Cao, L. Liu, and A. M. Ahmed, “Data dissemination using interest-tree in socially aware networking,” *Computer Networks*, vol. 91, pp. 495–507, 2015.

[26] P. Yuan, P. Liu, and S. Tang, “Exploiting partial centrality of nodes for data forwarding in mobile opportunistic networks,” in *17th IEEE CSE*, 2014, pp. 1435–1442.

[27] P. Yuan, L. Fan, P. Liu, and S. Tang, “Recent progress in routing protocols of mobile opportunistic networks: A clear taxonomy, analysis and evaluation,” *Journal of Network and Computer Applications*, vol. 62, pp. 163 – 170, 2016.

[28] Y. Zheng, X. Xie, and W.-Y. Ma, “Geolife: A collaborative social networking service among user, location and trajectory,” *IEEE Data(base) Engineering Bulletin*, June 2010. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=131038