Exploiting Friendship Relations for Efficient Routing in Mobile Social Networks

Eyuphan Bulut, *Member*, *IEEE*, and Boleslaw K. Szymanski, *Fellow*, *IEEE*

Abstract—Routing in delay tolerant networks is a challenging problem due to the intermittent connectivity between nodes resulting in the frequent absence of end-to-end path for any source-destination pair at any given time. Recently, this problem has attracted a great deal of interest and several approaches have been proposed. Since Mobile Social Networks (MSNs) are increasingly popular type of Delay Tolerant Networks (DTNs), making accurate analysis of social network properties of these networks is essential for designing efficient routing protocols. In this paper, we introduce a new metric that detects the quality of friendships between nodes accurately. Utilizing this metric, we define the community of each node as the set of nodes having close friendship relations with this node either directly or indirectly. We also present *Friendship-Based Routing* in which periodically differentiated friendship relations are used in forwarding of messages. Extensive simulations on both real and synthetic traces show that the introduced algorithm is more efficient than the existing algorithms.

Index Terms—Delay tolerant networks, mobile social networks, routing, efficiency

1 Introduction

Delay Tolerant Networks (DTNs) [1] [2] are a class of wireless networks in which a stable path from source to destination is unlikely to exist at any time instance, thus, long and variable delays occur in routing of messages. These networks are usually sparse and the connection between their nodes changes frequently. Among many real life examples of DTNs, Mobile Social Networks (MSNs) are of growing significance as a result of the rapid and wide spread use of various personal wireless devices (e.g., cell phones, GPS devices) among people and their surroundings.

In mobile social networks, there is a potential of collaborative data gathering via already deployed and human maintained devices. Therefore, opportunistic routing of messages in these networks has been studied by many researchers. However, due to the challenging network environment (intermittent connectivity causing lack of stable end-to-end path between nodes) in these networks, efficient routing of messages is not an easy task. To ease these difficulties and enable nodes to make right forwarding decisions while routing messages, inherent social network properties of these networks need to be utilized. The direct connectivity (opportunity for message transfers) between human-carried devices is enabled when they get into each other's range. Thus, the relationship defining the frequency and duration of the connectivity between nodes has to be analyzed to route messages efficiently. For example, consider a high school network. A student has a higher chance to see students in the same class (and therefore higher chance to transfer data to them) than the students from other classes that this student can meet only during breaks.

In this paper, exploiting friendship-based social network features of an MSN, we present a new routing algorithm: *Friendship-Based Routing*. To analyze social relations between nodes (i.e., people), we need to define their *friendships* in terms of their behaviors. For this purpose, we introduce new metric measuring different aspects of friendship behavior from the encounter history of nodes. Utilizing this metric, we find both the direct and indirect close friends of each node. We handle the indirect relations between nodes in the context of routing in a way that differs from previous approaches. Moreover, we also take into account the periodicity of friendship relations and propose to use different friendship communities in different time periods of a day as a better way of handling periodic variation of relations than what previous solutions proposed.

The rest of the paper is organized as follows: in Section 2, we present an overview of previous work. In Section 3, we give the detailed design of proposed routing algorithm. In Section 4, we present our simulation model and its results. In Section 5, we discuss the presented algorithm and our future work. Finally, we offer conclusions in Section 6.

2 Related Work

In this section, we first overview the state-of-the-art in general DTN routing algorithms and then specifically look at the social-based routing that utilize the social network features in their designs and are proposed mainly for MSNs.

2.1 General DTN Routing Algorithms

Recently, several DTN routing algorithms have been proposed based on different techniques (multicopy based [3], [4], [5], [6], single-copy based [7], [8], [9], [10], erasure coding based [11], [12], [13], [14], [15]).

• *The authors are with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180.*
E-mail: {bulute, szymansk}@cs.rpi.edu.

Manuscript received 22 Aug. 2011; accepted 17 Feb. 2012; published online 2 Mar. 2012.
Recommended for acceptance by P. Santi.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-08-0556.
Digital Object Identifier no. 10.1109/TPDS.2012.83.

The pioneering algorithm in the category of multicopy-based routing algorithms is Epidemic Routing [3] in which whenever two nodes are in contact with each other, they exchange their messages so that they both have the same list of message copies. As the result, the fastest spread of copies is achieved yielding the shortest delivery time and the minimum delay. The major drawback of this approach is excessive usage of bandwidth, buffer space and energy due to the greedy spreading of copies. Therefore, several algorithms were proposed (e.g., Controlled Flooding [4], Spray and Wait [5], Multiperiod Spray and Wait [6]) to limit the distribution of the message copies while still achieving high delivery rates.

One of the first studies in the category of single-copy-based algorithms is Prophet [7]. The approach is based on the observation that the movement of nodes in a typical mobile ad hoc network might be predictable based on repeating behavioral patterns (i.e., if a node has visited a location several times before, it is likely that it will visit that location again). Accordingly, Lindgren et al. propose a probabilistic routing model where the forwarding decisions are made by considering the predicted future delivery probabilities (that are computed from previous node encounters and updated with aging and transitivity mechanisms) of meeting nodes. Following the same forwarding idea, several algorithms, mainly differing from each other in terms of delivery probability computation, are proposed. For example, in [9] and [30], the time passed since the last encounter of nodes with the destination is utilized and the messages are forwarded toward nodes with recent meetings with the destination. Moreover, in MaxProp [10] prioritization of the schedule of packets that will be transmitted to other nodes or that will be dropped from the buffer (due to overflow) is also taken into account in the routing decisions, thus better performance results are achieved when the nodes have limited resources (e.g., buffer, bandwidth).

In erasure-coding-based routing, the source converts its message into a large set of blocks such that the original message can be constructed from a subset of these blocks. Then, these blocks are distributed to the different nodes in the network and delivery of at least some minimum number of them to the destination is equivalent to delivery of the entire message. The idea is first used in [11], where Wang et al. present its advantages (i.e., robustness to failures) over multicopy-based routing. Then, in some subsequent studies, its different variants are presented. In [12] and [13], optimal splitting of erasure coded blocks over multiple delivery paths (contact nodes) and multiple time periods to optimize the probability of successful message delivery is studied. A similar approach focusing on nonuniform distribution of encoded blocks among the nodes is also presented in [14]. Moreover, in [15], even a hybrid routing algorithm combining the strengths of multicopy-based and erasure coding-based approaches is proposed. In addition to encoding each message into a large number of small blocks, the algorithm also replicates these blocks to increase the delivery rate.

2.2 Social-Based Routing for MSNs

Besides the above studies, most of which assume that nodes move according to simplistic random mobility models, many recent studies have focused on MSNs (special type of DTNs consisting of human-carried devices) and analyzed the social network properties of these networks to assist the design of efficient routing algorithms. Even though the algorithms presented in these studies could fit into the categories listed in previous section, below we will list them separately since, similar to the proposed algorithm in this paper, they all use social network features in their designs.

In [16], Daly and Haahr use both social similarity (to detect nodes that are part of the same community) and egocentric betweenness metric (to identify nodes bridging different communities) to increase routing performance. When two nodes encounter each other, they calculate the joint utility function comprised of these two metrics for each of their destinations. Then, the node having higher utility for the message's destination is given the message.

In [17], each node is assumed to have two rankings: global and local. While the former denotes the popularity (i.e., connectivity) of the node in the entire society, the latter denotes its popularity within its own community. Messages are forwarded to nodes having higher global ranking until a node in the destination's community is found. Then, the messages are forwarded to nodes having higher local ranking within destination's community. A distinction between local community members and others is also made in [18] and the distribution of message copies is optimally balanced between these two kinds of encountered nodes. In [19], a community-based epidemic forwarding scheme is introduced. First, the community structure of the network is detected using local information of nodes. Then, the message is forwarded to each community through gateways.

Additionally, in some other studies, several interesting properties of social networks are considered. In [20], irregular deviations from the habitual activities of nodes are considered and it is shown that the worst case performance of routing can be improved by scattering multiple copies of a message in the network such that even deviant (less frequently encountered) nodes will be close to at least one of these copies. In [21], the effect of socially selfish behavior of nodes on routing is studied.

In this paper, we introduce a new routing algorithm different from all above studies. First, we define a new metric to understand social relations between nodes more accurately. Second, we propose a local community formation based on this new friendship detection metric. We use not only direct relations but also indirect ones in a different way than it was considered previously. Third, we introduce a new approach to handle periodic changes of node relations. Throughout the presentation of all these features of our design, we show in detail how they differ from the previous work.

It is also important to note that since in the design of social-based routing algorithms, community detection is an important parameter (since it defines the forwarding of messages between encountering nodes), some recent studies [22] specifically focus on the detection of communities or clusters in MSNs (or in general, in DTNs). To this end, algorithms and metrics from different areas (e.g., complex networks [23]) are also utilized. Both, strong interactions among the nodes within a community and interactions (which might be different from intracommunity

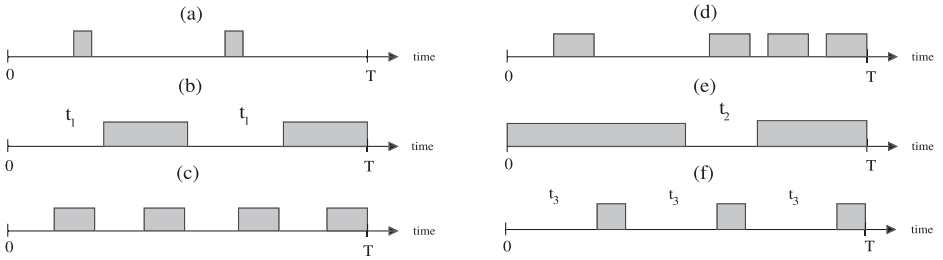


Fig. 1. Six different encounter histories between nodes i and j in the time interval [0, T]. Shaded boxes show durations of the encounters between nodes.

behavior [24]) between the nodes of different communities (through bridging nodes) are considered.

In this paper, we also use community concept in the design of the proposed routing algorithm. However, unlike others, we define communities from each node's point of view, utilizing time dependent interactions of a node with other nodes. Thus, it is possible that a node i may include another node j in its community (through indirect friendship definition), but node j might not include node i in its community. We elaborate on this issue in Section 3.2.

3 The Proposed Algorithm

3.1 Analysis of Node Relations

The intermittent connectivity between nodes in an MSN makes the routing of messages possible only in opportunistic manner. That is, message exchanges occur only when two nodes come within the range of each other and one of them assesses that it has lower delivery probability than the other. Hence, the link quality between each pair of nodes needs to be estimated accurately (from contact history) to consider the possible forwarding opportunity arising from the encounter. As a result, the periodic encounters between nodes can be condensed to a single link weight and the corresponding graph including links with weights exceeding certain threshold can be constructed.

In previous works, several metrics, including the encounter frequency, the total or average contact period and the average separation period [19], were used to extract the quality of links between pairs of nodes. However, all these metrics have some deficiencies in accurate representation of the forwarding opportunities arising from encounters between nodes. For example, consider the six different encounter histories of two nodes, i and j, in Fig. 1, where shaded boxes show encounter durations between these nodes in the time interval T. In cases a and b, the encounter frequencies are the same but the contact durations between nodes are longer in case b than in case a. Hence, encounter pattern b offers better forwarding opportunities than a does.[1] Comparing cases b and c, we notice that the contact durations are the same but the encounter frequencies are different. Since frequent encounters enable nodes to exchange messages more often, case c is preferable to case b for opportunistic forwarding.

1. It should be noted that the comparison of all configurations in terms of message exchange opportunity obviously depends on the application scenario which defines the packet size. However, without loss of generality, we assume here that the encounter durations are long enough for sending a packet.

Among the previously proposed metrics, encounter frequency fails to represent the stronger link when cases a and b are considered, and the total contact duration fails for cases b and c. Although average separation period can assign correct link weights representing the forwarding opportunity in cases a, b, and c, it fails in other cases. When we compare cases c and d, both the contact durations and the encounter frequencies are the same. However, case c is preferred to d due to the even distribution of contacts. In [19], preference of case c is achieved by utilizing irregularities in separation period as a penalty factor. However, deciding on how much it will affect the link quality in different cases is still difficult. Furthermore, for the cases such as e and f, average separation period fails to assign accurate link weights. If $t_1 = t_2$, average separation period cannot differentiate between cases b and e but case e is preferable due to its longer contact duration (average separation period can even give preference to case b if t_1 is slightly less than t_2). Similarly, if $t_1 = t_3$, average separation period cannot differentiate between cases b and f, even though case b offers better forwarding opportunity.

To find a link metric that reflects the node relations (also the forwarding opportunities) more accurately, we considered the following three behavioral features of close friendships: high frequency, longevity, and regularity. In other words, for two nodes to be considered friends of each other, they need to contact frequently and regularly in long-lasting sessions. Here, frequency refers to average intermeeting time while regularity refers to the variance of the intermeeting time. Hence, two nodes may meet infrequently but regularly (e.g., once a week) and still be considered friends. This is of course a weaker friendship than the one with both frequent and regular contacts. The previous metrics take into account some of these features but not all of them at the same time. We account for these properties in a new metric that we called *Social Pressure Metric* (SPM). It may be interpreted as a measure of a social pressure that motivates friends to meet to share their experiences. In our setting, this amounts to answering the question "what would be the limit with time unit tending to zero of the average message forwarding delay to one node (j) if the other (i) had a new message to deliver at each time unit?" (we use the limit in the definition to make the measure time unit independent). Then, we define the link quality ($w_{i,j}$) between each pair as the inverse of this value. More formally:

$$SPM_{i,j} = \frac{\int_{t=0}^{T} f(t)dt}{T} \text{ and } w_{i,j} = \frac{1}{SPM_{i,j}},$$

where $f(t)$ represents the time remaining to the next encounter of the two nodes at time t. If at time t, the nodes are in contact, then $f(t) = 0$, otherwise, $f(t) = t_{next} - t$, where t_{next} is the time of the next meeting between nodes i and j. Hence, each intermeeting time t_{inter} contributes the term $t_{inter}^2/(2T)$ to SPM. If there are n intermeeting times in the time period T, then $SPM_{i,j} = (\sum_{x=1}^{n} t_{inter,x}^2)/(2T)$ and $w_{i,j} = (2T)/(\sum_{x=1}^{n} t_{inter,x}^2)$.

The larger the value of $w_{i,j}$, the closer the friendship (the higher the forwarding opportunities) between nodes i and j. Clearly, increasing the time the nodes are in contact decreases SPM as does equalizing the time between encounters. Finally, splitting the intermeeting times into the larger number of smaller pieces also decreases the SPM. Hence, indeed, this metric combines the three desired properties of the friendship behaviors discussed above into a single measure. To illustrate the benefits of this metric, we notice that when it is used to evaluate all cases in Fig. 1, the resulting weights will accurately indicate which case offers more forwarding opportunities. It should also be noted that SPM is computed from the history of the encounters of the node. As additional node encounters happen, the corresponding SPM value is updated easily.

3.2 Friendship Community Formation

Using its encounter history, each node can compute qualities ($w_{i,j}$ values) of its links with other nodes. Then, it can define its friendship community as a set of nodes having a link quality with itself larger than a threshold (τ). This set will include only direct friends. However, two nodes that are not close friends directly still can be close indirect friends. This happens if they have a very close friend in common so that they can contact frequently through this common friend. Moreover, the relations between nodes may show periodic changes. For example, they could depend on the time of the day or the day of the week considered. Therefore, both strong indirect relationships and periodic variations of relationships must be addressed when forming friendship communities.

3.2.1 Handling Indirect Relationships

To find indirect friendships between nodes in a way relevant for routing, we propose to use *relative SPM* (or simply $RSPM$) metric. Consider the sample encounter history shown in Fig. 2 in which the upper diagram shows the contacts between nodes i and j, while the lower one shows the contacts between nodes j and k. We define $RSPM_{i,k|j}$ as the answer to the question "what would be the average delivery delay of node i's continuously generated messages if they followed the path $<i, j, k>$?" Each indirect information passing consists of two stages. The first one starts at the last meeting of node i with node j and ends at the time node i's next contact with node j ends (assuming that any message generated at node i can be transferred to node j when they are in contact). However, if there are several subsequent meetings with j before any meeting of j with k, then the last one is considered. We denote duration of this stage as $t_{a,x}$ where x denotes the number of indirect information passing occurring. During this stage, node i transfers messages to node j. The second stage starts when the first one ends and it finishes when node j meets node k.

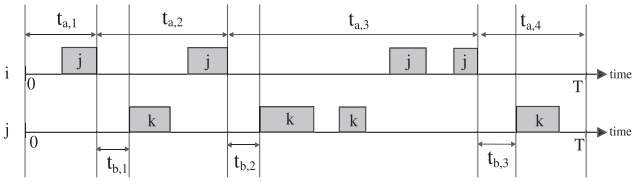


Fig. 2. Encounter history between nodes i and j (upper diagram) and between nodes j and k (lower diagram) in the same time interval [0, T].

The duration of this session is denoted $t_{b,x}$. During this stage, the messages accumulated at j merely wait for the meeting with the destination (without accumulating further at node j). Example is given in Fig. 2, in which in time T, there are three full information passing sessions between nodes i and k via node j, and the beginning of the fourth one. Denoting the number of such sessions as n, $RSPM_{i,k|j}$ is computed as:

$$RSPM_{i,k|j} = \left(\sum_{x=1}^{n} \int_0^{t_{a,x}} (t_{b,x} + t_{a,x} - t)dt\right) \Big/ T = \frac{\sum_{x=1}^{n} \left(2t_{b,x}t_{a,x} + t_{a,x}^2\right)}{2T}.$$

Since the intermediate node, j, records all of its past contact times with i and k, it can compute the value of $RSPM_{i,k|j}$.

In Algorithms 1-3, we give the details of computation of SPM and $RSPM$ values from a node's point of view. At the beginning, each node j initializes its parameters as described in Algorithm 1. Then, when a new node m is encountered, it updates the value of $SPM[m]$ and for each of its other contacts, i, it updates the value of $RSPM[i][m]$ if node m is encountered first time after its meeting with node i (Algorithm 2). When the meeting of a node with another node m ends, it also updates the value of $SPM[m]$ and sets the end time of current $t_a(m,k)$ and start time of next $t_b(m,k)$ to the current time t for each of its other contacts k (Algorithm 3).

Algorithm 1. initialize (node j)
1: **for** each $i \in N$ and $i \neq j$ **do**
2: $cur\_total1[i] = 0$
3: $t_{pre}^{end}(i) = 0$
4: **for** each $k \in N$ and $k \neq j \neq i$ **do**
5: $t_a^{start}(i,k) = 0$
6: $t_a^{end}(i,k) = 0$
7: $cur\_total2[i][k] = 0$
8: **end for**
9: **end for**

Algorithm 2. neighborDetected (node m, time t)
1: $t_{cur}^{start}(m) = t$
2: $nt = t_{cur}^{start}(m) - t_{pre}^{end}(m)$
3: $cur\_total1[m] += \frac{nt(nt+1)}{2}$
4: $SPM[m] = cur\_total[m]/2t$
5: **for** each $i \in N$ and $i \neq m$ **do**
6: $t_b^{end}(i,m) = t$
7: **if** $t_a^{end}(i,m) > t_a^{start}(i,m)$ **then**
8: $t_b(i,m) = t_b^{end}(i,m) - t_b^{start}(i,m)$
9: $t_a(i,m) = t_a^{end}(i,m) - t_a^{start}(i,m)$

10: $cur\_total2[i][m] += 2t_b(i,m)t_a(i,m) + (t_a(i,m))^2$
11: $RSPM[i][m] = cur\_total2[i][m]/2t$
12: $t_a^{start}(i,m) = t_a^{end}(i,m)$
13: **end if**
14: **end for**

Algorithm 3. neighborLeft (node m, time t)
1: $t_{pre}^{end}(m) = t$
2: $SPM[m] = cur\_total1[m]/2t$
3: **for** each $k \in N$ and $k \neq m$ **do**
4: $t_a^{end}(m,k) = t$
5: $t_b^{start}(m,k) = t$
6: **end for**

In an MSN, each node can detect its direct friendships from its own history (by computing SPM values). However, to detect indirect friendships, a node needs $RSPM$ values from its friends. Once such $RSPM$ values are received and updated at the encounter times with friends, each node can form its friendship community using the following definition:

$$F_i = \{j | w_{i,j} > \tau \text{ and } i \neq j\} \cup \{k | w_{i,j,k} > \tau \text{ and } w_{i,j} > \tau \text{ and } i \neq j \neq k\},$$

where $w_{i,j,k} = 1/RSPM_{i,k|j}$. The above equation enables nodes to detect their one-hop direct and two-hop indirect friends. Indirect friendships can also be generalized to friends more than two hops away. However, we have not included such extension because [19] demonstrated that nodes in the same community are usually at most two hops away from each other.[2]

Clearly, the introduced method for detecting the indirect strong links between nodes is different than previous approaches (based on transitivity [7], [9], [19]) which basically consider the links between node pairs separately and assume a virtual link between node i and k if $w_{i,j}w_{j,k} > \tau$. However, in our model, we can detect indirect relations more accurately. For example, if node j has a weak direct link with node k, $w_{i,j}w_{j,k}$ may be less than τ. However, if node j usually meets node k in a short time right after its meeting with node i, our metric can still identify node k as a friend of node i. This definition of indirect node relations is particularly meaningful within the context of routing because a node receives a message from one of its contacts and sends it to another contact. That is, it holds the message between its meetings with two different nodes. Hence, this metric accurately estimates the quality of indirect message exchange opportunity between two nodes.

Relativity is significant in handling indirect node relations because recent studies [34] [35] have shown that the intermeeting times between most of the node pairs fit to log-normal distribution. Thus, their future contact times might depend on their past contacts and the time passed since their last encounters (due to nonmemoryless property of the log-normal distribution). Moreover, some other studies (see [36]) point out that the mobility of many real objects are at least weakly periodic in which case also the future contact times depend on the time passed since the last encounter. Inspired by these studies, we investigated a possible correlation between the meetings of two different nodes. To this end, we computed average relative intermeeting times of a popular node with other nodes using the MIT [29] and Haggle [39] traces. We define average relative intermeeting time of node A with C relative to B, $\tau_A(C|B)$, as the average time that passes from the moment node A met B to the time it meets C.

Fig. 3 shows these results. For each data set, the upper plot shows the 3D view of $\tau_A(C|B)$ and the lower plot shows the contour plot displaying the isolines of $\tau_A(C|B)$. The diagonals and the Ath row and column in the plots are assumed to be zero since we computed $\tau_A(C|B)$ values for $A \neq B \neq C$. Moreover, if there is no instance of the case (in all traces) in which node A meets node C after it meets node B, we set $\tau_A(C|B) = -1$.

If there were no correlation between the meetings of node A with other nodes, for a given C, $\tau_A(C|B)$ would have the same value for all B's and entire rows would have been of the same color in contour plots. In contrast, we observe different colors within the rows, indicating that the meetings of node A with different nodes are not independent of each other (since there are many nodes in MIT traces and the contacts are infrequent, plots require a careful inspection to reveal that). This observation is consistent with the real-world scenarios. For example, consider the meetings of a person going from home to work every morning. After meeting family members (while leaving home), this person meets later office friends. Yet, on the way to the office, this person meets the security guard at the gate to the workplace a few moments before meeting office friends. In other words, for this person, meetings with office friends are correlated with meetings with the security guard. Consequently, in many MSNs in real life, there will be correlation between the meetings of nodes. Thanks to such correlation, the quality of indirect relationship computed from the history of encounters will be useful for routing of messages.

3.2.2 Handling Periodic Variation in Node Relations

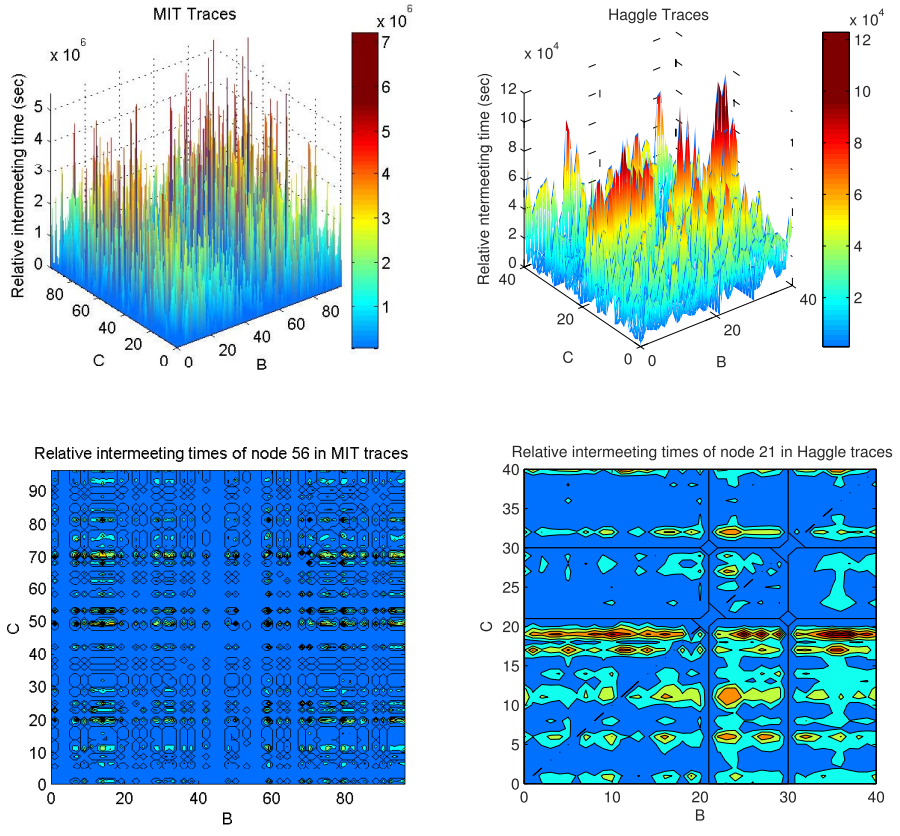
Node relations in an MSN often change with time periodically. Such periodic changes must be addressed for accurate computation of link qualities between nodes. When we analyzed two commonly used mobile social network data (MIT Reality data set [29] and Haggle data set [39], see Section 4 for details), we have observed periodic [25] variations in node relations.

In Figs. 4 and 5, we plotted the distribution of encounter times of two different nodes[3] in each data set with other nodes in their data set. Clearly, nodes encounter other nodes in some specific periods of the day. For example, in MIT traces, node 28 meets with node 38 usually between 9 a.m. to 7 p.m. while it meets with node 48 usually between 1 p.m. to 7 p.m. Similar behavior is also seen in Haggle traces.

Considering the fact that main activities of people are periodic, it is reasonable to expect similar behaviors in other MSNs. For instance, i can be a friend from school, work, or home of a node j and their encounter times then would

2. Limiting friends to two hops away does not limit routing path to any number of hops. The message still may travel several hops before reaching destination, depending on the encounters of the nodes that carry it.

3. These nodes are 28 and 56 in MIT traces, 39 and 21 in Haggle traces. We selected these nodes because they are the ones with the highest number of encounters (with other nodes).



(a) $\tau_{56}(C|B)$ in MIT traces (b) $\tau_{21}(C|B)$ in Haggle traces

Fig. 3. Relative intermeeting times of popular nodes in MIT and Haggle traces. In figures, B represents the id of the node already met (relative node) and C represents the id of the node to be met.

differ accordingly. Moreover, i can be a friend from both school and home of j in which case they stay together during the entire day.

To capture the impact of temporal changes of node relations on the link quality, previous works have proposed to use some aging mechanisms [7] [27] and time and density window-based aggregations of node relations [8] [28]. However, these mechanisms do not take into account the periodicity of node relations and react slowly to temporal changes of link quality. For example, if an aging mechanism is used, around 7 p.m., the quality of link from node 56 to node 38 (see Fig. 4) starts to decrease with aging effect[4] but still keeps a high value for some time. Yet, node 56 usually does not meet with node 38 until 10 a.m. next day. Therefore, forwarding a message considering an aged but still strong link quality may cause high delays when the link is already in its periodic low. Similarly, if a window-based aggregation (e.g., last 6 hour encounter history) is used, at 10 a.m., the link quality will be zero due to lack of contact between nodes 56 and 38 in the last window of encounters. Thus, meeting of these two nodes in the near future will be considered very improbable, while the real data indicates the contrary.

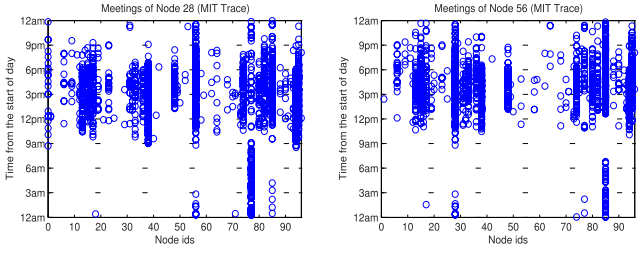


Fig. 4. Encounter distributions of node 28 and 56 in MIT traces.

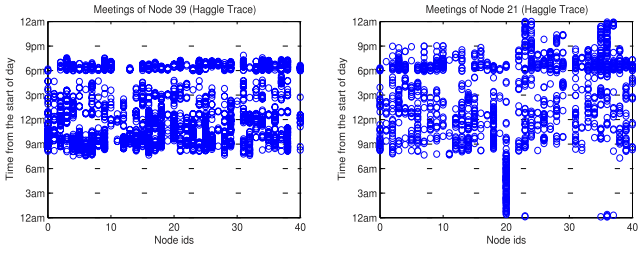


Fig. 5. Encounter distributions of node 39 and 21 in Haggle traces.

To reflect the periodic variation of the strength of friendship, we propose to use periodic friendship communities in our protocol. That is, each node i computes its F_i for different periods of the day and has different friendship communities in different periods. For example, if we divide a day into three hour periods, as shown in Fig. 4, node 85 will be the only friend of node 56 in period 3 a.m.-6 a.m., whereas nodes 28, 85, and 95 will be friends of node 56 in

4. $w_{i,j} = w_{i,j}\alpha^t$, where t is the time since the last encounter and $0 < \alpha < 1$ is aging parameter.

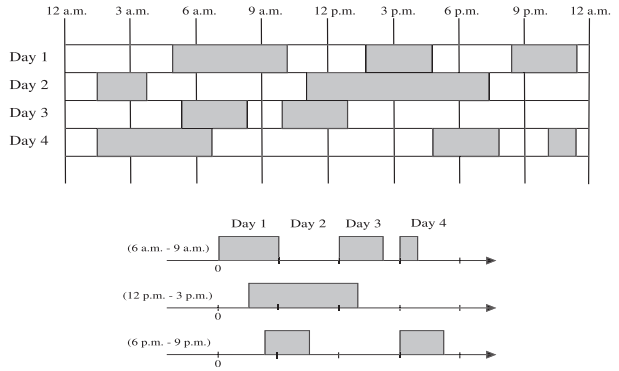


Fig. 6. Sample contact history between two nodes (upper) and the updated contact history for three different periods (lower).

period 9 p.m.-12 a.m. In Haggle traces (Fig. 5), we also observe similar situations. While nodes 23, 24, 35, and 36 are friends of node 21 in period 9 p.m.-12 a.m., node 20 is its only friend in period 12 a.m.-3 a.m. However, if an aging mechanism were used, these four nodes would have still been considered good friends of node 21 in period 12 a.m.-3 a.m. because even though the corresponding link weights are decreasing, they are still high enough to indicate friendship. In Haggle traces, for some nodes, all of the contacts may be squeezed into 9 a.m.-6 p.m. range of work hours. However, with a careful look, one can easily detect similar examples even within this time range (e.g., in Fig. 5a, node 11 is the friend of node 39 only from 9 a.m. to 1 p.m.).

To be able to compute its friendship community for each period, a node i first needs to convert the time of its encounters to the local time of each period. Consider the upper graph in Fig. 6 where a sample four day contact history between two nodes is illustrated. If three-hour ranges are used to define periods, the encounters within each specific three-hour period should be considered separately to analyze the friendship relations within the periods. Therefore, the encounter history has to be updated for each period as illustrated in the lower graph in Fig. 6. As shown, corresponding periods of the contact history in a day (global time) are concatenated to form the updated contact history of a period (local time). In Algorithm 4, we show how each period forms and maintains its own contact history as the events (neighbor entering or neighbor leaving the communication range of a node) occur in global time. Once each node generates an event handler for each of its periods with the given start and end indexes (e.g., the start and end indexes for period (6 a.m.-9 a.m) are 21,600 and 32,400 s), the local contact history for each period can be generated following the steps in Algorithm 4. First the global time of the event is converted to local time (in seconds) using the $update\_time()$ method (Algorithm 5). Then, using $stack$, the start and end times of contacts within the period are detected. Note that, as it is seen in lines 8-9 of Algorithm 4, when the event handler of a period notices that a full encounter according to the local clock of the period has finished, it calls $neighborDetected()$ (Algorithm 2) and $neighborLeft()$ (Algorithm 3) methods to compute the SPM and $RSPM$ values within the period (using encounter times according to the local clock of the period).

TABLE 1
Updated Times (in Seconds) of Encounters in Fig. 6 for Two Different Periods

Contact id	Global time of encounter (t_start, t_end)	Local time in (6 a.m.-9 a.m) period	Local time in (12 p.m.-3 p.m) period
1	(17800, 37000)	(**0**, **10800**)	(0, 0)
2	(48600, 61200)	(10800, 10800)	(**5400**, 10800)
3	(73800, 83700)	(10800, 10800)	(10800, 10800)
4	(91800, 100800)	(10800, 10800)	(10800, 10800)
5	(126000, 154800)	(21600, 21600)	(10800, 21600)
6	(191700, 203400)	(**21600**, **30600**)	(21600, 21600)
7	(207720, 219600)	(32400, 32400)	(21600, **25200**)
8	(264600, 283320)	(**32400**, **34920**)	(32400, 32400)
9	(318600, 331200)	(43200, 43200)	(43200, 43200)
10	(338400, 343800)	(43200, 43200)	(43200, 43200)

The bold values in local times show the start and end times of local encounters in corresponding period.

Algorithm 4. periodEventHandler (event e, node m, time t, time $start\_index$, time end_index)

```
1: t_upd = update_time(t, start_index, end_index)
2: if (Stack is not empty) then
3:    if (value at top of Stack ≠ t_upd) then
4:       if (Stack.size = 2) then
5:          if (e is neighbor detection) then
6:             t2 = Stack.pop()
7:             t1 = Stack.pop()
8:             neighborDetected(m, t1)
9:             neighborLeft(m, t2)
10:         else if (e is neighbor leaving) then
11:            Stack.pop()
12:         end if
13:      end if
14:      Stack.push(t_upd)
15:   else
16:      if (Stack.size = 1) then
17:         Stack.pop()
18:      end if
19:   end if
20: else
21:   Stack.push(t_upd)
22: end if
```

Algorithm 5. update_time (time t, time $start\_index$, time end_index)

```
1: d = ⌊(t − end_index)/86400⌋
2: period_length = end_index − start_index
3: if ((t < start_index + (d + 1) × 86400) and
       (t > end_index + d×86400)) then
4:       t_upd = (d + 1) × period_length
5: else
6:   t_upd = (t%86400) − start_index + (d + 1) ×
     period_length
7:   return t_upd
8: end if
```

In Table 1, we give the list of the encounter times in Fig. 6 according to both the global time (in seconds) and the local time of periods (6 a.m.-9 a.m.) and (12 p.m.-3 p.m.). The bold values in local times show the start and end times of local encounters (that trigger the run of

$neighborDetected()$ and $neighborLeft()$) in the corresponding period.

3.3 Forwarding Algorithm

Once a node constructs its friendship community for each period based on its current encounter history, it decides whether to forward a message to the encountered node using the procedure in Algorithm 6. If a node i having a message for node d meets with node j, it forwards the message to j if and only if node j's friendship community ($F_j(pid)$) in the current period pid includes[5] node d and node j is a stronger friend of node d than node i is. Accordingly, even if node j has a stronger link with node d than node i has, if node j does not include d in its current friendship community (i.e., weight of link between j and d is less than τ), node i will not forward the message to node j.

Algorithm 6. messageForward (met node j, time t, period length l)

1: // i = id of the node running the algorithm
2: pid = $\lfloor t/l \rfloor$
3: **if** $((pid+1) \times l - t) < t_b$ **then**
4: pid = pid + 1
5: **end if**
6: Request/Receive friendship quality (fq$(j,d) = \max\{w_{j,d}, w_{j,any,d}\}$) of j for each destination d of i's current messages in period pid
7: // fq(j,d) is not returned if it is less than τ and $d \in F_j(pid)$ if it is returned.
8: **for** each message m with destination d **do**
9: **if** $d \in F_j(pid)$ **then**
10: **if** fq$(j,d) >$ fq(i,d) **then**
11: Forward the message m to j
12: **end if**
13: **end if**
14: **end for**

For an efficient routing, we also need to handle period boundary cases which arise when the encounter of two nodes is close to the end of the current period. In such a case, nodes use their friendship communities in the next period. For example, if we use three hour periods for community formation and node i meets node j at 2:45 p.m., it would be better if both nodes use their communities (so the link weights) in the next three hour period (3 p.m.-6 p.m.) to check whether the destination is included. Since the time remaining in the current period is very short, using the current communities may lead to inefficient forwarding decisions. In our algorithm, we use threshold t_b and let the nodes use next period's community information if remaining time to the end of current period is less than t_b (lines 3-5 in Algorithm 6).

4 Evaluations

To compare the performance of the proposed algorithm and the existing algorithms via simulations, we have built a Java-based custom DTN simulator. It uses either the traces of real objects from real DTN environments or the synthetic traces. The network parameters (number of nodes, etc.) are defined by the traces used.

4.1 Data Sets

4.1.1 Real DTN Traces

We used the following two real DTN traces from the crawdad archive [38]:

- **MIT reality data set** [29] consists of the traces of 97 Nokia 6600 smart phones which were carried by students and staff at MIT over nine months. In our simulations, we used the contacts logged during a three month period from the beginning of February to the end of April. This is the time of the second academic semester where human relations are relatively stable and participants are active on campus [25].
- **Haggle project data set** [39] consists of many traces from different experiments. We selected the Bluetooth sightings recorded between the iMotes carried by 41 attendants of Infocom 2005 Conference held in Miami. Devices were distributed on 7 March, 2005 between lunch time and 5 p.m. and collected on 10 March, 2005 in the afternoon.

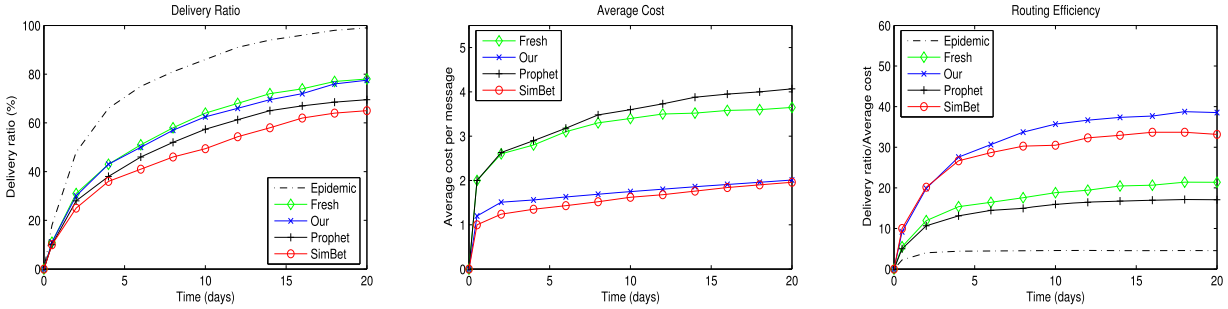
4.1.2 Synthetic Mobility Traces

Using a community-based mobility model similar to the models in [7], [31], [32], and [33], we also generated synthetic mobility traces. In a 1,000 m by 1,000 m square region (i.e., small town, campus), we generated N_c randomly located nonoverlapping community regions (home, work, buildings, etc.) of size 100 m by 100 m and we allocated N_p nodes (i.e., people) to these community regions as their home ranges. Then, for each node, we randomly assigned V different communities to visit regularly (i.e., the places visited regularly by a person in a day). Each node first selects a random point within the next community region in its list, assigns a random speed in range $[V_{min}, V_{max}]$ and moves toward the target point with that speed. When it reaches that point, it randomly assigns a visit duration in range $[T_{min}, T_{max}]$ and randomly walks within the community region for that visit duration. Once its visit ends, it moves to the next community in its list in a similar way. Each node visits all the communities in its list as indicated and comes back to its home community. Then, at the start of next day, all nodes follow again the same process and visit the communities in their list. To make the movements of nodes more realistic, we also considered irregular movements of nodes. When a node finishes its visit in one of the communities in its list, it may decide to visit a random community (other than the ones on its list) with probability p_r. After this irregular visit, the node then continues its community visits by moving to the next one on its list. All nodes have a transmission range of R. While nodes are moving, the meetings between them are recorded. The default values for the parameters are $N_c = 15$, $N_p = 100, V = 5, [V_{min}, V_{max}] = [20, 120]$ m/min, $[T_{min}, T_{max}] = [30, 120]$ min, $p_r = 0.1, R = 30$ m.

4.2 Algorithms in Comparison and Performance Metrics

Using simulations, we compare the proposed routing algorithm with three other benchmark algorithms: Prophet

5. This is equivalent to checking whether either of direct or indirect weights between nodes j and d is larger than threshold τ. Considering this usage of friendship in the design of the forwarding algorithm, each node indeed does not need to keep and maintain an explicit list of its friends. Friendship concept here is used to define possible good carriers of a message with sufficiently high link weights.



(a) Delivery ratio vs. time (b) Average cost vs. time (c) Routing Efficiency vs. time

Fig. 7. Comparison of algorithms using MIT traces.

[7], SimBet [16], and Fresh [30]. Each algorithm (including ours) uses only the contact history of nodes and deals with the forwarding of a single copy of the message to make the comparison fair. In Prophet, each node calculates its *delivery predictability* using its contact history along with transitivity and aging features and each node passes the carried packet if it meets a node with the higher predicted delivery probability. In SimBet, each node calculates a *simbet* metric using two social measures (social similarity and ego-centric betweenness) and during the meetings, the messages are forwarded to the encountered nodes with higher simbet value. Finally, in Fresh [30], a node forwards a message to the encountered node only if it had earlier meeting with the destination node than the encountered node. For Prophet and SimBet, we use the same parameters suggested in original studies [7], [16]. To show the optimal delivery ratio that could be achieved with the current setting in the network, we also present the results of epidemic routing [3].

In evaluations, we use the following three metrics: the message delivery ratio, the average cost, and the routing efficiency. The delivery ratio represents the proportion of all generated messages that are delivered to their destinations. The average cost is measured by the average number of forwards per message executed during the simulation. Finally, the routing efficiency [26] is defined as the ratio of the delivery ratio to the average cost.

4.3 Simulation Results

In the simulations, we used 1/5 of each data as warm up period during which nodes build their initial contact history. After the warm up period, we generated 5,000 messages, each from a random source node to a random destination node[6] every t s. To account for duration of experiments, we set $\mathrm{t} = 300$ s for MIT data, but for Haggle and synthetic traces, we set $\mathrm{t} = 15$ s. All messages are assigned a Time-To-Live (TTL) value representing the maximum delay requirement. To form friendship communities, we used three hour periods[7] and set $\tau = 1/80\ \mathrm{min}^{-1}$ and $t_b = 15$ min.

For main simulations, we assume that the nodes have sufficiently large buffer space to store every message they receive and the bandwidth is high enough to allow the exchange of all messages between nodes at encounter times.[8] These assumptions are reasonable in view of capabilities of today's technology and are also commonly used in previous studies [37]. Any change in the current assumptions is expected to affect the performance of the compared algorithms in the same way, since they all use one copy of the message. Moreover, following [5], we used a simplified slotted CSMA MAC model. We ran each simulation 10 times with different seeds and in each run, we collected statistics by running each algorithm on the same set of messages. All plots in figures show the averages of results obtained in such repeated runs.

In Fig. 7, we show comparison of all algorithms in terms of the three aforementioned metrics using MIT traces. Disregarding Epidemic routing, because of its unacceptable cost, our algorithm achieves the highest delivery ratio (78 percent, similar to Fresh) but it also has the minimum cost (similar to SimBet). Consequently, its efficiency is the best among all algorithms with a 16, 80, and 125 percent improvement over Simbet, Fresh, and Prophet, respectively.

In simulations with Haggle traces (Fig. 8), our algorithm delivers 82 percent of all messages with the cost similar to SimBet again. As a result, it provides 35, 210, and 450 percent improvement in the routing efficiency over SimBet, Fresh, and Prophet, respectively. From Fig. 9, we also observe the superiority of our algorithm in the results with synthetic traces. While our algorithm's delivery ratio is nearly 80 percent, Fresh's is only 60 percent while SimBet and Prophet deliver just 48 percent of all messages. Still, the average cost induced by our algorithm is only slightly higher than the cost of SimBet. Therefore, our algorithm achieves the best routing efficiency which is 33, 47, and 650 percent higher than the routing efficiencies of SimBet, Fresh, and Prophet, respectively.

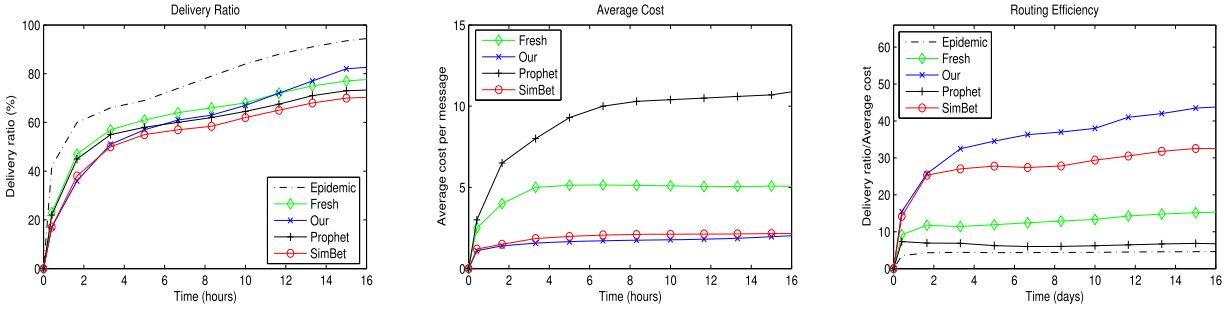
We also look at the effects of some parameters on the results. Since the effects are turned out to be similar on each data set, we only show the results with Haggle data set.

First, we look at the scenarios where the buffer space at each node is limited and FIFO buffer management scheme is used. With these assumptions, we computed the routing

6. In MIT traces, nodes that do not have any contacts with others in the selected three month period were not assigned as either source or destination to prevent meaningless messages.

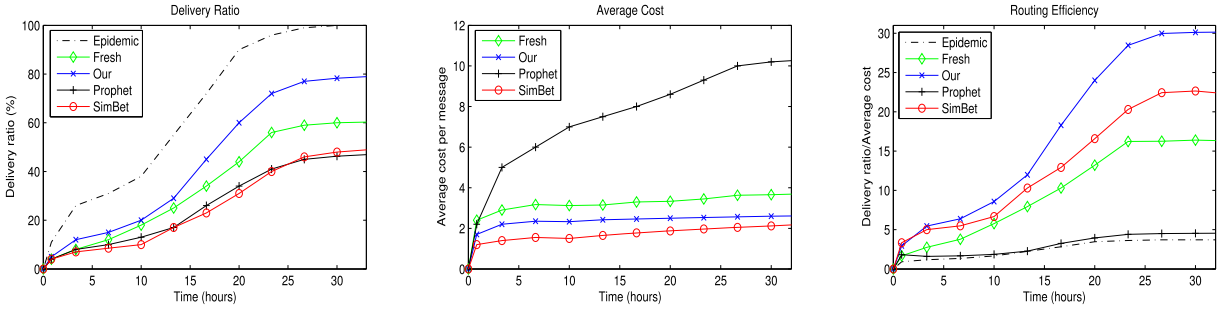
7. We selected the period length empirically considering the possible change of people's behavior in daily life and the regularity of their behavior between the boundary time points (6 to 9 a.m. and 3 to 6 p.m. may be considered as commuting times, etc.). Moreover, by finding dense regions of contacts between each pair of nodes and looking at the goodness of their fitting to the periods they are in, we also confirmed that three hour range is a good selection. In our future work, we will look at the effect of different number of periods with variable lengths on the performance of the proposed algorithm.

8. We also performed simulations with limited resources and different values of parameters. We present these results at the end of the section.



(a) Delivery ratio vs. time (b) Average cost vs. time (c) Routing Efficiency vs. time

Fig. 8. Comparison of algorithms using Haggle Project traces.



(a) Delivery ratio vs. time (b) Average cost vs. time (c) Routing Efficiency vs. time

Fig. 9. Comparison of algorithms using Synthetic traces.

efficiency achieved in all algorithms with different buffer sizes in the range of [10-50] messages. Fig. 10 shows the results. For these simulations, we set the message generation interval $t = 12$ s and TTL $= 16.6$ hours. The results show that routing efficiency of all algorithms increases as buffer space increases because messages are not dropped. Moreover, the routing efficiency of each algorithm converges to some constant value after sufficiently large buffer space is allocated. In Fig. 11, we show the routing efficiency of all algorithms with different message generation intervals (when the buffer size is 50 messages and TTL $= 16.6$ hours). The results are similar to Fig. 10, because as fewer messages are generated, fewer messages are dropped due to buffer overflows, thus more messages could be delivered, increasing the routing efficiency.

The simulation results with different traces having different number of nodes, contact frequencies, and durations, and also the results with different values of simulation parameters (buffer, message generation interval) show that our algorithm performs better than the other algorithms over wide range of environments.

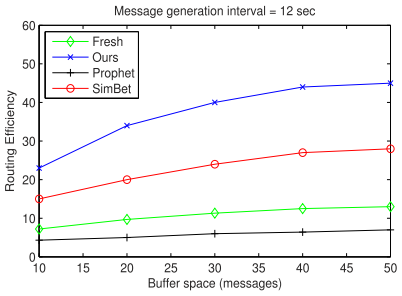


Fig. 10. Routing efficiency versus buffer space.

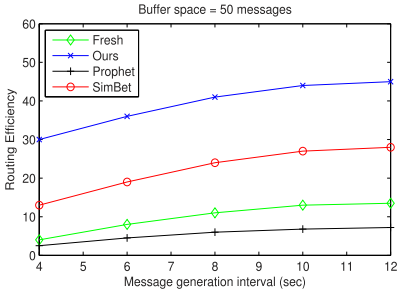


Fig. 11. Routing efficiency versus message generation interval.

5 Discussions and Future Work

5.1 Complexity of the Algorithm

In the introduced algorithm, each node determines its friendship community in each period using mainly its own encounter history without much control message overhead. The only information that a node needs from its contacts is their RSPM (or $w_{i,j,k}$) values with its noncontact nodes (needed to find indirect close friends). However, this information is requested only from the close friends[9] of nodes and performed with messages of small size compared to data messages. In contrast, the compared algorithms impose a significant control message overhead caused by exchange of the summary vectors during contact times.

9. We confirmed by inspecting real traces that not all nodes meet with each other over the entire data sets. Moreover, the average number of direct friends (nodes with sufficiently high link weights) is usually much smaller than the total number of nodes in the network. By the condition defining indirect friendship ($1/RSPM_{i,k|j} > \tau$), each indirect friend (k) is associated with a direct friend (so, j is also a friend)). Hence, the overhead is O(AC), where A is the average friend count and C is the period count.

5.2 The Effects of Number of Periods and Thresholds

Increasing the number of periods into which a day is divided (thus, the number of local friendship communities that each node has) may enable nodes to have more accurate information about the quality of their friendship relations with other nodes. Thus, when two nodes encounter, the forwarding decision could be made using the nodes' more local and accurate friendship quality with destination. On the other hand, this might degrade the delivery performance. This is because the friendship quality (i.e., delivery metric) of the node, to whom the message is forwarded, might be very local at the time of message exchange and it might become obsolete in the next period which may arrive very quickly (due to high number of periods with short periods). Moreover, the cost of computing the friendship communities (or link weights) in each period and also the space required to hold different communities will increase as well. However, as long as this cost could be handled and there is enough space at the nodes to hold the relevant data, better results could be achieved with a reasonable number of periods with proper lengths. Clearly, the threshold used is another parameter that might affect the performance of the proposed algorithm. As τ increases (decreases), friend lists of nodes get smaller (bigger), and as t_b changes, the forwarding algorithm may become more sensitive to period boundaries. In future work, we will look at these issues and try to find optimum values of τ and t_b as well as the optimal placement of period boundaries.

5.3 Extension of the Algorithm

We believe that the performance of the proposed algorithm can be improved by using the transitive friendship behavior of different nodes in the consecutive periods of the day. For example, assume that node i has a close friend j in period 12 p.m.-3 p.m., and node j has a close friend k in 3 p.m.-6 p.m. period. Then, when node s meets node i and has a message destined to node k in period 12 p.m.-3 p.m., it can forward this message to node i (even though node i has no direct or indirect close friendship with node k in the current period). This is because with high probability the message will be forwarded from i to j and then from j to k. However, such a solution will increase the algorithm's maintenance cost. We will study this issue in our future work and analyze the cost-benefit tradeoff.

6 Conclusion

In this paper, we study the routing problem in mobile social networks, which are a type of DTNs. First, we introduce a new metric to detect friendship-based node relations accurately. Then, we present a new routing algorithm in which a node forwards its messages to those nodes that contain the destination node in their friendship communities. To reflect the periodic changes on node relations, our friendship communities depend on the period of day in which forwarding is done. We also treat indirect relations between nodes in a novel way making them amenable to routing. We evaluate the proposed algorithm through simulations using two real DTN traces and a synthetic data. The results show that our algorithm performs better than three benchmark algorithms proposed previously.

Acknowledgments

Research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. Research was also sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," *Proc. ACM 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS),* 2002.

[2] J. Ott and D. Kutscher, "A Disconnection-Tolerant Transport for Drive-Thru Internet Environments," *Proc. IEEE INFOCOM,* 2005.

[3] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Technical Report CS-200006, Duke Univ., 2000.

[4] K. Harras, K. Almeroth, and E. Belding-Royer, "Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding Schemes in Sparse Mobile Networks," *Proc. Fourth IFIP-TC6 Int'l Conf. Networking Technologies, Services, and Protocols; Performance of Computer and Comm. Networks; Mobile and Wireless Comm. Systems,* May 2005.

[5] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-Copy Case," *IEEE/ACM Trans. Networking,* vol. 16, no. 1, pp. 77-90, Feb. 2008.

[6] E. Bulut, Z. Wang, and B. Szymanski, "Cost Effective Multi-Period Spraying for Routing in Delay Tolerant Networks," *IEEE/ACM Trans. Networking,* vol. 18, no. 5, pp. 1530-1543, Oct. 2010.

[7] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE Mobile Computing and Comm. Rev.,* vol. 7, no. 3, pp. 19-20, 2003.

[8] E.P.C. Jones, L. Li, and P.A.S. Ward, "Practical Routing in Delay Tolerant Networks," *Proc. ACM SIGCOMM Workshop Delay Tolerant Networking (WDTN),* 2005.

[9] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Single-Copy Case," *IEEE/ACM Trans. Networking,* vol. 16, no. 1, pp. 63-76, Feb. 2008.

[10] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," *Proc. IEEE INFOCOM,* Apr. 2006.

[11] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-Coding Based Routing for Opportunistic Networks," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking,* 2005.

[12] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using Redundancy to Cope with Failures in a Delay Tolerant Network," *Proc. ACM SIGCOMM,* 2005.

[13] E. Bulut, Z. Wang, and B. Szymanski, "Cost Efficient Erasure Coding Based Routing in Delay Tolerant Networks," *Proc. IEEE Int'l Conf. Comm. (ICC),* 2010.

[14] Y. Liao, K. Tan, Z. Zhang, and L. Gao, "Estimation Based Erasure Coding Routing in Delay Tolerant Networks," *Proc. Int'l Conf. Wireless Comm. and Mobile Computing,* July 2006.

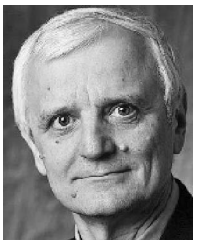
[15] L. Chen, C. Yu, T. Sun, Y. Chen, and H. Chu, "A Hybrid Routing Approach for Opportunistic Networks," *Proc. ACM SIGCOMM,* pp. 213-220, Sept. 2006.

[16] E. Daly and M. Haahr, "Social Network Analysis for Routing in Disconnected Delay-Tolerant Manets," *Proc. ACM MobiHoc,* 2007.

[17] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social Based Forwarding in Delay Tolerant Networks," *Proc. ACM MobiHoc,* 2008.

[18] E. Bulut, Z. Wang, and B.K. Szymanski, "Impact of Social Networks in Delay Tolerant Routing," *Proc. IEEE GLOBECOM,* 2009.
[19] F. Li and J. Wu, "LocalCom: A Community-Based Epidemic Forwarding Scheme in Disruption-Tolerant Networks," *Proc. IEEE Comm. Soc. Sixth Ann. Conf. Sensor, Mesh, and Ad Hoc Comm. and Networks (SECON),* pp. 574-582, 2009.
[20] T. Zhou, R.R. Choudhury, and K. Chakrabarty, "Diverse Routing: Exploiting Social Behavior for Routing in Delay-Tolerant Networks," *Proc. Conf. Computational Science and Eng.,* 2009.
[21] Q. Li, S. Zhu, and G. Cao, "Routing in Selfish Delay Tolerant Networks," *Proc. IEEE INFOCOM,* 2010.
[22] P. Hui and E. Yoneki, "Distributed Community Detection in Delay Tolerant Networks," *Proc. Second ACM/IEEE Int'l Workshop Mobility in the Evolving Internet Architecture (MobiArch),* p. 18, 2007.
[23] M.E.J. Newman, "The Structure and Function of Complex Networks," *SIAM Rev.,* vol. 45, pp. 167-256, Mar. 2003.
[24] T. Hossmann, T. Spyropoulos, and F. Legendre, "Putting Contacts into Context: Mobility Modeling Beyond Inter-Contact Times," *Proc. ACM MobiHoc,* 2011.
[25] P. Hui and J. Crowcroft, "Predictability of Human Mobility and Its Impact on Forwarding," *Proc. Third Int'l Conf. Comm. and Networking in China,* 2008.
[26] J.M. Pujol, A.L. Toledo, and P. Rodriguez, "Fair Routing in Delay Tolerant Networks," *Proc. IEEE INFOCOM,* 2009.
[27] J. Link, N. Viol, A. Goliath, and K. Wehrle, "SimBetAge: Utilizing Temporal Changes in Social Networks for Pocket Switched Networks," *Proc. ACM Workshop User-Provided Networking,* 2009.
[28] T. Hossmann, T. Spyropoulos, and F. Legendre, "Know Thy Neighbor: Towards Optimal Mapping of Contacts to Social Graphs for DTN Routing," *Proc. IEEE INFOCOM,* 2010.
[29] N. Eagle, A. Pentland, and D. Lazer, "Inferring Social Network Structure Using Mobile Phone Data," *Proc. Nat'l Academy of Sciences of USA,* vol. 106, no. 36, pp. 15274-15278, 2009.
[30] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli, "Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages," *Proc. ACM MobiHoc,* 2003.
[31] C. Chen and Z. Chen, "Exploiting Contact Spatial Dependency for Opportunistic Message Forwarding," *IEEE Trans. Mobile Computing,* vol. 8, no. 10, pp. 1397-1411, Oct. 2009.
[32] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Performance Analysis of Mobility-Assisted Routing," *Proc. MobiHoc,* 2006.
[33] M. Musolesi and C. Mascolo, "A Community Based Mobility Model for Ad Hoc Network Research," *Proc. Second Int'l Workshop Multi-Hop Ad Hoc Networks: From Theory To Reality (ACM REALMAN),* 2006.
[34] S. Srinivasa and S. Krishnamurthy, "CREST: An Opportunistic Forwarding Protocol Based on Conditional Residual Time," *Proc. IEEE Comm. Soc. Sixth Ann. Conf. Sensor, Mesh, and Ad Hoc Comm. and Networks (SECON),* 2009.
[35] P.U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. Amorim, and J. Whitbeck, "The Accordion Phenomenon: Analysis, Characterization, and Impact on DTN Routing," *Proc. IEEE INFOCOM,* 2009.
[36] C. Liu and J. Wu, "Routing in a Cyclic Mobispace," *Proc. ACM MobiHoc,* 2008.
[37] C. Liu and J. Wu, "An Optimal Probabilistically Forwarding Protocol in Delay Tolerant Networks," *Proc. MobiHoc,* 2009.
[38] CRAWDAD Data Set, http://crawdad.cs.dartmouth.edu, 2012.
[39] A European Union Funded Project in Situated and Autonomic Comm., www.haggleproject.org, 2012.

Eyuphan Bulut (M'08) received the BS, MS degrees in computer engineering from Bilkent University, Ankara, Turkey, and the PhD degree in the Computer Science Department of Rensselaer Polytechnic Institute (RPI), Troy, NY, in 2005, 2007, and May 2011, respectively. Currently, he is with Mobile Internet Technology Group (MITG) of Cisco Systems in Richardson, TX. His interests include design of protocols for wireless sensor and ad hoc networks such as routing protocols for delay-tolerant networks. He is a member of the IEEE.



Boleslaw K. Szymanski (M'82 F'99) received the PhD degree in computer science from National Academy of Sciences in Warsaw, Poland, in 1976. He is the claire and the Roland Schmitt distinguished professor of computer science and the director of the Social Cognitive Academic Research Center led by RPI. He is an author and coauthor of more than 300 publications and an editor of five books. He is also an editor-in-chief of scientific programming. His interests include parallel and distributed computing and networking. He is a fellow of the IEEE and a member of the ACM for which he was a national lecturer.