SEDUM: Exploiting Social Networks in Utility-Based Distributed Routing for DTNs

Ze Li, *Student Member*, *IEEE*, and Haiying Shen, *Member*, *IEEE*

Abstract—This work focuses on Delay Tolerant Networks (DTNs) in a social network environment. DTNs do not have a complete path from a source to a destination most of the time. Previous data routing approaches in DTNs are primarily based on either flooding or single-copy routing. However, these methods incur either high overhead due to excessive transmissions or long delays due to suboptimal choices for relay nodes. Probabilistic forwarding that forwards a message to a node with a higher delivery utility enhances single-copy routing. However, current probabilistic forwarding methods only consider node contact frequency in calculating the utility while neglecting the influence of contact duration on the throughput, though both contact frequency and contact duration reflect the node movement pattern in a social network. In this paper, we theoretically prove that considering both factors leads to higher throughput than considering only contact frequency. To fully exploit a social network for high throughput and low routing delay, we propose a Social network oriented and duration utility-based distributed multicopy routing protocol (SEDUM) for DTNs. SEDUM is distinguished by three features. First, it considers both contact frequency and duration in node movement patterns of social networks. Second, it uses multicopy routing and can discover the minimum number of copies of a message to achieve a desired routing delay. Third, it has an effective buffer management mechanism to increase throughput and decrease routing delay. Theoretical analysis and simulation results show that SEDUM provides high throughput and low routing delay compared to existing routing approaches. The results conform to our expectation that considering both contact frequency and duration for delivery utility in routing can achieve higher throughput than considering only contact frequency, especially in a highly dynamic environment with large routing messages.

Index Terms—Delay tolerant networks, social networks, utility-based routing, probabilistic routing, epidemic routing

1 Introduction

In Delay Tolerant Networks (DTNs), also known as intermittently connected networks, nodes are only intermittently connected. The intermittent connections may result from network dynamism [1], power management of mobile nodes [2], or node sparsity [3]. Examples of DTNs include mobile sensor networks [2], interplanetary communication networks [4], vehicular ad hoc networks (VANETs) [1], terrestrial wireless networks, and ocean sensor networks [5].

Routing methods specifically for DTNs have been widely studied in recent years. One group of routing methods use flooding [2], [6], [7], [8] to enable a message to opportunistically meet its destination node. Despite their high robustness and low transmission delay, flooding-based routing methods require high energy, bandwidth, and memory space that are precious resources in wireless networks. Under high-traffic loads, these methods suffer from severe resource contention and message dropping, which significantly degrade their efficiency. The other group of methods use single-copy routing, such as direct routing [9] and probabilistic (i.e., predicted) routing [3], [10], [11], [12], [13]. In direct routing, a source node spreads messages to several mobile nodes, which keep messages until they meet the destination node. In probabilistic routing, the messages are forwarded to mobile nodes that have higher probabilities of meeting the destination node as measured by the contact frequency utility. Although the single-copy methods save node resources and produce lower transmission overhead, they are likely to suffer from severe transmission delay if a suboptimal forwarding node (i.e., a node not in the shortest S-D path) is chosen.

In many DTN applications, such as mobile sensor networks [2], vehicular networks [1], and networks formed by mobile phone holders, the movements of mobile devices exhibit certain patterns in a social network because the device hosts (i.e., human or animals) normally have movement routines [11]. Some nodes, such as home neighbors and colleagues, have a high probability of meeting (meet and contact are interchangeable in this paper) with each other and staying close for a long time. This attribute of a movement pattern is called *colocation* [11] in a social network. Some nodes, such as students on a campus, meet each other with high frequency but for short periods of time. This attribute of the movement pattern is called *familiar stranger* in a social network [14]. The movement pattern of nodes can be leveraged to assist a node in finding a relay node with a high probability of successfully sending data to the destination. Familiar strangers normally have high-contact frequency, but cannot guarantee the transmission of a large number of messages during a contact due to limited contact time. On the other hand, colocation nodes may have low contact frequency, but they have a long meeting time during each contact, leading to a high transmission throughput between two nodes.

• *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29631. E-mail: {zel, shenh}@clemson.edu.*

Manuscript received 13 Mar. 2011; revised 31 Aug. 2011; accepted 22 Oct. 2011; published online 30 Nov. 2011.
Recommended for acceptance by J.C.S. Lui.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-03-0171.
Digital Object Identifier no. 10.1109/TC.2011.232.

Most current routing protocols [3], [10], [12], [15], [16], [17], [18], [19], [20], [21], [22] in DTNs only consider the contact frequency utility, which equals the number of contacts of two nodes in a time period. However, the frequency utility captures the familiar stranger attribute but does not completely capture the colocation attribute. Thus, these protocols fail to fully exploit movement patterns in the social network for higher routing throughput. We need a routing protocol that can capture both familiar stranger and colocation attributes in the social network. In this paper, we propose Social network oriented during utility-based distributed multicopy routing protocol (SEDUM) for DTNs that fully exploits node movement patterns to increase routing throughput and decrease routing delay. SEDUM consists of three distinguishing components.

- *Duration utility-based distributed routing.* We propose a *duration utility* which is the ratio of total contact duration between two nodes over a time period T. A high duration utility between two nodes indicates a high message transmission throughput between them. This utility can fully capture the colocation and familiar stranger attributes of node movement pattern in the social network. Forwarding messages to nodes that have higher duration utilities with destinations enhances routing throughput and decreases routing delay.
- *Efficient multicopy routing.* Rather than relying on flooding or single-copy routing, SEDUM uses multicopy routing to achieve a tradeoff between routing delay and overhead. It uses the optimal tree replication algorithm to enable a node to quickly replicate a number of copies to other nodes while moving. We theoretically analyze the efficiency of this replication algorithm and the influence of the replication delay on the routing delay. We also build a Markov chain to model the replication process, which helps to discover the minimum number of copies of a message to achieve a desired routing delay.
- *Effective buffer management.* The buffer management mechanism gives longer-lifetime messages a higher priority to be sent out from buffers, thus reducing the system's total transmission latency. It also gives higher utility messages higher priority to retain in a buffer when it is congested, thus increasing the system's total throughput. Further, it quickly deletes the replicas of delivered messages to release buffer congestion.

Simulation results show the higher performance of SEDUM and confirm that considering both contact frequency and duration for delivery utility in routing can achieve higher throughput than considering only contact frequency, especially in a highly dynamic environment with large routing messages. SEDUM shares similarity with some data routing methods in DTNs [15], [16], [17], [18], [19], [20], [21], [22], [23] in terms of exploiting social network in routing. However, these works focus on using contact information among nodes to form a social interaction graph to guide routing, which may not be suitable in a large network with no obvious social communities and dynamical network size changes.

The rest of this paper is structured as follows: in Section 2, we present a concise review of existing related works. Section 3 theoretically analyzes why a duration utility is better than a contact utility in enhancing throughput. Section 4 explains the SEDUM routing protocol in details. Section 5 provides a theoretical analysis of SEDUM. Simulation results are presented in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Epidemic routing (i.e., flooding) [7] is a widely used routing strategy in DTNs. This method requires that each node has a large buffer for storing messages in transmission. It can achieve a short delay by locating a shortest routing path at the cost of high-network resource consumption. There are some improved approaches proposed to reduce the overhead of epidemic routing [8], [24], [25], [26], [27]. The work in [24] uses gossip, in which a message is forwarded to partial neighbors. In [25], nodes remove redundant replicas of a message when the message has been transmitted by exchanging the "metadata" of delivered messages. The work in [8], [26], [27] uses network coding to improve the performance of epidemic routing. Although all of these methods can improve the performance of epidemic routing to a certain extent, they still inherit its shortcoming of high resource consumption.

The other widely studied routing proposal for DTNs is single-copy routing, including direct routing and probabilistic routing. Direct routing lets the source or a moving relay node carry a message all the way to the destination. Although this method can maximize the transmission capacity of the system, its transmission delay is very long. Probabilistic routing uses a variety of information to assist message routing [3], [9], [10], [11], [12], [13], [28]. In [10], [12], nodes record the history of past encounters in order to choose a node with a higher probability of encountering the destination. These methods reduce the transmission overhead of epidemic routing at the cost of possible delivery delays due to suboptimal relay node choices. Dubois-Ferriere et al. [28] pointed out that consulting the age of a node since it encountered the destination node when making forwarding decision results in superior performance over flooding. Spyropoulos et al. [9] proposed considering the estimated distance between a node and the destination when making a forwarding decision. Jain et al. [3] proposed a forwarding algorithm to minimize the average delay of message delivery using oracles that know the entire topology of the current network. However, such oracles are very difficult to implement because of the high mobility and intermittent connections between nodes. The context-aware adaptive routing (CAR) protocol [13] periodically refines the prediction of node mobility in order to identify the cluster where the destination nodes belong to and select an optimal node as a message carrier. Costa et al. [11] proposed a publish/subscribe system for DTNs. It relies on the Kalman filter [29] to predict the routing of nodes based on current topology states.

Recently, a few routing protocols have been proposed that explore social network communities in MANETs and DTNs. LABEL [15] exploits clustering algorithms to group nodes

into communities according to their mobility and forward messages between the communities. Li et al. [16] proposed to construct communities based on the neighboring relationships from node encounter histories in a distributed manner. They also proposed a locally weighted publish/subscribe method for data collection, storage, and propagation within and among the communities. Ghosh et al. [17] and Costa et al. [18] proposed using a machine learning technique to identify the communities in the network for message routing. The works in [19], [20], and [22] use social network analysis techniques to identify the social communities in the system for message routing. Gao et al. [21] focused on improving the cost effectiveness of multicast in DTNs by exploiting social centrality and social communities through social network analysis. Chen et al. [23] considered the age of the last encounter of two nodes and their cumulative contact durations as the utility for data routing. If the routing utility between two nodes is smaller than a threshold, the message is forwarded to a node with a high centrality.

All of these works focus on complex social graph analysis and community detection, while SEDUM focuses on improving the message routing performance based on node movement patterns, even if the patterns do not have community features.

Unlike most previous routing methods [3], [10], [12], [15], [17], [18], [19], [20], [21], [22] that measure delivery utility only based on node contact frequency, SEDUM also considers contact duration, which more accurately reflects the delivery utility due to intermittent connections. Although Li and Chen [16], [23] also used contact durations in routing utility calculation, they did not show the rationale to use contact duration utility and its advantages over contact frequency utility. Additionally, they focus on social community creation, which may not be suitable in a large network with no obvious social communities and dynamic network size changes. SEDUM neither assumes fixed node mobility pattern nor completely relies on community creation. It is also distinguished by its optimal tree replication mechanism and buffer management algorithm that assist message routing multicopy routing. However, SEDUM is superior to SW in two main aspects: 1) SEDUM uses probabilistic routing after a replication phase, while SW uses direct routing, and 2) SEDUM has a more effective buffer management mechanism while SW simply uses the Time To Live (TTL) strategy to manage buffers.

3 Why Duration Utility Is Better Than Frequency Utility

3.1 Frequency Utility

In DTN routing, the utility of a node is a measure of the contribution of the node to enhance a routing metric such as throughput or delay [30]. A node n_i's contact frequency utility to a node n_j is defined as the number of contacts between n_i and n_j over a time period. The contact frequency utility is widely used for probabilistic routing in DTNs. In contact frequency utility-based routing, a node chooses the neighbor with the highest utility to the destination as the next hop for routing.

3.2 Factors Affecting the Successful Transmission

We consider one message as a basic unit for the transmission between two nodes. If the link between two contacting nodes breaks before a message is completely transmitted, the message transmission fails. In a multicopy routing protocol, each copy is transmitted independently. Suppose that each message can have N_c copies and each of the copies can be successfully transmitted from the source to the destination with probability $P_{(S,D)}$. Then, the probability that at least one copy is sent to the destination node (P) is:

$$P = 1 - (1 - P_{(S,D)})^{N_c}. \tag{1}$$

Equation (1) shows that a larger $P_{(S,D)}$ and a larger N_c lead to a higher P_s. However, a larger N_c generates higher transmission overhead. Later on, we prove that increasing a large N_c leads to a linear increase in transmission overhead but a negligible delay decrease (Theorem 5.1). Therefore, we aim to increase the value of $P_{(S,D)}$. $P_{(S,D)} = \prod P_{(i,j)}$, where $P_{(i,j)}$ is the probability of successful transmission between two neighboring nodes n_i and n_j in a routing path. A large $P_{(i,j)}$ leads to a large $P_{(S,D)}$, and ultimately a large P_s. Next, we will find the factors that should be considered in order to increase $P_{(i,j)}$.

We use α to denote the smallest contact duration between two nodes at one contact. Specifically, $\alpha = \frac{R}{2v_{max}}$, where R is the transmission range of the mobile nodes and v_{max} is the maximum moving speed of a mobile node. We use f to denote the contact frequency between two nodes. Then, the two nodes have fT contacts during the time interval T. We use $P_{(i,j)}(fT = 1)$ to denote $P_{(i,j)}$ when $fT = 1$, and use $P_{(i,j)}(fT > 1)$ to denote $P_{(i,j)}$ when $fT > 1$.

Theorem 3.1. *The probability of a successful message transmission between two neighboring nodes, $P_{(i,j)}$, during a time interval T is:*

$$\begin{cases} P_{(i,j)}(fT = 1) = \left(\frac{\alpha \cdot w}{s}\right)^{\beta}, \\ P_{(i,j)}(fT > 1) = 1 - \left(1 - \left(\frac{\alpha \cdot w}{s}\right)^{\beta}\right)^{fT} (\beta > 0), \end{cases} \tag{2}$$

where w is the transmission rate of a node, s is the size of a message, and β is a constant parameter.

Proof. Chaintreau et al. [31] indicated that the communication time of one contact between two persons conforms to a power-law distribution, and given α and β, the distribution of the contact time period t can be modeled by

$$p(t) = \frac{\beta \cdot \alpha^{\beta}}{t^{\beta+1}} (0 < \alpha < t < \infty, \beta > 0). \tag{3}$$

As the amount of transmission traffic during time t is $W = wt$, Equation (3) can be transformed to [32]

$$p(W) = \frac{1}{w} \frac{\beta \cdot \alpha^{\beta}}{(\frac{W}{w})^{\beta+1}}. \tag{4}$$

Note that, for two contacting nodes, only when their communication capacity in the contact is larger than the message size ($W > s$), will the message be transmitted successfully. Therefore, based on (4), we get:

$$P_{(i,j)}(fT=1)=P_{(i,j)}(W>s)=\int_s^{\infty} p(W)\cdot dW=\left(\frac{\alpha\cdot w}{s}\right)^{\beta}, \tag{5}$$

$$P_{(i,j)}(fT>1)=1-\left(1-\left(\frac{\alpha\cdot w}{s}\right)^{\beta}\right)^{fT}.$$

□

From (6), we can see that the success probability for a message is determined by both $\alpha\cdot w$ and f. A large frequency utility cannot ensure a high transmission success probability if $\alpha\cdot w$ is very small. Also, a small frequency utility does not necessarily indicate a small transmission success probability if $\alpha\cdot w$ is very large. Therefore, frequency utility is not the only factor that affects the transmission throughput between two nodes. The frequency utility works well when the nodes in a network have a medium mobility rate (i.e., medium or large α), meaning a node can completely forward a message to the destination when they meet. However, when nodes have high mobility rates (i.e., small α), the communication time during one contact between two nodes is short. Then, it is likely that the link between two nodes breaks during the message transmission process, leading to message transmission failures.

3.3 Duration Utility

Therefore, the contact frequency utility f cannot guarantee high communication capacity and throughput of a DTN, and we need to have a new utility that can reflect both $\alpha\cdot w$ and f. Since w of a given pair of nodes is determined, to reflect α, we propose a duration utility between nodes n_i and n_j as

$$U_{(i,j)}=\left(\sum_{k=1}^{fT} t_{(i,j)}(k)\right)\Big/T, \tag{7}$$

where $t_{(i,j)}(k)$ is the encounter duration of the kth encounter.

Theorem 3.2. *A duration utility can reflect the transmission capacity between a pair of nodes with higher accuracy than a contact frequency utility.*

A large duration utility indicates either a large $a\cdot w$, a large fT, or both. Therefore, the duration utility can more accurately reflect the transmission success probability P than the contact frequency utility, especially in a DTN with high-mobility nodes and large messages.

4 Duration Utility-Based Distributed Multicopy Routing Protocol

In this section, we present the SEDUM routing protocol. We start off by describing the goals of the design of SEDUM and the strategies to achieve these goals, and briefly introduce SEDUM. Then, we present node movement models and the strategies of SEDUM in details.

Aiming to be an optimized routing protocol for DTNs, SEDUM has the following goals and corresponding strategies.

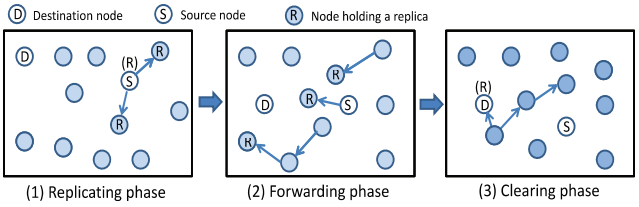


Fig. 1. An example of message routing in SEDUM.

1. To achieve a high throughput in a highly dynamic system, instead of using contact frequency as the utility in probabilistic routing, SEDUM considers both contact frequency and duration.
2. To reduce the delay of single-copy routing and reduce the overhead of epidemic routing, SEDUM employs a multicopy routing method that quickly replicates a source message to a certain number of other nodes.
3. To improve the transmission performance in a highly loaded system, SEDUM uses buffer management to effectively use node buffers.

In SEDUM, to route a message, a source node quickly spreads a number of message replicas to a number of nodes that it meets. The message replicas are transmitted simultaneously throughout the network until one copy reaches the destination node. Specifically, SEDUM can be generally divided into three phases: *Replicating phase*, *Forwarding phase*, and *Clearing phase* as shown in Fig. 1.

1. Replicating phase: every message originating at a source node is initially replicated to a number of different meeting nodes.
2. Forwarding phase: each node in the system maintains utility table recording its duration utilities to other nodes. A node always forwards a message to another node with a higher utility to the destination. This process is repeated until one copy arrives at the destination node. Nodes holding other copies will be notified about the message delivery in the clearing phase below.
3. Clearing phase: after a message transmission is completed, the destination node notifies the nodes in the system to discard the replicas of the delivered message by sending a delivered message list. The lists are exchanged between two nodes when they meet. Replica nodes that fail to delete the delivered messages still send the copies to the destinations, which will delete the duplicated received messages.

4.1 Node Movement Models

The traditional popular node movement models such as the random way-point model [33] assume that the nodes are identically and independently distributed in the system and that each node independently moves with equal frequency to every network location. Numerous recent studies on human traces (e.g., university campuses and conferences) demonstrate that these two models rarely hold true in real-life situations where mobile devices are held by humans [9]. In this case, mobile node movement is based on human decisions and social behaviors.

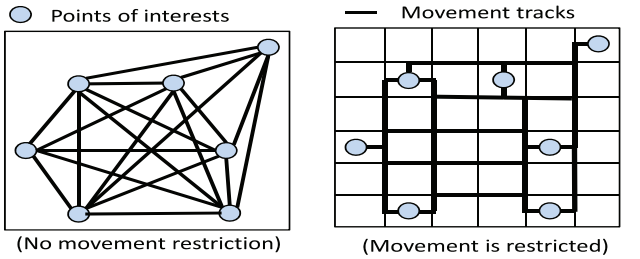


Fig. 2. Community models.

Therefore, we consider a more realistic mobility model, called the community model [31], in this work. This model captures the movement patterns of the human nodes in the social network. The community model has many communities, such as home, gathering places, and working places. In the model, every person has his/her movement routine. That is, when a person is at one place, (s)he will go to some places with high probability or go to other places with low probability. For example, if a node is at its home community, it will go to a gathering place (e.g., mall or park) with a high probability. If a node is at a gathering place, it is very likely that its next destination is home.

Fig. 2 shows a Basic community model [34] and a Manhattan community model [35]. In the former, there are no movement path restrictions on nodes. The nodes randomly select a speed and move to the destination directly. The tracks of the nodes' movements are stochastic. The Manhattan community model uses a grid road topology, in which the mobile nodes move along the grid in horizontal and vertical directions.

4.2 Duration Utility-Based Routing

In this section, we introduce a method for calculating the duration utility, which considers both contact frequency and duration between two nodes in a time interval T. Each node n_i periodically records the accumulated contact duration with the individual nodes it has met in a time period T.

Node n_i can directly send a message to n_j. Recall the direct duration utility $\widehat{U}_{(i,j)}$ between n_i and n_j in a time interval T is calculated by:

$$\widehat{U}_{(i,j)} = \left(\sum_{k=0}^{\mathcal{K}} t_{(i,j)}(k)\right) \Big/ T. \tag{8}$$

Node n_i can also send a message to n_j though n_k, i.e., $n_i \rightarrow n_k \rightarrow n_j$. In this case, we say n_i has an *indirect* duration utility $\widetilde{U}_{(i,j)}$ with node n_j. The indirect duration utility between n_i and n_j through n_k is calculated using the transitive principle:

$$\widetilde{U}_{(i,j)} = \widehat{U}_{(i,k)} * \widehat{U}_{(k,j)}. \tag{9}$$

Finally, the duration utility $U_{(i,j)}$ equals:

$$U_{(i,j)} = \max\Big(\widehat{U}_{(i,j)}, \max_{k \in N}(\widetilde{U}_{(i,j)})\Big), \tag{10}$$

where N is the set of all nodes in the network. That is, the duration utility between two nodes is the maximum of their direct utility and indirect utility. Thus, two nodes with a low contact frequency still have a high duration utility if they have a long meeting time. Even if two nodes have a low direct duration utility, if both of them have high duration utilities to a common node, they can still have a high utility to each other by forwarding messages through the common node.

Routing table

Node	Delivery utility	Relay
n_1	0.7	n_2
n_2	0.8	N/A
n_3	0.7	N/A
n_7	0.1	N/A
n_9	0.6	n_1

Fig. 3. An example of the utility table of a node.

Based on (10), a node periodically calculates its delivery utility with all other nodes. A node's movement pattern may change in a social network for some reasons, such as an office change, vocations. To make the duration utility more accurately reflect the current communication capacity, a node periodically updates the duration utility every T by taking into account both the historical utility and the current utility:

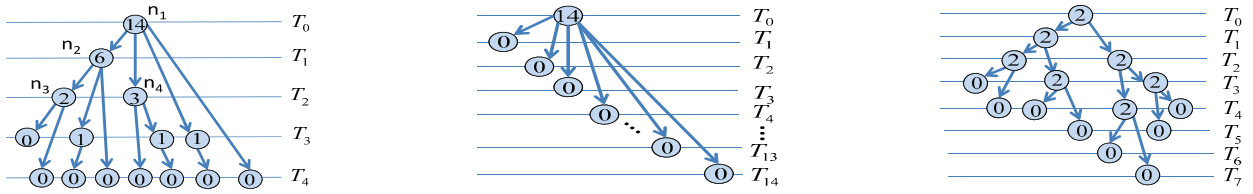
$$U_{(i,j)_{new}} = \gamma U_{(i,j)} + (1-\gamma) U_{(i,j)_{old}}, \gamma \in (0,1), \tag{11}$$

where γ is a weight constant and $U_{(i,j)_{new}}$ and $U_{(i,j)_{old}}$, respectively, denote the utility of the new and old time intervals. A system with high dynamic changes in movement pattern can set γ to a large value in order to give larger weight to newly calculated utility value to reflect the dynamic changes of the utility value.

Each node has a *utility table* to store its utilities with other nodes. Fig. 3 shows an example of the utility table of node n_i in SEDUM. It records the duration utility of n_i with all the nodes that n_i has met. The "Relay" in the table indicates whether the duration utility between n_i and n_j is a direct utility or an indirect utility. In this column, "N/A" means direct utility and node "n_k" means the utility is indirectly calculated through n_k. For example, the direct utility of n_i and n_2 is 0.8, and the indirect utility of n_i and n_1 is 0.7 calculated through n_2.

In routing, a source node or a relay node forwards a message to the neighbor that has the highest duration utility to the destination among all of its neighbors. As Fig. 3 shows, the utility of n_i to n_2 is 0.8 and the utility of n_i to n_9 is 0.6. If node n_i is asked to transmit a message to node n_2, n_i holds the message until meeting n_2 or meeting a node that has higher utility than 0.8. If node n_i is asked to forward a message to n_9, since the utility between n_i and n_9 is an indirect utility through n_1, n_i forwards the message to n_1 or a node that has a higher utility than 0.6.

Algorithm 1 shows the pseudocode for duration utility calculation. Every node n_i in the system periodically checks its connectivity. When node n_j moves into the transmission range of n_i, n_i, and n_j exchange their utility tables and update their own utility table accordingly based on (10). For example, $U_{(i,j)} = 0.4$ and $U_{(j,7)} = 0.5$. Then, $U_{(i,j)} * U_{(j,7)} = 0.2 > U_{(i,7)} = 0.1$. Therefore, n_i changes the entry of "n_7, 0.1, N/A" in its utility table to "n_7, 0.2, n_j." Also, n_i records the duration of the meeting with n_j. At each update



(a) Optimal tree replication (b) Source tree replication (c) Binary tree replication

Fig. 4. Message replication algorithms.

time period T, n_i updates the duration utilities between itself and all other nodes according to Formulas (10) and (11). We implement the utility table as a hash table. Hence, checking whether a node's utility exists in a utility table takes a constant time. Also, the utility calculation can be finished in a constant time. Therefore, an entire duration utility updating process can be finished in a constant time.

Algorithm 1: Pseudocode for duration utility calculation executed by n_i

```
//When meeting other nodes;
if meet node n_j then
  Exchange utilities that has been updated since last
  time meet with n_j
  if U_(i,j) exists in n_i's utility table then
    for each node n_k in updated utilities do
      if U_(i,j) * U_(j,k) > U_(i,k) then
        Update U_(i,k) using Formula (10)
  Record the contacting time with n_j
//Periodically update its utilities;
if currentTime=updateTime then
  updateTime+=T
  for each meeting node n_j in the last time period T do
    Calculate U_(i,j) using Formula (10)
    if U_(i,j)_old exists in its utility table then
      Update U_(i,j) using Formula (11)
```

Theorem 4.1. *SEDUM has a loop-free route from a source node to a destination node.*

Proof. Because of the movement patterns of the nodes in the network, the duration utility between each pair of nodes will converge to a stable value that can statistically reflect the communication capacity between the two nodes. Since SEDUM uses unidirectional message routing, a message is always forwarded to a node with higher delivery utility; thus, routing loop will not occur. □

4.3 Multicopy Routing

SEDUM uses a multicopy routing method to increase the probability that a message is successfully delivered to a destination node. Too many replicas will result in high-node resource consumption. Therefore, SEDUM aims to minimize the number of replicas of a message while achieving the desired routing delay. SEDUM can arrive at this minimum number based on network size, meeting interval, and desired transmission delay. We will introduce the details in Section 5.2.

There are two requirements for the replication algorithm. First, a message should be replicated quickly. Second, the algorithm can terminate the replication process after exactly N_c replicas (including the source message) are generated. To meet the requirements, SEDUM adopts the optimal tree replication algorithm [25]. In this algorithm, if node n_i is responsible for creating x replicas, when it meets node n_j, n_i sends a copy to n_j. Also, it entitles n_j to be responsible for half of its remaining responsibility; that is, it entitles n_j to replicate $\lfloor \frac{x-1}{2} \rfloor$ copies, and itself is responsible for the other $\lceil \frac{x-1}{2} \rceil$ replicas. Each replica node conducts the same operation until every node has no more responsibility.

Fig. 4a shows an example of the optimal tree replication algorithm. The number in a circle represents the number of replicas a node should create. We use epoch to denote the time step (T_i) in which a replica node replicates a message to a nonreplica node. Assume SEDUM allows each message to have $N_c = 15$ copies for message routing. Then, source node n_1 needs to create an additional $x = N_c - 1 = 14$ replicas in the network. It entitles the first meeting node n_2 to create $\lfloor \frac{N_c-1-1}{2} \rfloor = 6$ replicas at the first epoch T_1 and keeps the responsibility to create the remaining $\lceil \frac{N_c-1-1}{2} \rceil = 7$ replicas to itself. At epoch T_2, n_1 entitles the second meeting node n_4 to create $\lfloor \frac{\lceil \frac{N_c-1-1}{2} \rceil - 1}{2} \rfloor = \lfloor \frac{6}{2} \rfloor = 3$ replicas. At the same epoch, n_2 entitles its meeting node n_3 to create $\lfloor \frac{\lfloor \frac{N_c-1-1}{2} \rfloor - 1}{2} \rfloor = 2$ replicas, and itself is responsible for the remaining $\lceil \frac{\lfloor \frac{N_c-1-1}{2} \rfloor - 1}{2} \rceil$ replicas. Then, in epoch T_3, nodes n_1, n_2, n_3, and n_4 entitle their next meeting nodes with half of their own replication responsibility. The process repeats until each node completes its replicating task. The algorithm only needs $\Theta(\log_2 N_c)$ time steps and needs four steps for replicating 14 copies.

The optimal tree replication algorithm performs better than the source tree replication algorithm and the binary tree replication algorithm [25]. In the source tree replication algorithm, only a source node can replicate a message to others. As shown in Fig. 4b, the source node initially tries to create 14 replicas in the network. Since a replica is created only when the source node meets a new node, it takes $\Theta(N_c)$ epochs to create N_c replicas and needs 14 epochs to replicate 14 copies. Fig. 4c shows an example of a binary routing tree algorithm, in which each node can only replicate a message to two other nodes. It needs $\Theta(\log_2 N_c)$ epochs for replicating N_c copies, and six steps for 14 copies.

Although both the binary tree replication algorithm and optimal tree replication algorithm have $\Theta(\log_2 N_c)$ replication epochs, the former's replication process is slower than the latter on average. This is because in the optimal tree replication algorithm, the nodes entitled to replicate messages keep replicating messages until they finish their entitled replication responsibility. At this time, N_c replicas are generated in the system. In contrast, in the binary tree replication algorithm, the nodes that are entitled to replicate messages stop replicating messages after generating two

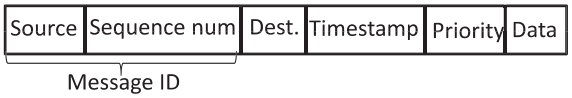


Fig. 5. The structure of the message head.

replicas. Even if the nodes meet some nodes that do not have any replicas before the entire replication processes complete, they cannot replicate messages. For example, at epoch 3, eight nodes are able to replicate the message to others in the optimal tree replication algorithm, while only four nodes can replicate messages to others in the binary tree replication algorithm. Also, the binary tree replication algorithm cannot terminate the replication process after N_c replicas are generated in the system.

4.4 Buffer Management

Because of the intermittent connections between nodes in DTNs, each node uses a buffer to store the messages needed to transmit out. When two nodes meet each other, since the communication time between two nodes is limited, the order in which different messages are transmitted affects the transmission throughput and delay of a DTN. Since the size of a buffer is limited, whether to accept an incoming message and which message to drop in order to make space for an incoming message also affects the delivery throughput and delay of a DTN. In addition, SEDUM routes multiple copies of a message in the network. If one replica is successfully delivered, a method to delete other replicas of the message in time to leave space for undelivered messages is also important. We propose a buffer management mechanism to deal with these problems in order to increase the network throughput and reduce transmission delay.

Fig. 5 shows the structure of a message head. The total size of the message head is 24 bytes. *Source* indicates the ID of the source node that generates the message. The size of the *Source* is 4 bytes. *Sequence number* indicates the sequence of the messages generated from the same source node. The size of the *Sequence number* is 4 bytes. *Destination* indicates the ID of the destination node that should receive the message. The size of the *Destination* is 4 bytes. *Timestamp* records a message's creation time with a size of 8 bytes, and *priority* indicates the priority of a message determined by the tolerable delay specified by the source node with a size of 4 bytes.

As shown in Fig. 6, each node orders the messages in its buffer according to the messages' priorities (priority 1 has higher priority than priority 2) and timestamps. The messages are ordered in descending order of their priorities. In each priority level, the messages are sorted in ascending order of their timestamps. A larger timestamp means a shorter time since a message was initiated. For example, in the group of priority 1 messages, M_{11} was created earlier than M_{73}, so M_{11} is on top of M_{73}. When two nodes, say n_i and n_j, meet each other, the messages whose destination is the other node are transmitted first. For the other messages, a node fetches a message from its buffer in a top-down manner. Based on the other node's utility table, it compares its utility and the other node's utility to the message's destination. Recall that we use U_i to denote

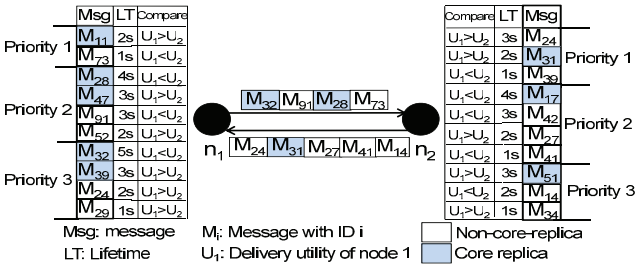


Fig. 6. An example of message exchange.

node n_i's utility to a message's destination. For each fetched message, if $U_i < U_j$, n_i forwards the message to n_j. Node n_j conducts the same operations. For example, in Fig. 6, M_{73}, M_{28}, M_{91}, and M_{32} satisfy $U_1 < U_2$, so n_1 sends these messages in sequence. Therefore, the messages with higher priorities are sent out first. Within each priority level, the messages with longer transmission delays are sent out first. The consideration of priority helps to deliver messages within their specified tolerant delay. The consideration of timestamps ensures that the longer a message stays in the network, the higher the chance that it is delivered first. This avoids having a message always stuck in a buffer, which decreases the message delivery delay of the DTN.

Next, we discuss how a node deals with an incoming message. In order to avoid losing messages due to buffer congestion, we adopt the congestion control method in [36]. That is, for a number of copies of a message, a source node initially creates a core-replica and stores it in its neighbor that was the highest delivery utility. A core-replica in a buffer cannot be replaced, but it can replace a non-core-replica in a buffer if the buffer is congested. In SEDUM, a node can be selected as a relay node by a number of nodes. Then, it needs to store a number of messages with different delivery utilities. In order to make wise use of the limited buffer resources, messages with higher utilities should have a higher priority to use the buffer. In this way, more messages can be delivered to their destinations during a certain time period. Core-replicas have a higher priority to stay in the buffer than non-core-replicas. When a node with a full buffer receives a message, if the message is a core-replica, it replaces the non-core-replica with the smallest utility in the buffer. If the incoming message is a non-core-replica with utility U_j, the node finds the non-core-replicas in its buffer whose utility is lower than U_j, then replaces the non-core-replica with the lowest utility. If all non-core-replicas's utilities are larger than U_j, the node drops the incoming message. SEDUM's buffer management method ensures that the core-replica of a message must remain in the system, thus guaranteeing successful delivery of each message. Also, it gives higher buffer priority to higher utility messages, thus enhancing system throughput.

In SEDUM, when nodes n_i and n_j meet each other, they transmit messages according to Algorithm 2. Specifically, they exchange their utility tables. According to n_j's utility table, n_i finds in its buffer the messages that would have higher utilities if residing in n_j and forwards the messages to n_j. If n_j has free space in the buffer, it accepts the incoming messages. If n_j's buffer is filled up with core-replicas, it rejects n_i's messages. If the incoming message M_i

is a non-core-replica and there is no message in the buffer with lower utilities than M_i's, n_j also rejects M_i. Otherwise, the incoming message M_i replaces the message with the lowest duration utility in the buffer. We implement the buffer as a min-heap structure [37], in which the minimum value is always at the top of the heap. Therefore, we can get the message with the minimum utility in a constant time. Since the heap structure maintenance needs time complexity $O(\log n)$ for each message insertion and deletion operation, the complexity of Algorithm 2 is $O(\log n)$ for each message transmission.

Algorithm 2: Pseudocode for message transmission from n_i to n_j.

```
//Sending messages;
Send messages with Dest.=n_j to n_j
for each message M in the buffer do
  if U_j > U_i then
    Send message M to n_j
//Receiving messages;
for each received message M with U_j do
  if its buffer is not full then
    Accept message M
  else if all messages in buffer are core-replicas then
    Reject message M
  else if message M is a non-core-replica then
    if no message has utility lower than U_j then
      Reject message M
    else then
      replace the messages with the lowest utility with message M
  else then
    replace the messages with the lowest utility with message M
```

Delivered message deletion. We define a message's ID as the concatenation of its source ID and sequence number. The replicas, especially the core-replicas, of the delivered messages need to be deleted in a reasonable time in order to free buffer space for undelivered messages. In SEDUM, every node keeps a delivered message list (*deliveredMsgList*) that records the IDs of all delivered messages. When node n_i meets node n_j, they exchange their *deliveredMsgList*. Each node then deletes the messages in its buffer indicated in the other's *deliveredMsgList* and merges this list with its own *deliveredMsgList*. A node holding a delivered message may not receive the *deliveredMsgList* of the message in time, but the node will finally receive it with high probability because of the flooding feature of the notification. A node that does not receive *deliveredMsgList* will continuously hold the delivered message copy until meeting the destination node, or the messages is replaced by other messages according to the buffer management algorithm. Even if a node does not delete a delivered message and sends it to the destination, the destination will discard the message since it has been already received. In order to restrict the size of *deliveredMsgList*, each node periodically discards outdated message IDs. Specifically, it deletes the message IDs recorded in the last time period. In order to guarantee that one of the replicas of a message is delivered in the system before the message's ID is discarded from all *deliveredMsgList*, the ID discarding period should be set as the upper bound of message transmission time in the system within which a message should be delivered.

5 Performance Analysis

5.1 Analysis of the Routing Protocol

The single-copy routing protocols generate lower overhead but lead to a longer delivery delay than the multicopy routing protocols. Epidemic routing can produce short delay in a lightly loaded transmission environment since a message is delivered to the destination along the shortest path by flooding. However, flooding consumes significant energy resources, which are precious to microdevices. Small and Haas [25] indicated that restriction of transmission traffic can save energy in the network. Multicopy routing can reach a balance between the single-copy routing and epidemic routing. Therefore, SEDUM creates N_c ($N_c \ll N$) replicas for a message to increase the probability of a message being delivered to its destination node (i.e., offloading probability) while reducing resource consumption in the network.

Theorem 5.1. *If the number of replicas per message in multicopy routing is large, adding more replicas leads to a linear increase of energy consumption but a negligible delivery delay decrease.*

Proof. Replicating $N_c - 1$ copies of a message consumes $(N_c - 1)E$ amount of energy, where E is the average energy consumption for each transmission. Suppose the average message offloading probability of each node is p. If N_c is constant over the entire lifetime of the message, the offloading delay follows a geometric distribution with mean $\frac{1}{N_c p}$. If we create one more replica, the message offloading delay is reduced by $\frac{1}{N_c p} - \frac{1}{(N_c+1)p} = \frac{1}{(N_c+1)N_c p}$. Therefore, if N_c is large, as the number of replicas of the message increases, the decrease rate of message offloading delay decreases while the energy consumption increases linearly. □

SEDUM routing consists of three phases: replicating, forwarding, and clearing. Since each of N_c relay nodes of a message looks for a routing path independently in the forwarding phase, delay in the replicating phase adversely affects the delivery delay of the message in the forwarding phase and sequentially deteriorates the whole system transmission efficiency.

Theorem 5.2. *Suppose the average total replication delay of a message is T_r. Then the average delivery delay is in the order of $O(T_r)$.*

Proof. The meeting time of two randomly selected nodes is exponentially distributed with average T_m [38]. The expected duration of the forwarding phase is $T_f = \frac{T_m}{n_c}$, where n_c is the number of generated replicas when a replica meets the destination. The average number of replicas at time t is $\frac{N_c}{T_r}t$ $(t \in [1, T_r])$. Then, the average forwarding delay is $T_f = \sum_{t=1}^{T_r} \frac{T_m}{(N_c/T_r)\cdot T_r \cdot t}$. The average delivery delay of a message is:

$$\sum_{t=1}^{T_r}\left(\frac{t}{T_r} + \frac{T_m}{N_c \cdot t}\right) = \frac{T_r + 1}{2} + \frac{T_m}{N_c} O(lnT_r) = O(T_r).$$

□

Therefore, in order to reduce the delivery delay, we need to reduce the replication delay. Thus, in the replicating phase of SEDUM, a source node should replicate a message

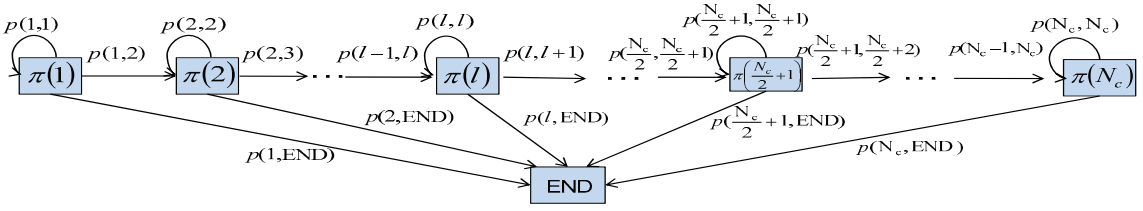


Fig. 7. A Markov chain that models a message replication process.

to its neighbors as fast as possible regardless of their utilities, instead of only replicating the message to high-utility nodes as in the forwarding phase. The initial low-utility nodes will meet high-utility nodes later on.

In Section 4.3, we explained three message replication algorithms: source tree, binary tree, and optimal tree. We compare their delay performance below.

Theorem 5.3. *The optimal tree replication algorithm can reduce the delay of the source tree replication algorithm by*

$$\sum_{i=1}^{N_c}\left(\frac{N-1}{N-i}\right)-\sum_{i=1}^{\lfloor log_2 N_c\rfloor}\left(\frac{N-1}{N-2^i+1}\right), \tag{13}$$

where N is the number of nodes in the system and N_c is the number of replicas in the system.

Proof. In the source tree replication algorithm, message replication occurs only when a source node meets a nonreplica node. Assume it has generated l replicas. The probability that it sends the $(l+1)$th replica to a new node follows a geometric distribution with mean $\frac{N-l}{N-1}$. Therefore, the average delay for replication is $\frac{N-1}{N-l}$. Since the number of epochs to replicate l replicas is $l-1$, if N_c replicas are needed to create in the system, the average delay of the creation is $\sum_{i=1}^{N_c}(\frac{N-1}{N-i})$. During the ith epoch in the optimal replication algorithm, the probability of creating a replica node equals $\frac{N-2^i+1}{N-1}$. Since the number of epochs to replicate N_c replicas is $\lfloor log_2 N_c\rfloor$, the average delay for creating N_c replicas is $\sum_{i=1}^{\lfloor log_2 N_c\rfloor}(\frac{N-1}{N-2^i+1})$. □

Theorem 5.4. *To replicate N_c replicas, the binary tree replication algorithm takes $log_2 N_c+2$ epochs, and the optimal tree replication algorithm takes $log_2 N_c$ epochs.*

Proof. In the optimal tree replication algorithm, since the nodes that are entitled to replicate messages keep on creating replicas until N_c replicas are generated in the system, the algorithm needs $log_2 N_c$ epochs. In the binary tree replication algorithm, it takes $log_2 N_c$ epochs to entitle nodes to replicate messages. After that, it takes two additional epochs for the last entitled node to replicate messages. The total number of replication epochs equals $log_2 N_c+2$. □

Theorem 5.5. *The average delivery delay $\overline{T}_d$ for the SEDUM routing protocol follows $O(\sqrt{N})>\overline{T}_d>O(\log N)$, where N is the number of nodes in the system.*

Proof. In the worst situation of the forwarding phase in SEDUM, where replica nodes cannot find a higher utility node for relaying, SEDUM becomes two-hop multicopy routing, the delay of which is $O(\sqrt{N})$ [39]. On the other hand, if replica nodes in SEDUM can always find a relay node with higher utility, SEDUM becomes optimal redundancy multihop routing [39], the delay of which is $O(\log N)$. □

5.2 Analysis of the Message Replication Process

In multicopy routing, too many replicas lead to high overhead while too few replicas may generate a long message delivery delay. The number of replicas of a message is an important issue that affects routing performance. In order to find the smallest number of replicas of a message that can guarantee a specified routing delay, we build a Markov chain to analyze the message replication process. Fig. 7 shows a Markov chain that models a message replication process. In the figure, $\pi(l)$ $(l\in[1,N_c])$ denotes the network state in which l replicas (including the original message) have been generated before any of the replicas meet the destination. END denotes the state that a replica meets the destination and the message transmission completes. The arrow in the figure indicates the state changing direction. We use $p(\pi(l),\pi(l+1))$ to denote the probability that a nonreplica node in network state $\pi(l)$ receives a replica, which changes the network state to $\pi(l+1)$.

We called the replica nodes that are entitled to replicate messages to other nonreplica nodes *entitled nodes*, and the replica nodes without replication responsibility *nonentitled nodes*. In the optimal tree replication algorithm, when $l<\frac{N_c}{2}$, every replica node is an entitled node because every replica node is entitled to replicate at least one replica. When $l>\frac{N_c}{2}$, only a portion of the replica nodes are entitled nodes. In the case of $l\in[1,\frac{N_c}{2}]$, when two nodes meet each other, because there exist l replica nodes, the probability that one node is an entitled node equals $\frac{l}{N}$, and the probability that the other node is a nonreplica node is $\frac{N-l-1}{N-1}$. The "1" in $(N-l-1)$ means the destination, and the "1" in $(N-1)$ means the first meeting node. Also, the probability of node n_i meeting node n_j is the same as the probability of n_j meeting n_i. Then, $p(\pi(l),\pi(l+1))=2\cdot\frac{l}{N}\cdot\frac{N-l-1}{N-1}$.

We calculate the number of entitled nodes when $l\in[\frac{N_c}{2}+1,N_c]$ (i.e., in the last epoch T_l). Suppose there are x entitled nodes and $l-x$ nonentitled nodes in epoch T_l. The $l-x$ nonentitled nodes in epoch T_l are separated from the $(1-x)/2$ entitled nodes in epoch T_{l-1}. Since the total number of replica nodes in epoch T_{l-1} is $N_c/2$, $x+(l-x)/2=N_c/2$. Then, in epoch T_l, the number of entitled nodes is $x=N_c-l$ and the number of nonentitled nodes is $l-x=2l-N_c$. Therefore, the probability that an entitled node meets a nonreplica node is $2\cdot\frac{N_c-l}{N}\cdot\frac{N-l-1}{N-1}$. That is, $p(\pi(l),\pi(l+1))=2\cdot\frac{N_c-l}{N}\cdot\frac{N-l-1}{N-1}$.

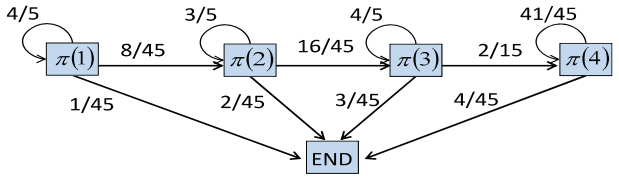


Fig. 8. An example of a constructed Markov chain.

If one of the replica nodes meets the destination node in the next epoch, the message transmission finishes. Therefore, the probability that the message can be delivered in state l in the next epoch is $P(l, \text{END}) = 2 \cdot \frac{l}{N} \cdot \frac{1}{N-1}$, where $\frac{l}{N}$ is the probability that one node is a replica node and $\frac{1}{N-1}$ is the probability that the other node is the destination when two nodes meet. In all other encountering cases, the network state stays the same. That is, $p(\pi(l), \pi(l)) = 1 - p(\pi(l), \pi(l+1)) - p(\pi(l), \text{END})$. In conclusion, the transition probability in the Markov chain between two states is

$$\begin{cases} p\big(\pi(l), \pi(l+1)\big) = 2 \cdot \dfrac{l}{N} \cdot \dfrac{N-l-1}{N-1}, l \in \left[1, \dfrac{N_c}{2}\right]; \\ p\big(\pi(l), \pi(l+1)\big) = 2 \cdot \dfrac{N_c - l}{N} \cdot \dfrac{N-l-1}{N-1}, l \in \left[\dfrac{N_c}{2}+1, N_c\right]; \\ p\big(\pi(l), \text{END}\big) = 2 \cdot \dfrac{l}{N} \cdot \dfrac{1}{N-1}; \\ p\big(\pi(l), \pi(l)\big) = 1 - p\big(\pi(l), \pi(l+1)\big) - p\big(\pi(l), \text{END}\big). \end{cases} \tag{14}$$

Suppose the average meeting interval between two nodes is T_m. We use $T_d(l)$ to denote the latency to reach network state $\pi(l)$, i.e., to replicate l replicas:

$$\begin{aligned} T_d(l) = {} & p\big(\pi(l-1), \pi(l)\big) \cdot \big(T_d(l-1) + T_m\big) \\ & + p\big(\pi(l), \pi(l)\big) \cdot \big(T_d(l) + T_m\big). \end{aligned} \tag{15}$$

When the number of replicas that a message is allowed to generate equals N_c, the average message delivery delay $T_d(END)$ is:

$$T_d(\text{END}) = \sum_{i=1}^{N_c} \big((T_d(i) + T_m) \cdot p(i, \text{END})\big). \tag{16}$$

According to Formulas (14), (15), and (16), given an average node meeting interval T_m, the number of nodes in the network N, and the number of replicas of a message N_c, the average delivery delay $T_d(\text{END})$ can be calculated. For example, when $N = 10$, $N_c = 4$, and $T_m = 1$ s, we can construct a Markov chain as shown in Fig. 8 based on (14). Then, based on (15), we can get:

$$\begin{cases} T_d(1) = \dfrac{4}{5}(T_d(1) + 1), \\ T_d(2) = \dfrac{4}{9}T_d(1) + \dfrac{35}{18}, \\ T_d(3) = \dfrac{16}{9}T_d(2) + \dfrac{52}{9}, \\ T_d(4) = \dfrac{3}{2}T_d(3) + \dfrac{47}{4}. \end{cases} \tag{17}$$

Then, based on (16), we get:

$$T_d(END) = \frac{1}{45}T_d(1) + \frac{2}{45}T_d(2) + \frac{3}{45}T_d(3) + \frac{4}{45}T_d(4). \tag{18}$$

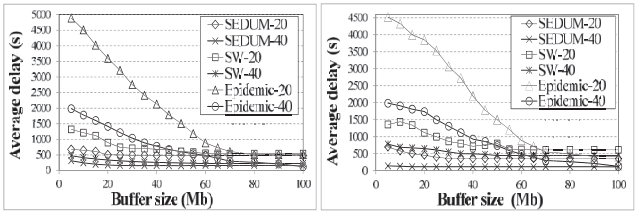
By solving (18), we retrieve $T_d(END) = \frac{34}{9}s$. Thus, given a delay tolerance $\mathcal{T}$, we can adaptively adjust the value N_c to guarantee $T_d(\text{END}) \leq \mathcal{T}$.

6 Performance Evaluation

The "THEONE" simulator [40] is an Opportunistic Networking Environment (ONE) simulator specifically designed for evaluating DTN routing and application protocols written in Java. This section demonstrates the distinguishing properties of SEDUM through simulation on "THEONE" in comparison with Epidemic routing [7] (denoted by Epidemic), and *Spray and wait* routing [6] (denoted by SW). In Epidemic, when two nodes n_i and n_j meet each other, n_i copies to n_j its messages that n_j has never received before. In SW, a source node replicates a certain number of replicas to its neighbor nodes in the system. The replica nodes buffer the replicas until meeting the destination node. We also compared the frequency utility-based SEDUM and the duration utility-based SEDUM. The experimental results confirm that the duration utility produces higher throughput and lower delay than the frequency utility.

We used two node movement models (Fig. 2): the basic community model and the Manhattan community model. In the experiments, 150 nodes are identically and independently distributed in a $2{,}000\ \text{m} \times 2{,}000\ \text{m}$ area. We assigned 15 interest points including seven home communities and eight gathering places, and randomly chose 10 nodes to share an interest point in order to show the colocation node movement pattern. The home communities and gathering places are randomly distributed in the area. Unless otherwise specified, the moving speeds of mobile nodes are randomly chosen from (0-20] m/s. Each node has a 40 m transmission range and a 40 Mb buffer size. At every second, two nodes are randomly chosen to generate a new message with a size of 1 Mb for a randomly selected destination node with a transmission rate of 2 Mb/s for 2,000 s. We assigned the same priority of tolerable delay to all messages. Initially, each node randomly chooses three points as its interested points and is assigned a probability to visit each of these three points. The probability reflects the likelihood that a node will move to the point. For moving from one interest point to another interest point, a node chooses the shortest path based on the Dijkstra shortest path algorithm. Basic community model imposes no node movement's restrictions. The Manhattan community model confines the routing paths of the mobile nodes to certain paths that reflect their real moving pattern in addition to the colocation pattern. The Manhattan model consists of grids in a matrix, in which all nodes can only move on the sides of a grid.

We set a 5-hop Time to Live for messages in the three protocols. The number of replicas for a message in SEDUM and SW was set to 8. When a node is in a home community, it will go to a gathering place with a probability of 0.8 or go to other randomly chosen places with a probability of 0.2. When a node is in a gathering place, it will then go home with a probability of 0.5 or go to other places with a



(a) Basic community model (b) Manhattan community model

Fig. 9. Delivery delay versus buffer size.

probability of 0.5. After reaching a point of interest, the node will stay there for a time period randomly selected between [10-15] s. When a node is at other places, it will go back to its home community directly.

All experimental results were averaged over 10 runs. A warm up period of 500 s is used at the beginning of the simulations to initialize the utility of SEDUM. The simulation time is 4,000 s. We use SEDUM-20 and SEDUM-40 to represent SEDUM with node transmission ranges 20 and 40 m, respectively. The same applies to SW-20, SW-40, Epidemic-20, and Epidemic-40. We are mainly interested in four metrics in the simulation:

- Message delivery delay: the average time period that it takes a message to be delivered to its destination.
- Message delivery capability: the total amount of messages that are delivered to the destinations.
- Message delivery overhead: the total amount of traffic needed to deliver the messages, including control message (e.g., delivered message list and utility table)
- Success rate: the ratio of the number of successful delivered messages to the total number of initiated messages.

6.1 Message Delivery Delay

Figs. 9a and 9b show the average message delivery delay versus the node buffer size in Basic community model and Manhattan community model, respectively. From Fig. 9, we can observe that the delivery delay for all systems decreases greatly as the node transmission range increases from 20 to 40 m. This is because a larger transmission range enables a node to contact more neighbor nodes, which increases the probability of meeting the destination or nodes with a high utilities for the destination. Both figures show that the average delivery delay decreases as the node buffer size increases. Epidemic exhibits a sharp drop, while SEDUM and SW show slight drops. A larger buffer size enables a node to buffer more messages in transmission, thus reducing the probability that a message is discarded due to buffer congestion. Because of the flooding, Epidemic produces many more message copies, which makes buffers become congested easily. Thus, in small buffers many messages are dropped whereas large buffers enable nodes to store more messages in routing and hence greatly reduce the routing delay. When the buffer size of a node is big enough to store all of the messages in a system, a source node can route a message through the shortest path to the destination by flooding messages in the network. In this situation, the delay of Epidemic is the lower bound of the network's delay.

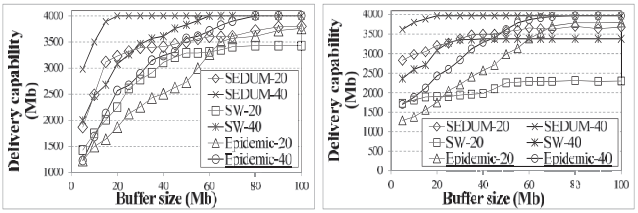
We can also see from both figures that SEDUM has the least delay most of the time, although SEDUM replicates a message to several nodes just as SW does. SW uses direct routing with $O(\frac{N}{N_c})$ delay, in which messages are routed to their destinations merely by chance. SEDUM uses probabilistic routing, in which the messages are routed to nodes with a higher probability of meeting their destinations. In addition, the buffer management mechanism in SEDUM further reduces delivery delay. SEDUM gives higher priority to longer-lifetime messages for transmission in routing. Also, it gives higher priority to higher-utility messages when a buffer is congested. Further, the multi-copy-based probabilistic routing can increase the probability that a replica is forwarded to its destination through a relatively shorter path. Thus, SEDUM leads to lower message transmission delay than SW. Relying on flooding, Epidemic suffers from severe buffer congestion because of limited buffer sizes and a tremendous number of messages. When the buffer size is small, buffer congestions lead to many message drops, which causes the high delay in Epidemic. We also see that the delay of SEDUM is almost the same as Epidemic with a large buffer size. This means SEDUM can reach the lower bound of the delay performance of the network with a large buffer size.

Comparing Figs. 9a and 9b, we find that the transmission delay of SEDUM in the Manhattan model is lower than that in the Basic model when the transmission range equals 40 m. Since the nodes can only move along the edges of the grids, the probability that two nodes which share similar movement patterns meet each other increases. By capturing the colocation and familiar stranger attributes of the node movement patterns, SEDUM's duration utility can accurately reflect the communication capacity of two nodes. Therefore, messages are forwarded to destination nodes through a number of relay nodes that share increasingly similar movement patterns with the destination.

In contrast, SW in the Manhattan community model incurs a higher delivery delay than in the Basic community model when the transmission range equals 40 m. In SW, a source node replicates a message to a number of neighbor nodes. A message is delivered to the destination only when one replica node meets the destination. However, since the nodes move along certain movement paths in the Manhattan community model, it is very likely that no replica node can meet the destination node if the replica nodes belong to different communities. Therefore, the transmission delay of SW in the Manhattan community model increases. We notice that with a 20 m transmission range, SEDUM generates similar delays in both models, so does SW. This is because when the transmission range is already small, different node movement models do not significantly change the number of nodes a node can contact. For Epidemic, the messages are flooded in the system with TTL, which is not affected by node movement patterns. Thus, its delay in both models remains approximately the same.

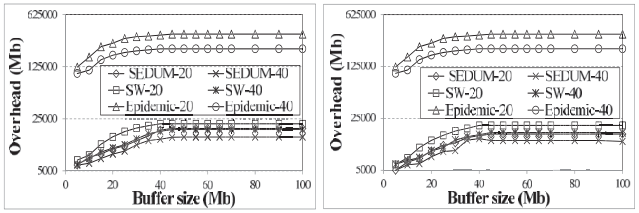
6.2 Message Delivery Capability

Fig. 10a and 10b show the total delivery throughput versus the buffer size in Basic community model and Manhattan community model, respectively. From both figures, we can see that a larger transmission range increases the throughput of SEDUM, SW, and Epidemic. A shorter node transmission range reduces the probability of node contacts, thus



(a) Basic community model (b) Manhattan community model

Fig. 10. Delivery throughput versus buffer size.



(a) Basic community model (b) Manhattan community model

Fig. 11. Delivery overhead versus buffer size.

reducing the number of successfully delivered messages. Figs. 10a and 10b also show that SEDUM produces higher throughput than SW, followed by Epidemic. SEDUM forwards messages to nodes with longer contact durations with the destination, thus enabling greater message delivery capacity. Also, it has a buffer management protocol to give higher-utility messages a higher priority to retain in a congested buffer and to give longer-lifetime messages a higher priority to be sent out from a buffer. Without the utility and buffer management strategies, SW only relies on direct routing with a TTL in the forwarding phase. Each node holds message replicas until it meets the destination node or the number of the message's forwarding hops exceeds the TTL. Messages need to stay longer in the buffer to meet their destinations as the node transmission range decreases. Therefore, more messages are dropped as their TTL expires. As a result, SW suffers more than other routing protocols from the transmission range decrease. Epidemic severely suffers from buffer congestion, especially under high load, due to flooding.

We can also see from the figures that as the buffer size increases, so does the number of messages successfully delivered to their destinations. A larger buffer size means that more messages can be buffered and the probability that a message is thrown away from a buffer decreases. Therefore, the number of messages delivered to their destination nodes is increased. The figures also demonstrate that when the buffer is large enough for most messages, the throughput of SEDUM is comparable to Epidemic in a low-load network, indicating the high throughput performance of SEDUM.

Comparing Figs. 10a and 10b, we observe that SEDUM can deliver more messages in the Manhattan community model than in the Basic community model. This is because nodes with similar movement patterns have a high probability of meeting each other in the Manhattan community model. Also, the duration utility that can capture the colocation and familiar stranger attributes of node movement patterns enables a message to travel along a path consisting of nodes with similar movement patterns as the destination. This expedites message delivery and also avoids message congestion in node buffers. Therefore, more messages can be successfully delivered to their destination nodes. In contrast, since the confined routing paths may reduce the meeting probability of a replica node with the destination node, the delivery rate of SW in the Manhattan community model decreases. In Epidemic, since a source always floods a message to the destination node, the number of received messages in the Basic community model and the Manhattan model remains nearly the same.

6.3 Message Delivery Overhead

Figs. 11a and 11b show the total overhead for message delivery in the systems versus the buffer size in the Basic community model and the Manhattan community model, respectively. We can see from the figure that as the nodes' transmission range increase from 20 to 40 m, the message delivery overhead is reduced. This is because a large transmission range increases the opportunity for a node to meet the destination node in a short time, leading to a reduced number of replicas in the system.

Fig. 11 also shows that SEDUM generates less overhead than SW, followed by Epidemic. The optimal tree replication and buffer management protocols in SEDUM enable it to deliver messages to their destinations in a much shorter time than SW and Epidemic, reducing the number of replicas created in the system. We can also see from the figures that as the buffer size increases, so does the overhead in transmission. When the buffer size is small, some of the replicas are dropped because of the congestion in the buffer. Therefore, the number of replicas transmitted in the system is reduced. However, such low overhead is at a cost of high message delivery delay and low message delivery capability as shown in Figs. 9 and 10.

Comparing Figs. 11a and 11b, we observe that SEDUM has slightly less overhead in the Manhattan community model than that in the Basic community model. This is because nodes with similar movement patterns have a high probability of meeting each other in the Manhattan community model. Therefore, fewer replicas are created in order to deliver a message, which leads to less overhead. In Epidemic, since a source always floods a message to the destination node, the generated overhead is almost the same in both models. We can also see from the figure that the overhead in SW in the Manhattan model increases slightly. It is very likely that no replica node can meet the destination node if the replica nodes belong to different communities. Therefore, the number of replicas transmitted in SW in the Manhattan community model increases.

Fig. 12 shows a breakdown of the message delivery overhead in SEDUM. Compared to the replica overhead, the control overhead in SEDUM is negligible. Although the nodes in the system need to exchange their utility tables and delivered message lists all the time, the size of the metadata in these tables and lists is only a few kilobytes. The size is so small that the control overhead imposes negligible effects on system performance. We can also see from the figure that both of the control overhead and the replication overhead in the Manhattan community model are less than those in the Basic community model. Because the message delivery in

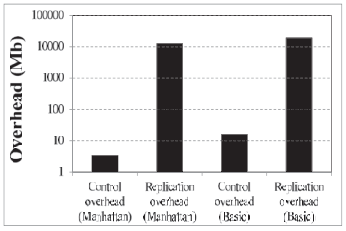


Fig. 12. Overhead in SEDUM.

the Manhattan community model is faster than that in the Basic community model as shown in Fig. 9, fewer replicas are buffered in each node, resulting in less amount of traffic for replicas and delivered message lists.

6.4 The Effect of the Number of Replicas Per Message on Delay

In this experiment, we varied the number of replicas per message from 2 to 10 increasing by 2 in each step, and measured the CDF of delivered messages versus the transmission delay in the Manhattan community model, as shown in Fig. 13. In the figure, "250+" means that the message delivery delay is larger than 250 s. We can see from the figure that as the number of replicas per message increases, more messages can be delivered in a short time. However, the delay decrease rate is not proportional to the number of replicas per message. As we can see, the message delivery delay with eight replicas per message is slightly less than that with 10 replicas per message. The result is in line with Theorem 5.1.

6.5 Comparison of Different Replication Methods

In this section, we compare the performance of SEDUM in the Manhattan community model using different replication methods: optimal tree replication, source tree replication, and binary tree replication. Table 1 shows the comparison results of the methods in terms of the average overhead, average delay, and average throughput. We see that the source tree replication method produces the highest overhead, the longest average delay and the smallest average throughput. This is because the source tree replication method is the slowest in creating the same number of replicas, which reduces the opportunity of a replica to meet the destination node. If a message needs a long time to be delivered, it needs to stay in a buffer for a long time, which may congest the buffer, leading to reduced throughput. Also, the longer that a message stays in a buffer, the higher the opportunity that the message is exchanged among nodes, which leads to a higher delivery overhead. The performance of binary tree replication is worse than the optimal tree replication and better than the source tree replication because the replication delay in the binary tree is longer than the optimal tree but less than the source tree.

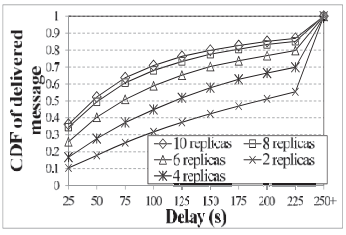


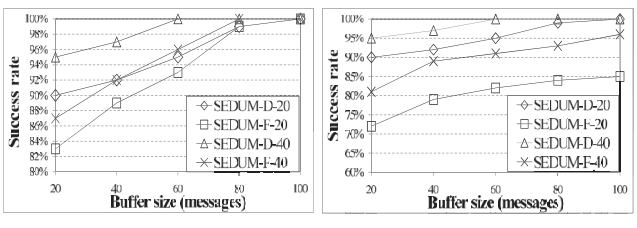
Fig. 13. Effect of the number of replicas per message.

TABLE 1
Comparison of Different Replication Methods

	Optimal tree	Source tree	Binary tree
Ave. overhead	16112.4Mb	17514.4Mb	16943.8Mb
Ave. delay	117s	213s	141s
Ave. throughput	3970Mb	3714Mb	3873Mb

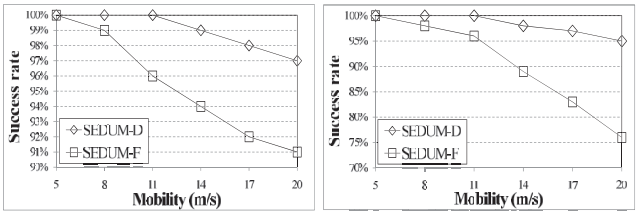
6.6 Comparison of Frequency Utility and Duration Utility

In this section, we compare the frequency utility-based routing and the duration utility-based routing in order to verify our analytical results in Section 3. We use the Manhattan community model to simulate the movement of nodes since this model is more realistic. Figs. 14a and 14b show the success rate of SEDUM using the duration utility and the frequency utility with 0.4 and 4 Mb message sizes, respectively. In the figures, SEDUM-D-20 denotes SEDUM using the duration utility and a 20 m node transmission range. SEDUM-F-20 denotes SEDUM using the frequency utility and a 20 m node transmission range. SEDUM-D-40 and SEDUM-F-40 use a 40 m node transmission range. The experimental results follow SEDUM-D-40 $>$ SEDUM-D-20 $\approx$ SEDUM-F-40 $>$ SEDUM-F-20. Since a larger transmission range enables a node to contact more nodes, SEDUM-F and SEDUM-D with 40 m node transmission range generate higher success rates than with 20 m transmission range. Since the duration utility can more accurately reflect the communication capacity between two nodes, the transmission success rate in SEDUM-D is much higher than in SEDUM-F with the same transmission range. Comparing Fig. 14, we can see that as the message size increases, the success rate of SEDUM-F decreases while that of SEDUM-D remains almost the same. A larger message size increases the probability that messages are dropped during transmission due to broken links. This result is consistent with Theorem 3.1 which implies that larger messages reduce throughput between two nodes. The nodes with a high frequency utility do not necessarily have a long communication time for each contact. A relay node may fail to send a complete message because of the short communication period between two nodes despite of their frequent meetings. The result confirms the higher accuracy of the duration utility in reflecting node transmission capacity than the frequency utility.



(a) Message size = 0.4Mb (b) Message size = 4Mb

Fig. 14. Comparison of duration utility and frequency utility based routings versus buffer size.



(a) Message size = 0.4Mb (b) Message size = 4Mb

Fig. 15. Comparison of duration utility and frequency utility based routings versus mobility.

Figs. 15a and 15b show the relationship between success rate and node mobility with 0.4 and 4 Mb message sizes, respectively. The figures show that as the node mobility increases, the message delivery throughput of both SEDUM-D and SEDUM-F decreases. As the node mobility increases, the expected contact duration between the nodes decreases. Then, fewer messages can be transferred between nodes due to limited contact time. We also observe that SEDUM-D exhibits a slight decrease and SEDUM-F exhibits a dramatic decrease. This is because the duration utility can more accurately reflect the communication capacity between nodes based on node movement patterns. Using the duration utility, messages are always forwarded to the nodes with higher communication capacities with the destination nodes, which leads to a higher transmission success rate. SEDUM-F uses contact frequency as the routing utility that helps to route a message to a node that frequently meets the destination. As node mobility increases, the contact duration between nodes decreases even if they have high contact frequency, thereby increasing the probability of message drops. Consequently, SEDUM-F decreases more quickly than SEDUM-D as node mobility increases.

Comparing Figs. 15a and 15b, we notice that the performance of SEDUM-F is more affected by the message size than SEDUM-D for the same reason as Figs. 14. The frequency utility mainly captures the familiar stranger attribute of the node movement patterns. A node with a high-frequency utility to another node may have small contact duration with the node in a contact, which makes large size messages more likely to be dropped. The duration utility can capture both colocation and familiar stranger attributes of node movement patterns. Since the nodes with long contact durations can transmit messages with large sizes, the transmission success rate of SEDUM-D is only marginally affected by the message size. The experimental results are in line with Theorem 3.2.

7 Conclusions

This work focuses on DTN routing in a social network environment. Most current probabilistic routing protocols forward a message to a node with higher frequency utility to the destination. The contact frequency utility can capture the familiar stranger attribute but fails to completely capture the colocation attribute of node movement patterns in social networks; thus, it cannot accurately reflect the communication capacity of two nodes. In this paper, we propose a duration utility that fully captures both attributes. We theoretically prove that a duration utility can more accurately reflect node communication capacities. We then propose the SEDUM routing protocol for DTNs that fully exploits node movement patterns in the social network to increase delivery throughput and decrease delivery delay while generating low overhead. SEDUM replicates a new message to a certain number of nodes, which hold the replicas until meeting other nodes with higher duration utilities to the destinations. A message is forwarded in this way until one of its replicas reaches its destination. SEDUM includes a buffer management mechanism to improve performance. We also introduce a method using a Markov chain to calculate the minimum number of copies of a message to achieve a given delivery delay. Simulation results show that duration utility-based routing generates higher delivery success rates than frequency utility-based routing. The results also show that SEDUM outperforms the epidemic routing and *Spray and wait* routing in terms of message delivery throughput and delay. In the future, we will implement and evaluate the performance of SEDUM on the GENI real-world testbed [41].

Acknowledgments

This research was supported in part by US NSF grants OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282. An early version of this work [42] was presented in the Proceedings of ICPP' 08.

References

[1] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter, "MDDV: Mobility-Centric Data Dissemination Algorithm for Vehicular Networks," *Proc. First ACM Int'l Workshop Vehicular Ad Hoc Networks (VANET '04)*, 2004.

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

[3] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," *Proc. SIGCOMM*, 2004.

[4] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, B. Durst, V. Cerf, and K. Scott, "Delay-Tolerant Networking: An Approach to Interplantary Internet," *IEEE Comm. Magzine*, vol. 41, no. 6, pp. 128-136, June 2003.

[5] J. Partan, J. Kurose, and B.N. Levine, "A Survey of Practical Issues in Underwater Networks," *Proc. First ACM Int'l Workshop Underwater Networks (WUWNet '06)*, 2006.

[6] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05)*, 2005.

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

[8] Y. Wang et al., "Erasure-Coding Based Routing for Opportunistic Networks," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05)*, 2005.

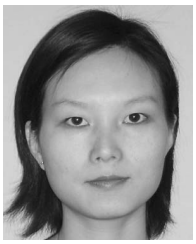
[9] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Single-Copy Case," *ACM/IEEE Trans. Networking*, vol. 16, no. 1, pp. 63-76, Feb. 2007.

[10] 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.

[11] P. Costa, C. Mascolo, M. Musolesi, and G.P. Picco, "Socially-Aware Routing for Publish-Subscribe in Delay-Tolerant Mobile Ad Hoc Networks," *IEEE J. Selected Areas Comm.*, vol. 26, no. 5, pp. 748-760, June 2008.

[12] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for Vehicle-Based Disruption-Tolerant Networks," *Proc. IEEE INFOCOM,* 2006.
[13] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks," *Proc. IEEE Sixth Int'l Symp. World of Wireless Mobile and Multimedia Networks (WOWMOM '05),* 2005.
[14] E. Paulos and E. Goodman, "The Familiar Stranger: Anxiety Comfort, and Play in Public Places," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '04),* 2004.
[15] P. Hui and J. Crowcroft, "How Small Labels Create Big Improvements," *Proc. PERCOMW,* 2007.
[16] F. Li and J. Wu, "MOPS: Providing Content-Based Service in Disruption-Tolerant Networks," *Proc. Int'l Conf. Distributed Computing Systems,* 2009.
[17] J. Ghosh, S.J. Philip, and C. Qiao, "Sociological Orbit Aware Location Approximation and Routing (SOLAR) in MANET," *Ad Hoc Networks,* vol. 5, pp. 189-209, Mar. 2007.
[18] P. Costa, C. Mascolo, M. Musolesi, and G.P. Picco, "Socially-Aware Routing for Publish-Subscribe in Delay-Tolerant Mobile Ad Hoc Networks," *IEEE J. Selected Areas Comm.,* vol. 26, no. 5, pp. 748-760, June 2008.
[19] E.M. Daly and M. Haahr, "Social Network Analysis for Routing in Disconnected Delay-Tolerant Manets," *Proc. MobiHoc,* 2007.
[20] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble Rap: Social-Based Forwarding in Delay Tolerant Networks," *IEEE Trans. Mobile Computing,* vol. 10, no. 11, pp. 1576-1589, Nov. 2010.
[21] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in Delay Tolerant Networks: A Social Network Perspective," *Proc. MobiHoc,* 2009.
[22] V. Conan, J. Leguay, and T. Friedman, "Fixed Point Opportunistic Routing in Delay Tolerant Networks," *IEEE J. Selected Areas Comm.,* vol. 26, no. 5, pp. 773-782, June 2008.
[23] S. Chen, J. Zhang, and Q. Gao, "An Efficient Hybrid Routing Based on Contact History in DTN," *Proc. Seventh Int'l Conf. Wireless and Optical Comm. Networks (WOCN),* 2010.
[24] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance Modeling of Epidemic Routing," *Proc. IFIP,* 2006.
[25] T. Small and Z. Haas, "Resource and Performence Tradeoffs in Delay-Tolerant Wireless Networks," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05),* 2005.
[26] J. Widmer and J.Y.L. Boudec, "Network Coding for Efficient Communication in Extreme Networks," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05),* 2005.
[27] Y. Wang and H. Wu, "A New Paradigm for Pervasive Information Gathering," *IEEE Trans. Mobile Computing,* vol. 6, no. 9, pp. 1021-1034, Sept. 2007.
[28] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli, "Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages," *Proc. MobiHoc,* 2003.
[29] S.S. Haykin, *Kalman Filtering and Neural Networks.* Wiley-Interscience, 2001.
[30] B.N. Levine, A. Balasubrama, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," *Proc. SIGCOMM,* 2007.
[31] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms," *Proc. IEEE INFOCOM,* 2006.
[32] L.B. Koralov and Y.G. Sinai, "Theory of Probability and Random Processes," *Springer-Verlag Berlin Heidelberg,* 2007.
[33] N. Bansal and Z. Liu, "Capacity Delay and Mobility in Wireless Ad-Hoc Networks," *Proc. IEEE INFOCOM,* 2003.
[34] K. Lee, S. Hong, S.J. Kim, I. Rhee, and S. Chong, "Slaw: A Mobility Model for Human Walks," *Proc. IEEE INFOCOM,* 2009.
[35] A. Jardosh, E.M. Belding, K.C. Almeroth, and S. Suri, "Towards Realistic Mobility Models for Mobile Ad Hoc Networks," *Proc. MobiCom,* 2003.
[36] B. Pasztor, M. Musolesi, and C. Mascolo, "Opportunistic Mobile Sensor Data Collection with Scar," *Proc. IEEE Int'l Conf. Mobile Adhoc and Sensor Systems (MASS),* 2007.
[37] T.H. Cormen, *Introduction to Algorithms.* The MIT Press, 2001.
[38] A. Balasubrama, B. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," *Proc. SIGCOMM,* 2007.
[39] M.J. Neely and E. Modiano, "Improving Delay in Ad-Hoc Mobile Networks via Redundant Packet Transfers," *Proc. Conf. Information Sciences and Systems (CISS),* 2003.
[40] A. Keranen, J. Ott, and T. Karkkainen, "The One Simulator for DTN Protocol Evaluation," *Proc. Second Int'l Conf. Simulation Tools and Techniques (SIMUTools '09),* 2009.
[41] Orbit, geni, http://groups.geni.net/geni/wiki/ORBIT, 2011.
[42] Z. Li and H. Shen, "Utility-Based Distributed Routing in Intermittently Connected Networks," *Proc. 37th Int'l Conf. Parallel Processing (ICPP '08),* 2008.

Ze Li received the BS degree in electronics and information engineering from Huazhong University of Science and Technology, China, in 2007. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering of Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks, social networks. He is a student member of the IEEE.



Haiying Shen received the BS degree in computer science and engineering from Tongji University, China in 2000, and the MS and PhD degrees in computer engineering from Wayne State University in 2004 and 2006, respectively. She is currently an assistant professor in the Holcombe Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed and parallel computer systems and computer networks, with an emphasis on peer-to-peer and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the program cochair for a number of international conferences and member of the program committees of many leading conferences. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE.