2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems

GreedyFlow: Distributed Greedy Packet Routing between Landmarks in DTNs

Kang Chen and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {kangc, shenh}@clemson.edu

***Abstract*—Delay Tolerant Networks (DTNs) have attracted significant interests due to the adaptability in areas without infrastructures. In such scenarios, moving data from one place (landmark) to another place (landmark) is essential for data communication between different areas. However, current DTN routing algorithms either fail to fully utilize node mobility or have additional requirements that cannot be satisfied easily (i.e., require base stations or the global traffic distribution). Therefore, in this paper, we propose a distributed greedy routing algorithm, namely GreedyFlow, for efficient packet routing between landmarks. GreedyFlow builds a local traffic map and a global landmark map on each node. The local traffic map indicates the node's knowledge about the amount of traffic (node transition) between landmarks in the area where it primarily visits. It is constructed by collecting encountered nodes' transit frequencies between these landmarks. The global landmark map shows the distribution of landmarks in the system and is built offline. In packet routing, the global landmark map shows the general packet forwarding direction, while the local traffic map helps determine the next-hop landmark on the fastest path in the forwarding direction. As a result, packets are greedily forwarded toward their destination landmarks. Extensive real trace driven experiments demonstrate the high efficiency of GreedyFlow.**

I. Introduction

In delay tolerant networks (DTNs) [1], mobile nodes rely on the encountering to communicate with each other directly without infrastructures. Therefore, such a network structure is suitable for areas where infrastructures are costly to be built or are unavailable. In these scenarios, it is desirable to be able to forward packets from one place (landmark) to another place (landmark), i.e., packet routing between landmarks, to support many practical DTN applications.

For example, people live in villages in a mountain area may wish to communicate with each other through their computers. However, it is costly to build infrastructures to interconnect them or enable satellite connection in each village. In this case, we can exploit the DTN consisting of mobile devices carried by people or vehicles moving in the area to transfer data between these villages [2]. We can also enable satellite connection in one village and rely on the DTN based packet routing between villages to enable the Internet connection for all villages. Though such a connection has a long delay, it is still useful for delay tolerant applications such as email. Similarly, such a communication structure can be used to collect data from sensors attached to animals [3] or deployed in mountain areas without infrastructures. Even in areas with infrastructures, it can be an effective backup scheme to support the dissemination of important messages in extreme scenarios such as disaster and outage [4].

Packet routing between landmarks in DTNs can be realized [5]–[7] by always forwarding a packet to the node that is more likely to move to its destination landmark. In other words, these methods rely on nodes that can frequently visit a packet's destination landmark to deliver the packet. Therefore, the mobility of nodes that rarely visit the destination landmark cannot help forward the packet. When the number of nodes that can frequently visit the destination landmark is limited, the packet routing efficiency is also limited.

To solve this problem, some researchers have proposed to forward a packet along a sequence of landmarks (called landmark path) to better utilize node mobility for efficient packet routing between landmarks [8]–[10]. In each hop, the packet is carried by a node to move from current landmark to the next landmark in the path. With such a design, nodes moving between two consecutive landmarks on the landmark path can help forward the packet, even when these nodes rarely or never visit the packet's destination landmark. This means that node mobility is better utilized to forward packets.

However, these methods require either base stations [8], [9] or the global traffic distribution [10] to calculate the optimal landmark path for each packet. Such requirements cannot be satisfied easily in real DTNs. First, due to the long delay in DTN routing, the global traffic information cannot be updated timely on each node. Second, in some DTN scenarios, such as battlefields and mountain areas, it is hard to build base stations. Such a limitation poses a significant challenge on realizing efficient packet routing between landmarks in DTNs.

To solve the above challenge, we propose GreedyFlow, a distributed packet routing algorithm in DTNs. In GreedyFlow, to better utilize node mobility, packets are forwarded in a landmark-by-landmark manner. Each node maintains a global landmark map and a local traffic map (Figure 1). The network is split into sub-areas, each of which is represented by a landmark. The global landmark map indicates the distribution of landmarks in the network and is built offline. The local traffic map reflects a node's knowledge about how frequently nodes transit between landmarks where it primarily visits. When a node meets another node, it collects the encountered node's transit frequencies between landmarks covered by its local traffic map to update the traffic map.

The global landmark map and local traffic map are used to

978-1-4673-9101-6/15 $31.00 © 2015 IEEE
DOI 10.1109/MASS.2015.67

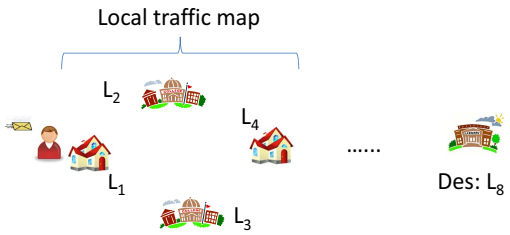


Fig. 1: Illustration of the design rationale.

guide packet routing. The basic idea is to greedily forward a packet to a landmark closest to the destination landmark (called temporary destination landmark) within current packet holder's local traffic map. When a node (say N_i) receives a packet, it first determines the temporary destination landmark for the packet. Then, the node determines the fastest landmark path to the temporary destination landmark on its local traffic map and selects the next landmark on the path as the next-hop landmark. Next, N_i forwards the packet to the node that is predicted to move to the selected next-hop landmark.

For example, in Figure 1, for a packet destined to far-away landmark L_8, the node uses its global landmark map to identify the landmark that is closest to L_8, i.e., L_4, and uses the local traffic map to find the fastest path to L_4. With such a design, the packet is always forwarded to the landmark closest to the destination landmark through the fastest path based on local information. That is, packets are greedily forwarded toward their destination landmarks based on the two maps.

In GreedyFlow, all packet forwarding decisions are made locally without the requirement of base stations or global traffic distribution. This is the major contribution of this work compared to previous works [8]–[10] that also adopt landmark based relay. In summary, the contributions of this paper include

- We propose a distributed traffic map generation method, which enables each node to learn the node transition frequencies between landmarks in the area where it primarily visits. Such a design rationale is similar to the phenomenon that people often know the traffic volume between places they visit a lot.
- We propose a fully distributed greedy packet routing algorithm that can realize efficient packet routing between landmarks in DTNs based on the local traffic map and the global landmark map.
- Extensive real trace based experiments demonstrate the efficiency of the proposed algorithm.

The remaining of this paper is arranged as follows. Section II introduces related work. Section III presents the detailed system design. Section IV conducts performance evaluation through real trace driven experiments. Finally, Section V concludes the paper with remarks on our future work.

II. Related Work

A. Packet Routing between Landmarks in DTNs

Packet routing between landmarks in DTNs [5]–[10] has been studied recently. The authors in [5] observe the long term mobility pattern of each node and use such information to forward packets to nodes that frequently move to their destinations. GeoOpps [6] routes packets to geographical locations through vehicle networks. It always forwards packets to vehicles on the route with the smallest minimal estimated time of delivery (METD). In the work of [7], a packet is always forwarded to the node that has closer distance to its destination landmark. These methods mainly rely on nodes that are likely to visit the destination landmarks. When the number of such nodes is limited, the routing efficiency is also limited.

In order to better utilize node mobility, researchers have proposed to forward packets in a landmark-by-landmark manner to reach the destination landmarks [8]–[10]. In LOUVER [8], base stations are built on road intersections for packet relay. Vehicle mobility is then exploited to forward packets from one base station to another to reach the destination area. DTNFLOW [9] expands vehicle networks in LOUVER to general DTNs. It splits the network into sub-areas represented by landmarks. Then, predicted node mobility is used to carry packets from one landmark to another landmark. Geomob [10] utilizes the global traffic distribution to forward packets to different areas through the landmark based relay.

Above methods need either base stations or the global traffic distribution. These requirements can hardly be realized easily in DTNs. Such limitations constraint the feasibility of these methods. On the contrary, our method does not rely on any base stations and makes forwarding decisions locally, which is more suitable for distributed DTNs.

B. Packet Routing between Nodes in DTNs

There are already many algorithms for packet routing between nodes in DTN [11]–[19]. PROPHET [11] updates two node's future meeting probability upon their encountering and ages it over time. It always forwards a packet to the node that has a higher probability of meeting its destination. RAPID [12] and MaxContribution [13] specify different packet forwarding and storage priorities to realize different routing performances, e.g., maximal success rate and minimal delay. The work in [14] further propose to exploit transient contact patterns to select forwarder for more efficient packet routing.

Considering that mobile device owners often belong to certain social networks, social networking properties have been utilized for packet routing in DTNs [15]–[19]. MOPS groups frequently encountered nodes as communities and assigns different roles to nodes with different community visiting patterns to facilitate the publish/subscribe service in DTNs. BUBBLE [16] first forwards a packet to the community that contains its destination and then routes the packet within the community. SimBet [17] considers both a node's centrality and its similarity with the packet destination to evaluate a node's suitability to carry the packet. The works in [18] and [19] exploit fixed community and transient community structure in DTNs for efficient packet routing, respectively.

III. System Design

In this section, we first introduce the network modeling and design rationale. We then present how to construct the global

landmark map and the local traffic map. Finally, we introduce the detailed packet routing algorithm.

A. Network Modeling and Design Goal

We assume a DTN consisting of n nodes, denoted by N_i ($i \in [1, n]$). We also assume that the network is split into sub-areas, each of which is represented by a landmark. A landmark is often selected as the area where nodes gather together, such as a village in the rural area and a building on the campus. This means that the landmark is just the notation of an area and does not require any base stations to be built. We assume there are m landmarks, denoted by L_j ($j \in [1, m]$). Then, node mobility can be regarded as continuous transit between landmarks. We also assume that nodes present certain landmark visit patterns, which exists in many DTNs. For example, in DTNs consisting of mobile devices carried by people in rural areas or students on a campus, a person or a student may mainly transit between a few landmarks (i.e., villages or buildings).

The goal of this paper is to realize efficient and distributed packet routing between identified landmarks in DTNs. This means that no infrastructures or global information is needed to support packet routing, and the packet forwarding decision is made locally. Such a function can support many interesting services or applications, such as data communication between rural villages, where infrastructures are too costly to build.

B. Rationale of System Design

In this work, we relay packets in a landmark-by-landmark manner to reach their destination landmarks. Such a strategy can better utilize node mobility for routing packet to landmarks. For example, suppose we need to forward packets from L_1 to L_{16}. With the landmark based routing strategy, these packets are forwarded through a landmark path, say $L_1 \rightarrow L_6 \rightarrow L_{12} \rightarrow L_{16}$. As a result, nodes moving between any two neighboring landmarks on the path, e.g., L_1 and L_6, can forward these packets one step closer to destination L_{16} even though these nodes rarely or never visit L_{16}. This means that more node mobility is utilized for packet routing between landmarks, leading to better routing efficiency.

1) Challenges: The key problem in these methods is how to select a suitable landmark path for each packet. Recall that packets are carried by nodes to move from one landmark to another. This means that the more frequently nodes move from one landmark to a neighbor landmark, say L_1 to L_6, the more quickly a packet can be forwarded from L_1 to L_6, and the smaller the expected delay of this forwarding step. Then, we can calculate the expected delay of a landmark path as the sum of the expected delays on each hop and select the landmark path with the smallest expected delay for packet forwarding. However, how to find the landmark path with the smallest expected delay efficiently and accurately is non-trivial. This is because nodes often are sparsely distributed in DTNs. Previous methods [8]–[10] realize this step by either building extra base stations on each landmark [8], [9] or requiring that each node knows the global traffic distribution [10]. Unfortunately, both requirements cannot be satisfied easily in real DTNs.

2) Our Solution: GreedyFlow routes packets through landmark path in a distributed manner. Without global information, GreedyFlow does not try to determine the whole relay path for each packet. Rather, it only selects the next-hop landmark for each packet based on the local information on current carrier to greedily route it to its destination landmark. To realize this goal, GreedyFlow builds a global landmark map and a local traffic map on each node, which represent the node's understanding of the landmark distribution in the network and node transition frequencies between landmarks in the area where the node primarily visits, respectively. The two maps help decide the next-hop landmark for each packet.

Such a design rationale matches with our daily experiences. People usually know the traffic delays on roads connecting places they visit frequently and the general direction to reach a far-away unfamiliar place (e.g., in south or north). Then, people can use such knowledge to greedily relay a message to a far-away place efficiently.

C. Global Landmark Map

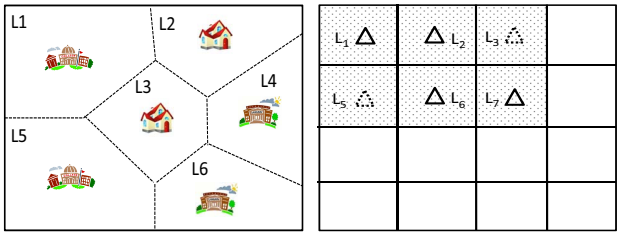
The global landmark map shows the distribution of landmarks in the network. It includes the GPS position of each landmark and the neighboring relationships between landmarks, i.e., the neighbor landmarks of each landmark. It is designed to ensure that packets are forwarded on the right direction towards their destination landmarks.

The global landmark map is generated and maintained by the network administrator. When a DTN is deployed, the administrator selects landmarks in the network. It can collect the mobility information of nodes in the system to determine landmarks. When a node joins in the system, it first obtains the global landmark map from the network administrator. The global landmark map usually remains unchanged for a relative long period of time, which means that the global landmark map on each node does not need frequent updates. When the global landmark map changes, each node can obtain the updated version when it has access to the network administrator, e.g., when moving to a place with network connection.

We split the network into sub-areas based on landmarks and let each landmark be responsible for the sub-area it resides in. The area between two landmarks is evenly split to the two neighboring sub-areas (i.e., the borderline passes through the midpoint of the line connecting the two landmarks and is perpendicular to it), as shown in Figure 2(a). Each sub-area is stored as the list of vertices in clockwise direction. When a node enters the sub-area of a landmark, we regard it as transiting to the landmark. As a result, node mobility can be summarized as consecutive transitions between landmarks.

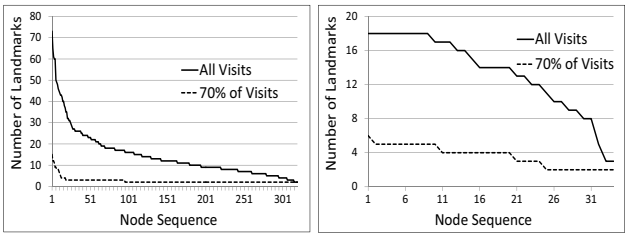
D. Local Traffic Map

Each node maintains a local traffic map to record its knowledge about how frequently nodes transit between landmarks in the area where it primarily visits. It is designed to select the locally optimal landmark path on the direction to the destination landmark for packets carried by the node. Below, we first introduce how to determine the area a node visits



(a) Global landmark map. (b) Coverage of local traffic map.

Fig. 2: Example of global landmark map and local traffic map coverage.



(a) DART trace. (b) DNET trace.

Fig. 3: Distribution of the # of visited landmarks.

primarily, i.e., the coverage of the local traffic map, and then present how to construct the local traffic map on each node.

1) Local Traffic Map Coverage: As previously introduced, the local traffic map helps determine the next-hop landmark on the path that can lead to the destination landmark quickly. Therefore, the more landmarks the local traffic map includes, the more likely that the next-hop landmark that can lead to smaller expected delay can be found. However, nodes often have limited storage resources, and the information dissemination often has a long delay in DTNs. This means that a node can neither store the node transit frequencies between all landmarks nor collects such information timely in DTNs. Therefore, we need to determine which landmarks to be included in the local traffic map.

To solve this problem, we first examine how nodes visit landmarks in DTNs. We analyzed two real DTN traces: Dartmouth Campus Trace (DART) [20] and DieselNet AP Trace (DNET) [21]. The former shows the association records of the WiFi access points (APs) and students' devices on Dartmouth campus, while the latter includes the AP association records of 34 buses in a college town (UMass). We preprocessed the trace to abstract landmarks, i.e., a building or an area with a certain size, from the two traces and merge neighboring records with the same device and landmark. Finally, the DART trace contains 320 nodes and 159 landmarks, and the DNET trace has 34 nodes and 18 landmarks.

We measured the total number of landmarks a node visits in the trace and the number of landmarks that account for 70% of a node's landmark visits in the trace. The test results with the two traces are shown in Figure 3(a) and Figure 3(b), respectively. We ranked nodes in descending order of the two metrics in the two figures. From the two figures, we find that more than 80% of nodes in the DART trace and all nodes in the DNET trace visit fewer than 20 landmarks. This means that most nodes visit a few landmarks throughout the two traces. Besides, more than 90% of nodes in both traces spend their 70% of visits on fewer than 5 landmarks. Such results demonstrate that nodes often only frequently transit between a limited number of landmarks.

Therefore, we let the local traffic map of each node only include landmarks in the area where the node primarily visits. Specifically, each node ranks the landmarks in decreasing order of its visit frequencies and selects the first k landmarks that account for $V_f\%$ of its total landmark visits. The area covered by these landmarks, i.e., the area covered by the most left-up landmark and the most right-bottom landmark, is defined as the coverage of the local traffic map. Figure 2(b) shows an example of the coverage of a local traffic map. In this example, the primarily visited landmarks are L_1, L_2, L_6, and L_7. Then, the shadowed area is determined as the coverage of the local traffic map, which includes L_1, L_2, L_3, L_5, L_6, and L_7. We can see that the larger V_f is, the more information the local traffic map provides, but also the more overhead incurred. Therefore, a suitable V_f can be determined based on the requirement on routing efficiency and overhead.

2) Local Traffic Map Construction: Each node updates the local traffic map upon encountering other nodes. Specifically, suppose the coverage of a node's local traffic map is $C_i = \{L_a, L_b, L_c, L_d\}$, $\{a, b, c, d\} \in [1, m]$, it queries each encountered node about how frequently it transits between these landmarks, i.e., from L_x to L_y, $x, y \in \{a, b, c, d\}$ and $x \neq y$. To enable such a function, each node builds an individual landmark transit table to record its transit frequencies between landmarks, as shown in Table I. Each row represents the node's transit frequency (i.e., how many transits per day) between two neighboring landmarks. The third column and the fourth column are the transit frequencies from L_x to L_y and from L_y to L_x, respectively.

TABLE I: Landmark transit table.

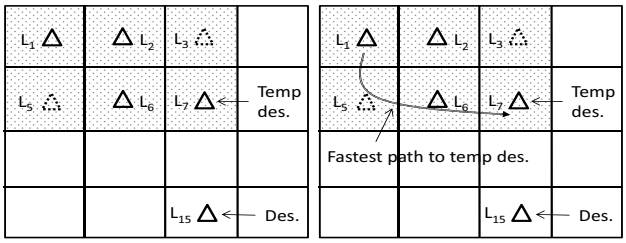
$Landmark_x$	$Landmark_y$	$Frequency_{xy}$	$Frequency_{yx}$
L_1	L_5	3	2.5
L_2	L_{13}	8	9
L_{15}	L_{24}	7	4
...	...	...	...

Each node collects encountered nodes' transit frequencies to update its local traffic map. For example, suppose a node's local traffic map contains landmark L_1 and landmark L_2. Then, to get the overall transit frequency from L_1 to L_2, the node gets every encountered node's transit frequency from L_1 to L_2 and sums up these transit frequencies. However, in this process, a node may meet another node and obtain its transit frequency multiple times. Then, to avoid summing up a node's transit frequency multiple times, each node maintains a record on which nodes' transit frequencies have already been included in the calculation of the overall transit frequency from one landmark to another. We name such a record as the transit element table. Table II shows an example of transit element table on a node for the transition from L_1 to L_2.

The "overall frequency" in each transit element table finally is stored in the local traffic map. Table III shows the traffic

TABLE II: Transit element table.

Transit	Node	Frequency
$L_1 \rightarrow L_2$	N_8	0.7
	N_5	2
	N_7	1.5
	...	...
	Overall	10.5



(a) Temporary destination landmark. (b) Fastest path to the temp des.

Fig. 4: Determining the next-hop landmark.

map following the example in Figure 2(b).

TABLE III: Local traffic map.

Landmark ID	Neighbor Landmark	Frequency
L_1	L_2	10.5
	L_5	13
L_2	L_1	9
	L_6	4
	L_3	10.2
L_3	L_2	4.2
	L_7	5.5
...	...	...

In summary, in GreedyFlow, when a node, say N_i, meets another node, say N_j, N_i obtains N_j's transit frequencies between each pair of landmarks covered in N_i's local traffic map to update its local traffic map. In detail, for $L_x \rightarrow L_y$, if N_j's transit frequency (denoted by f_{xy}^j) is not 0, N_i first checks whether it has a transit element table for this transit. If not, it creates a transit element table for this transit with only one entry, i.e., the entry for N_j. N_i then updates N_j's entry in the transit element map to f_{xy}^j. N_i also updates the overall frequency for transit $L_x \rightarrow L_y$ accordingly in both the transit element table and the local traffic map.

For example, suppose a node's transit element table for $L_1 \rightarrow L_2$ is as shown in Table II, and its local traffic map is as shown in Table III. When the node meets N_8 and finds that N_8's transit frequency for $L_1 \rightarrow L_2$ has changed to 1.4, it updates the entry for N_8 in its transit element table to 1.4. Then, it updates the overall transit frequency for $L_1 \rightarrow L_2$ to 11.2 in both the transit element table and the local traffic map.

E. Packet Routing in GreedyFlow

We introduce the packet routing algorithm in this section. We first give out the overview and then present details.

1) Overview: The packet routing works in a greedy manner in GreedyFlow. When a node generates or receives a packet, it checks its local traffic map and the global landmark map to decide the next-hop landmark to forward the packet. Specifically, since a node's local traffic map may not include the destination landmark of the packet, it first selects a temporary destination landmark in the local traffic map that has the closest distance to the destination landmark. Each landmark's distance to the destination landmark is obtained from the global landmark map. This ensures that the packet forwarding is always on the right direction. We introduce this step in Section III-E2. Then, the node finds the fastest path from current landmark to the temporary destination landmark based on the local traffic map and selects the next landmark on the path as the next-hop landmark. We introduce how to determine such a path in Section III-E3. Figure 4(b) shows that the fastest path from L_1 to temporary destination landmark L_7 is $L_1 \rightarrow L_5 \rightarrow L_6 \rightarrow L_7$ and the next-hop landmark is L_5.

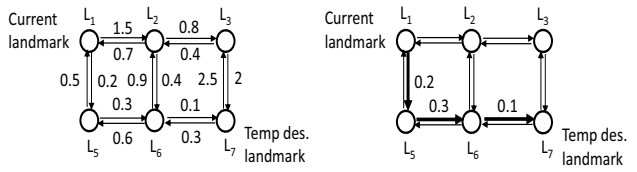
Then, the packet is forwarded to the selected next-hop landmark L_5. In detail, the node queries its neighbors about where they are predicted to move to and forwards the packet to the node that is predicted to move to the selected next-hop landmark L_5. The details on how each node predicts the next landmark it is going to transit to is introduced in Section III-E4. When the node arrives at L_5, it repeats the above process to further forward the packet. Finally, the packet is greedily forwarded toward its destination landmark. However, a node that is predicted to move to the next-hop landmark may not always be found. A node may not always move to the predicted landmark. A packet may revisit the same landmark during the forwarding. We discuss how to handle these exceptions in Section III-E5.

2) Selecting Temporary Destination Landmark: The packet carrier first checks whether its local traffic map contains the destination landmark of the packet or not. If yes, the temporary destination landmark is the destination landmark. Otherwise, the temporary destination landmark is the landmark in the local traffic map that has the closest distance to the destination landmark. The distance between two landmarks is calculated as the minimal number of landmark hops between them. As shown in Figure 4(a), suppose the destination landmark is L_{15}, landmark L_7 is selected as the temporary destination landmark since it has the closest distance to L_{15}.

3) Determine the Fastest Path to the Temporary Destination: As introduced in Section III-B, the expected delay to a forwarding hop is determined by the frequency of node transition on the hop. We use such a property to find the fastest path to the temporary destination landmark.

For better illustration, we abstract each landmark as a circle and two circles are connected if they represent two neighbor landmarks. We also abstract the node transition from one landmark to another landmark as a link connecting the two landmarks. Therefore, two neighbor circles are connected by two links in two direction. Each link has a weight, which represents the delay to forward a packet through the link. It is calculated as $1/f_t$, where f_t is the frequency of node transition in the direction of the link. Figure 5(a) shows the abstracted local traffic map.

Using such a graph, the problem of finding the fastest path to the temporary destination landmark turns into the problem of finding the shortest path to the temporary destination



(a) Abstracted local traffic map. (b) The selected fastest path.

Fig. 5: Determine the fastest path to the temporary destination.

landmark. We use the Dijkstra algorithm [22] to fulfill this task. In detail, we take the current landmark L_1 as the root. We then calculate its shortest paths to all nodes iteratively until the shortest path to the temporary destination landmark L_7 is found. The bold links in Figure 5(b) show the fastest path from L_1 to L_7. Finally, the node selects the next landmark on the path as the next-hop landmark, i.e., L_5.

4) Predicting Node Mobility: In order to forward a packet from current landmark to the determined next-hop landmark, the current packet holder queries its neighbors about where they are most likely to transit to and forwards the packet to the node that is going to move to the next-hop landmark. To fulfill this function, each node predicts its next transit upon moving to a new landmark. In detail, each node uses its historical landmark visit information to feed the Order-1 Markov predictor to deduce the landmark it is going to transit to. In detail, suppose a node's landmark visit history can be represented by $V_H = L_{x_1} L_{x_2} L_{x_3} \cdots L_{x_{n-1}} L_{x_n}$. Then, the probability that the node is going to visit landmark $L_{x_{n+1}}$ can be calculated by

$$Pr(L_{x_n} L_{x_{n+1}} | L_{x_{n-1}} L_{x_n}) = \frac{Pr(L_{x_{n-1}} L_{x_n} L_{x_{n+1}})}{Pr(L_{x_{n-1}} L_{x_n})}, \quad (1)$$

where

$$Pr(L_{x_{n-1}} L_{x_n} L_{x_{n+1}}) = \frac{N(L_{x_{n-1}} L_{x_n} L_{x_{n+1}})}{N(All_3)} \quad (2)$$

and

$$Pr(L_{x_{n-1}} L_{x_n}) = \frac{N(L_{x_{n-1}} L_{x_n})}{N(All_2)} \quad (3)$$

Note that $N(L_{x_{n-1}} L_{x_n} L_{x_{n+1}})$ denotes the number of times that the node visits landmarks $L_{x_{n-1}}$, L_{x_n}, and $L_{x_{n+1}}$ consecutively, $N(L_{x_{n-1}} L_{x_n})$ denotes the number of times that the node visits landmarks $L_{x_{n-1}}$ and L_{x_n} consecutively, and $N(All_k)$ $(k = 2, 3)$ means the number of visits on consecutive k landmarks in history. Then, the landmark that leads to the largest transit probability in Equation (1) is selected as the landmark that the node is going to move to.

5) Handle Exceptions: We further design three additional schemes to handle three exception cases. Firstly, a node that is expected to transit to the next-hop landmark for a packet cannot be found. Secondly, the packet carrier moves to a landmark that is different from the predicted one.

To handle the first exception, we simply let the current carrier of the packet continue carrying it until arriving at another landmark. If this landmark is the next-hop landmark of the packet, the exception is solved automatically. Otherwise, the first exception turns into the second exception: the carrier moves to an unexpected landmark. For this exception, the current carrier checks whether a node that is predicted to move to the next-hop landmark of the packet can be found. If yes, it forwards the packet to the node. Otherwise, it forwards the packet to the neighbor node that has the highest centrality, which handles the packet as mentioned in Sections III-E2, III-E3, III-E4. The centrality of a node is defined as the number of nodes it can meet in a unit time. We select such a node to carry the packet since it can meet more nodes and thus has more options on forwarder selection.

6) Summary: We summarize the process of packet routing in GreedyFlow as follows.

- When a node generates a packet or carries a packet to its next-hop landmark, the node follows the method introduced in Section III-E2 to determine its temporary destination landmark.
- The node determines the fastest path to the temporary destination landmark based on its local traffic map and selects the next-hop landmark for the packet by following the method in Section III-E3
- Then, the node checks whether a neighbor node is predicted to move to the next-hop landmark. If yes, it forwards the packet to the neighbor node. The prediction of node transition is introduced in Section III-E4.
- If no suitable carrier can be found or the selected carrier moves to a landmark other than the next-hop landmark, the packet is handled by the schemes in Section III-E5.
- The above process repeats until the packet arrives at the destination landmark.

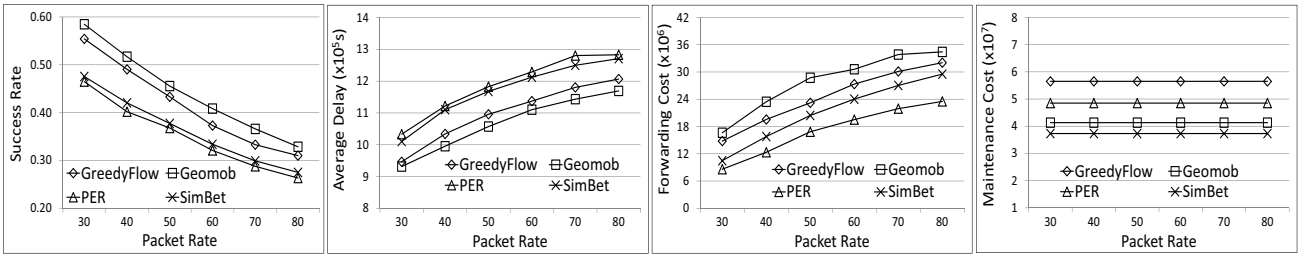
IV. Performance Evaluation

We conducted event driven experiments with the two real traces, namely Dartmouth campus trace (DART) [20] and DieselNet AP trace (DNET) [21], to evaluate the performance of the GreedyFlow in comparison with three state-of-art methods. Section III-D1 introduces the two traces.

A. Experiment Settings

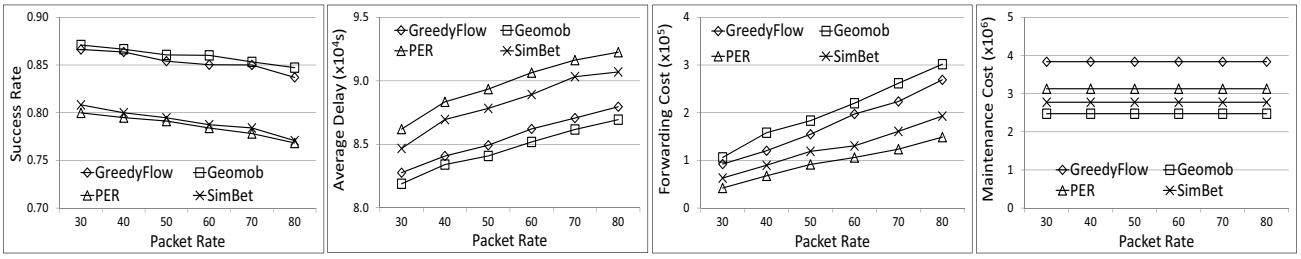
We compared GreedyFlow with three representative DTN routing algorithms: Geomob [10], PER [23] and SimBet [24]. Geomob is similar to GreedyFlow that it also routes packets in a landmark-by-landmark manner. However, it requires that all nodes know the traffic distribution in the network to decide the landmark path for each packet. As mentioned in the introduction, such a requirement is not practical. We still use it to measure whether GreedyFlow can lead to comparable performance with Geomob. PER estimates each node's probabilities to visit each landmark and forwards packets to nodes that have a high probability to visit their destination landmarks before they expire. SimBet evaluates a node's suitability to carry a packet by considering both its centrality and its visit frequency with the packet's destination landmark.

In the experiment, we used the first 1/3 of both traces for initialization, in which nodes build the local traffic map in GreedyFlow and accumulate related metrics, e.g., landmark visit frequencies, in comparison methods. After this step, we



(a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Maintenance cost.

Fig. 6: Performance with different packet rates using the DART trace.



(a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Maintenance cost.

Fig. 7: Performance with different packet rates using the DNET trace.

generated packets with randomly selected destination landmarks at the rate of r_p packets per landmark per day. The TTL (Time to Live) of each packet was set to 30 days in the DART trace and 6 days in the DNET trace. When a packet's TTL expires, it is dropped directly. We assume that each packet has the same size of 1 KB and the available storage on each node is m_n KB. We set r_p and m_n to 50 and 150 by default and varied them for extensive performance evaluation. When the storage on a node is full, the oldest packet is dropped. We set V_f to 70 since we find that it is sufficient for efficient packet routing. We set the confidence interval to 95%.

We used four metrics in the experiments: success rate, average delay, forwarding cost, and maintenance cost. The success rate refers to the percentage of packets that have been successfully delivered within the TTL. The average delay refers to the average delay of successfully delivered packets. The forwarding cost refers to the number of packet forwarding operations. The maintenance cost refers to the number of messages exchanged between nodes to support packet routing (e.g., transit information in GreedyFlow and landmark visit frequencies in comparison methods).

B. Performance with Different Packet Rates

We first conduct performance evaluation with different packet rates (r_p). We varied r_p from 30 to 80 in the test.

1) Success Rate: Figures 6(a) and 7(a) show the success rates of the four methods in the experiments with different packet rates using the DART trace and the DNET trace, respectively. We see from the two figures that the success rates follow: $Geomob \approx GreedyFlow > SimBet \approx PER$.

Geomob and GreedyFlow lead to higher success rate than the other two methods because they forward packets in a landmark-by-landmark manner, thereby better utilizing node mobility for more efficient packet routing. Such a result demonstrates the advantage of the landmark path based routing strategy in routing packets to different landmarks. GreedyFlow shows slightly lower success rate than Geomob. This is because nodes in Geomob are assumed to know the global traffic distribution beforehand, while nodes in GreedyFlow only know the traffic distribution in the area they often visit. However, the assumption in Geomob is not practical in real DTNs. We see that the difference on success rate is very marginal. This means that GreedyFlow can also effectively select a fast landmark path for each packet even without the global information. In other words, the performance of the distributed GreedyFlow is comparable to that of Geomob.

We also find that SimBet shows slightly higher success rate than PER. This is because SimBet considers not only a node's visiting frequency with the destination landmark but also its centrality in the network, while PER only considers the visit frequency. Then, in addition to nodes that frequently visit packet destinations, SimBet also utilizes nodes with a high centrality to route packets. As a result, SimBet utilizes more mobile nodes for routing, leading to higher success rate.

2) Average Delay: Figures 6(b) and 7(b) show the average delays of the four methods in the experiments with different packet rates using the DART trace and the DNET trace, respectively. We find from the two figures that the average delays follow: $Geomob < GreedyFlow < SimBet < PER$.

Geomob has the least average delay because each node knows the global traffic distribution, which enables it to always select the landmark path with the minimal expected delay for each packet. In GreedyFlow, each node only knows the node transit frequencies between landmarks in the area where it primarily visits. Therefore, it has slightly higher average delay than Geomob. However, we see that the difference is very

small. Since GreedyFlow does not require the global traffic information as Geomob, it is more suitable for DTNs.

SimBet and PER exhibit much higher average delay than Geomob and GreedyFlow. This is because they rely on nodes that frequently visit destination landmarks for packet routing, and such nodes may not always exist. On the contrary, Geomob and GreedyFlow forward packets in the landmark-by-landmark manner to reach their destinations. Nodes that rarely visit the destination landmark of a packet can still be utilized to forward it to a landmark closer to the destination landmark, leading to a small average delay of successfully delivered packets.

3) Forwarding Cost: Figures 6(c) and 7(c) show the forwarding costs of the four methods in the experiments with different packet rates using the DART trace and the DNET trace, respectively. We find from the two figures that the forwarding costs follow: $Geomob{>}GreedyFlow{>}SimBet{>}PER$.

PER generates the least packet forwarding cost because it only forwards a packet to the node that has a high probability of delivering the packet before it expires, leading to few forwarding opportunities. SimBet works in a similar manner as PER but additionally considers centrality for forwarder selection, resulting in more packet forwarding than PER.

GreedyFlow and Geomob generate more packet forwarding cost than PER and SimBet because they forward packets in a landmark-by-landmark manner, which exploits more nodes to carry packets. However, we can see that the increase of forwarding cost is not significant, which is worthwhile considering their improvement on routing efficiency.

4) Maintenance Cost: Figures 6(d) and 7(d) show the maintenance costs of the four methods in the experiments with different packet rates using the DART trace and the DNET trace, respectively. We see from the two figures that the maintenance costs follow: $GreedyFlow{>}SimBet{>}PER{>}Geomob$.

GeoMob generates the least maintenance cost because it assumes that nodes already know the global traffic distribution beforehand. Therefore, nodes only need to exchange their probabilities of going to neighbor landmarks to support packet routing. In PER, encountering nodes exchange their probabilities to visit all landmarks to determine packet forwarders, leading to more maintenance cost than Geomob. In addition to landmark visit frequencies, nodes in SimBet also exchange centrality information. Therefore, it has higher maintenance cost than PER. GreedyFlow has more maintenance cost than others because nodes need to exchange transit frequencies for local traffic map update. However, we see that the maintenance cost of GreedyFlow is on the same level with others.

Combining all above results, we conclude that the two methods that forward packets through landmark paths lead to better performance than other methods. Such a result justifies the correctness of such a packet routing strategy. We also see that GreedyFlow shows close performance with Geomob, which however requires global traffic distribution. Such a result demonstrates that GreedyFlow can realize efficient packet routing in a fully distributed manner.

C. Performance with Different Memory Sizes

We further evaluate the performance of the four methods with different memory sizes on each node (m_n). We varied m_n from 100 to 200 in the test.

1) Success Rate: Figures 8(a) and 9(a) illustrate the success rates of the four methods in the experiments with different memory sizes using the DART trace and the DNET trace, respectively. We see from the two figures that the success rates follow: $Geomob{\approx}GreedyFlow{>}SimBet{\approx}PER$. Such a result is consistent with those in Figures 6(a) and 7(a) for the same reasons. We also find that when the memory size on each node increases, the success rates of all methods increase. This is because when the memory size increases, each node can carry more packets. This means that the capacity of the network is enhanced, leading to more successful packets.

2) Average Delay: Figures 8(b) and 9(b) plot the average delays of the four methods in the experiments with different memory sizes using the DART trace and the DNET trace, respectively. We find from the two figures that the average delays follow: $Geomob{<}GreedyFlow{<}SimBet{<}PER$. We see that this relationship is the same as in Figures 6(b) and 7(b) due to the same reasons. Similarly, we see that when the memory size on each node increases, the average delays of all methods decrease. This is because when the memory size increases, more packets can be carried by nodes that are most likely to deliver them to their destinations, thereby reducing the average delay of successfully delivered packets.

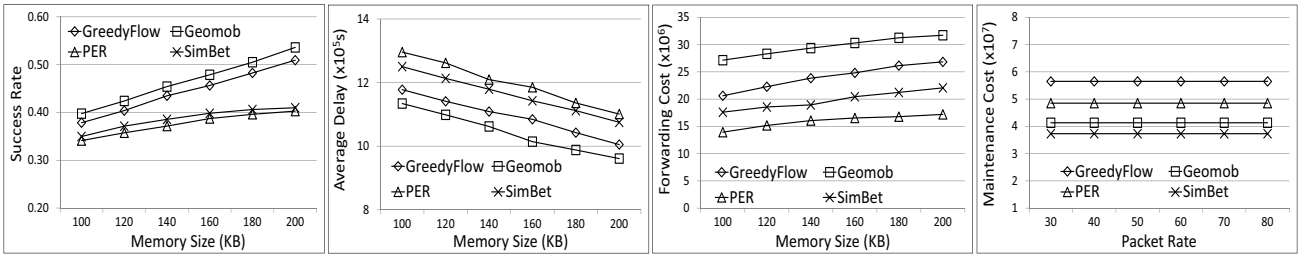
3) Forwarding Cost: Figures 8(c) and 9(c) show the forwarding costs of the four methods in the experiments with different memory sizes using the DART trace and the DNET trace, respectively. The two figures show that the average forwarding costs follow: $Geomob{>}GreedyFlow{>}SimBet{>}PER$. Again, this is the same as in Figures 6(c) and 7(c) due to the same reasons. We also find that when the memory size increases, the forwarding costs of all methods increase. This is because when each node can carry more packets, there are more packet forwarding.

4) Maintenance Cost: Figures 8(d) and 9(d) show the maitenance costs of the four methods in the experiments with different memory sizes using the DART trace and the DNET trace, respectively. The two figures show that the maintenance costs follow: $GreedyFlow{>}SimBet{>}PER{>}Geomob$

We can find that this relationship is the same as in Figures 6(d) and 7(d). Also, the maintenance costs remain unchanged with different packet rates or memory sizes. This is because the maintenance costs of these methods are only affected by the number of node encountering. Since we use the same traces in the each test, the maintenance costs remain the same. The results with different memory sizes further confirm the superior performance of GreedyFlow.

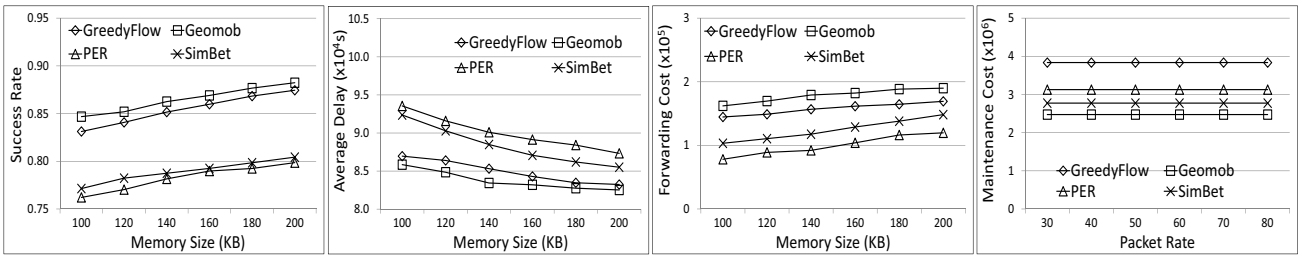
V. Conclusion

Data transmission between different places (landmarks) in a DTN can be used in many applications. However, previous algorithms on packet routing between landmarks either cannot achieve high efficiency or have impractical requirements. In



(a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Maintenance cost.

Fig. 8: Performance with different memory sizes using the DART trace.



(a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Maintenance cost.

Fig. 9: Performance with different memory sizes using the DNET trace.

this paper, we propose a novel algorithm, namely GreedyFlow, to route packets between landmarks in DTNs in a fully distributed manner. To better utilize node mobility, GreedyFlow forwards packets in a landmark-by-landmark manner to let them gradually reach their destination landmarks. Each node collects node transit frequencies between landmarks in the area it primarily visits and uses such information to build a local traffic map. A global landmark that shows the distribution of landmarks is also built on each node off-line. The two maps are used to greedily forward packets toward their destination landmarks. Extensive real trace driven experiments show that GreedyFlow has better performance than state-of-art routing algorithms and can achieve performance comparable to the routing algorithm that requires the global traffic information on each node. In the future, we plan to further enhance routing efficiency by considering social communities in DTNs.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, Microsoft Research Faculty Fellowship 8300751.

REFERENCES

[1] S. Jain, K. R. Fall, and R. K. Patra, "Routing in a delay tolerant network," in *Proc. of SIGCOMM*, 2004.

[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. of SIGCOMM*, 2003.

[3] 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," in *Proc. of ASPLOS-X*, 2002.

[4] U. Weinsberg, A. Balachandran, N. Taft, G. Iannaccone, V. Sekar, and S. Seshan, "CARE: Content aware redundancy elimination for disaster communications on damaged networks," *CoRR*, 2012.

[5] J. Kurhinen and J. Janatuinen, "Geographical routing for delay tolerant encounter networks," in *Proc. of ISCC*, 2007.

[6] I. Leontiadis and C. Mascolo, "GeOpps: Geographical opportunistic routing for vehicular networks," in *Proc. of WOWMOM*, 2007.

[7] J. Lebrun, C. nee Chuah, D. Ghosal, and M. Zhang, "Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks," in *Proc. of VTC*, 2005.

[8] K. C. Lee, M. Le, J. H0Ł1rri, and M. Gerla, "LOUVRE: Landmark overlays for urban vehicular routing environments." in *Proc. of VTC Fall*, 2008.

[9] K. Chen and H. Shen, "DTN-FLOW: Inter-landmark data flow for high-throughput routing in DTNs." in *Proc. of IPDPS*, 2013.

[10] L. Zhang, B. Yu, and J. Pan, "GeoMob: A mobility-aware geocast scheme in metropolitans via taxicabs and buses." in *Proc. of INFOCOM*, 2014.

[11] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks." *Mobile Computing and Communications Review*, vol. 7, no. 3, 2003.

[12] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem." in *Proc. of SIGCOMM*, 2007.

[13] K. Lee, Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong, "Max-Contribution: On optimal resource allocation in delay tolerant networks." in *Proc. of INFOCOM*, 2010.

[14] W. Gao and G. Cao, "On exploiting transient contact patterns for data forwarding in delay tolerant networks." in *Proc. of ICNP*, 2010.

[15] F. Li and J. Wu, "MOPS: Providing content-based service in disruption-tolerant networks," in *Proc. of ICDCS*, 2009.

[16] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proc. of MobiHoc*, 2008.

[17] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *Proc. of MobiHoc*, 2007.

[18] J. Wu, M. Xiao, and L. Huang, "Homing spread: Community home-based multi-copy routing in mobile social network," in *Proc. of INFOCOM*, 2013.

[19] X. Zhang and G. Cao, "Transient community detection and its application to data forwarding in delay tolerant networks." in *Proc. of ICNP*, 2013.

[20] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proc. of MOBICOM*, 2004.

[21] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Enhancing interactive web applications in hybrid networks," in *Proc. of MOBICOM*, 2008.

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, 1959.

[23] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: an efficient routing in disruption-tolerant networks." in *Proc. of MobiHoc*, 2009.

[24] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *MobiHoc*, 2007.