DTN-FLOW: Inter-Landmark Data Flow for High-Throughput Routing in DTNs

Kang Chen, *Student Member, IEEE*, and Haiying Shen, *Senior Member, IEEE, Member, ACM*

***Abstract*—In this paper, we focus on the efficient routing of data among different areas in delay tolerant networks (DTNs). In current algorithms, packets are forwarded gradually through nodes with higher probability of visiting the destination node or area. However, the number of such nodes usually is limited, leading to insufficient throughput performance. To solve this problem, we propose an inter-landmark data routing algorithm, namely DTN-FLOW. It selects popular places that nodes visit frequently as landmarks and divides the entire DTN area into subareas represented by landmarks. Nodes transiting between landmarks relay packets among landmarks, even though they rarely visit the destinations of these packets. Specifically, the number of node transits between two landmarks is measured to represent the forwarding capacity between them, based on which routing tables are built on each landmark to guide packet routing. Each node predicts its transits based on its previous landmark visiting records using the order-*k* Markov predictor. When routing a packet, the landmark determines the next-hop landmark based on its routing table and forwards the packet to the node with the highest probability of transiting to the selected landmark. Thus, DTN-FLOW fully utilizes all node movements to route packets along landmark-based paths to their destinations. We analyzed two real DTN traces to support the design of DTN-FLOW. We deployed a small DTN-FLOW system on our campus for performance evaluation. We also proposed advanced extensions to improve its efficiency and stability. The real deployment and trace-driven simulation demonstrate the high efficiency of DTN-FLOW in comparison to state-of-the-art DTN routing algorithms.**

***Index Terms*—Delay tolerant networks, inter-landmark, routing.**

I. Introduction

DELAY tolerant networks (DTNs) are featured by intermittent connection and frequent network partition. Thus, DTN routing is usually realized in a carry-store-forward manner [1], which makes it possible to develop useful applications over such challenging environments. Among many applications, we are particularly interested in those that exchange data among or collect data from different areas because DTNs usually exist in areas without infrastructure networks and thereby are good mediums to realize data communication among these areas. For example, researchers have proposed to provide data communications (i.e., Internet access) to remote and rural areas [2] by relying on people or vehicles moving among rural villages and cities to carry and forward data. The concept of DTN has also been applied in animal tracking [3], which collects logged data from the digital collars attached to zebras in Kenya without infrastructure network.

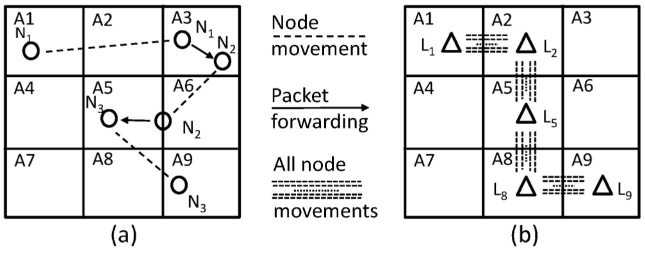


Fig. 1. Comparison between (a) previous methods and (b) DTN-FLOW.

Since these applications tend to transmit high volumes of data (i.e., Internet data and collected logs), a key hurdle in developing these applications is efficient data routing with high throughput. The *throughput* refers to the amount of packets that can be routed between two areas within their time-to-live (TTL) specifications in a unit time period. The work in [4] has shown that node mobility can improve the throughput between source-to-destination pairs in mobile ad hoc networks by reducing the number of forwarding hops and incurred interferences among wireless links. Node mobility is also the key for high-throughput DTN routing. This is because nodes are sparsely distributed in a DTN. Then, packets heavily rely on the contact opportunities created by node mobility to be forwarded to their destinations. Therefore, in this paper, we focus on how to better utilize node mobility to increase throughput, i.e., transmit as many as possible packets from one area to another area within their TTL specifications.

This challenge is nontrivial due to the features of DTNs mentioned previously. State-of-art DTN routing algorithms [5]–[20] exploit either past encounter records (probabilistic routing), social network properties (social network routing), or past moving paths (location-based routing) to deduce a node's probability of reaching a certain node or area, and forward packets to nodes with higher probability than current packet holder. Fig. 1(a) illustrates the routing process in previous routing algorithms. A packet is generated in area $A1$ for area $A9$. It is first carried by node N_1 and then forwarded to N_2 in area $A3$ since N_2 visits $A9$

Manuscript received April 17, 2013; revised August 14, 2013 and October 27, 2013; accepted November 30, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor X. Lin. This work was supported in part by the US NSF under Grants IIS-1354123, CNS-1254006, CNS-1249603, CNS-1049947, CNS-0917056, and CNS-1025652 and Microsoft Research Faculty Fellowship 8300751. An early version of this work was presented in the Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Boston, MA, USA, May 20–24, 2013.

The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29631 USA (e-mail: kangc@clemson.edu; shenh@clemson.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2013.2296751

more frequently. Later, similarly, the packet is forwarded to N_3, which finally carries the packet to area $A9$. Since the number of nodes with high probability of visiting the destination usually is limited, by only relying on such nodes, previous routing algorithms fail to fully utilize all node movements, leading to degraded overall throughput. For example, even if there are many nodes moving between $A2$ and $A5$, they are not utilized to forward the packet.

To deal with this problem, we propose an inter-landmark data flow routing algorithm, called DTN-FLOW, that fully utilizes all node movements in DTNs. Fig. 1(b) demonstrates DTN-FLOW. We assume that there is a popular place in each of the nine subareas in Fig. 1(a). DTN-FLOW then determines landmarks from these popular places and adopts the same subarea division as in Fig. 1(a). Each subarea is represented by one landmark. Each landmark is configured with a central station, which is an additional infrastructure with high processing and storage capacity. Then, node movement can be regarded as transits from one landmark to another landmark. DTN-FLOW utilizes such transits to forward packets one landmark by one landmark to reach their destination areas. Nodes transiting between landmarks relay packets, even though they rarely or even may not visit the destinations of the relayed packets. We denote the landmark in each area in Fig. 1(b) by L_i $(i \in [1,9])$. For packets originated from area $A1$ targeting area $A9$, they are forwarded along landmark L_1, L_2, L_5, L_8 to finally reach L_9. Thus, DTN-FLOW fully utilizes the node transits between different pairs of landmarks for data transmission (i.e., $A2$ and $A5$ in previous example), thereby increasing the data flow throughput.

DTN-FLOW measures the amount of nodes moving from one landmark, say L_i, to another landmark, say L_j, to represent the inter-landmark forwarding capacity from L_i to L_j. This capacity indicates the data transfer capacity between landmarks, hence is similar to the concept of "bandwidth" for physical wired or wireless links. With the measured capacity, each landmark uses the distance-vector method [21] to build its routing table that indicates the next-hop landmark to reach each destination landmark. DTN-FLOW predicts node transits based on their previous landmark visiting records using the order-k Markov predictor. Then, in packet routing, each landmark determines the next-hop landmark based on its routing table for each packet and forwards the packet to the node with the highest probability of transiting to the selected landmark. Thus, DTN-FLOW fully utilizes the node transits to forward packets along landmark paths with the shortest latency to reach their destinations. In DTN-FLOW, the transmission of all information (i.e., routing tables and packets) among landmarks is conducted through mobile nodes.

Therefore, the proposed DTN-FLOW is suitable for applications designed to transfer data among different areas in DTNs with skewed node visiting preferences. Two common scenarios that satisfy the above requirements are the data exchange between different buildings on campus and between different villages in rural areas. In both scenarios, mobile nodes (people) usually frequently visit different buildings/villages with energy supply.

We analyzed two real DTN traces to confirm the shortcoming of the current routing algorithms and to support the design of DTN-FLOW. We also proposed extensions on dead-end prevention, routing loop detection and correction, and load balance to improve the efficiency and stability of DTN-FLOW. We further deployed a real DTN-FLOW system on our campus using nine mobile phones. This real deployment and extensive trace-driven simulation demonstrate the high throughput and high efficiency of DTN-FLOW in comparison to state-of-the-art routing methods in DTNs.

The remainder of this paper is arranged as follows. Sections II and III present related work and network model and real trace analysis. Section IV introduces the detailed design of the DTN-FLOW system. In Section V, the performance of DTN-FLOW is evaluated through extensive trace-driven experiments and a real deployment. Section VI concludes this paper with remarks on our future work.

II. Related Work

A. *Probabilistic Routing Methods*

Probabilistic routing methods [5]–[8] use nodes' past encounter records to predict their future encounter probabilities, which is used to rank the suitability of a node to carry a packet. PROPHET [5] updates the encountering probability between two nodes when they meet and ages the probability over time. A packet is always forwarded to nodes with higher probability of meeting its destination. MaxProp [6], RAPID [7], and MaxContribution [8] extend PROPHET by further specifying forwarding and storing priorities based on the probability of successful delivery. Packets with higher priorities are forwarded first, and high-priority packets replace low-priorityy packets when a node's storage is full.

B. *Social-Network-Based Routing Methods*

Considering that people carrying mobile devices usually belong to certain social relationships, social-network-based routing algorithms [9]–[14] exploit social network properties in DTNs for packet routing. MOPS [9] is a publish–subscribe system. It groups frequently encountered nodes into a cluster for efficient intracommunity communication and selects nodes having frequent contacts with foreign communities for intercommunity communication. BUBBLE [10] uses two layers of ranks: global and local. The global ranking is used to forward a packet to the destination community, and the local ranking helps to find the destination within the community. SimBet [11] adopts centrality and similarity to rank the suitability of a node to carry a packet. It is based on the concept that nodes having high centrality and similarity with the destination node tend to meet it frequently. The event dissemination system in [12] is similar to MOPS. It groups well-connected nodes into communities and selects nodes with the highest closeness centrality as brokers for intercommunity dissemination. Costa *et al.* proposed a social-network-based publish–subscribe system [13]. It forwards messages to nodes that meet subscribers of the packet's interest category frequently and have high connectivity with other nodes. HiBop [14] defines node context by jointly considering various information, including personal interests, residence, and work, and forwards packets to the nodes that have frequent encounter records with the context of the destination.

C. Location-Based Routing Methods

Location-based routing methods [15]–[20] use previous geographical location to assist packet routing in DTNs. GeoDTN [15] encodes historical geographical movement information in a vector to predict the possibility of two nodes becoming neighbors. Then, packets are forwarded to nodes that are more likely to be a neighbor of the destination node. PGR [16] uses observed nodes' mobility pattern to predict nodes future movement to forward packets to a certain geographical destination. GeoOpps [17] exploits the navigation system to calculate the minimal estimated time of delivery (METD) by considering the closest point of possible routes to the destination and forwards packets to vehicles that lead to smaller METD. In MobyPoints [18], a node's meeting probabilities with all possible locations are encoded in vectors. Then, forwarding decisions are made based on the similarity score between the vectors of relay node and destination node. In GeoComm [19], the geocentrality of each geocommunity is calculated based on its contact probabilities with each node. Such centralities are then exploited to realize efficient packet dissemination in DTNs. In PER [20], a node's past transits among landmarks or sojourn on landmarks are summarized to predict its probability of visiting a landmark within a time limit. Such information is further exploited to deduce two nodes' future contact probability for packet routing.

LOUVRE [22] is a similar work with DTN-FLOW. It builds landmarks on road intersections and uses the landmark overlay for routing in vehicle networks. However, LOUVRE focuses on vehicular networks in which GPS and map are used to determine the connected landmark and the next landmark the node is moving toward. On the contrary, DTN-FLOW is designed for general DTNs, in which it is hard to know the next landmark a node is moving to since nodes move freely in the whole area. To solve this problem, DTN-FLOW adopts a k-order Markov predictor and a novel method to handle prediction errors, as shown in Sections IV-B and IV-D.

D. Highlights of DTN-FLOW

The major highlight of DTN-FLOW compared to all the above three categories of methods is a new type of routing architecture. In the previous methods, a packet is always individually forwarded to nodes that have higher probabilities to meet its destination, as shown in Fig. 1(a). However, the number of nodes that frequently meet a destination may be limited, which would constrain the routing throughput. On the contrary, DTN-FLOW does not only rely on nodes that frequently visit a packet's destination to forward the packet. It splits the whole area into subareas and forwards packets one subarea by one subarea to reach the final destination, as shown in Fig. 1(b). In this way, nodes that seldom visit the destination but frequently transit between subareas in the middle of a routing path can also help forward the packet to approach its destination. Consequently, all node mobility can be better utilized to realize efficient data forwarding among different areas in DTNs.

III. Network Model and Trace Analysis

A. Network Model

1) Network Description: We assume a DTN with mobile nodes denoted by N_i. Each node has limited storage space and communication range. We select landmarks, denoted by L_i $(i = 1, 2, 3, \cdots, M)$, from places that nodes visit frequently. Then, the entire DTN area is split into subareas based on landmarks, each of which is represented by a landmark. We configure a central station at each landmark, which has higher processing and storage capacity than mobile nodes and can cover its whole subarea. As in other social-network-based DTN routing algorithms [9]–[14], DTN-FLOW also assumes the existence of social network structure in DTNs. Such social structures determine node movement, leading to reappearing visiting patterns to these landmarks.

A transit means a node moves from one landmark to another landmark. We denote the *transit link* from landmark L_i to landmark L_j as T_{ij}. For a transit link, say T_{ij}, we define the *bandwidth* as the average number of nodes transiting from L_i to L_j in a unit time (T), denoted by B_{ij}. For simplicity, we assume that each packet has a fixed size. Our work can be easily adapted to the case when packets have different sizes by dividing a large packet into a number of the same-size segments.

2) Packet Routing: In this paper, we focus on routing packets to landmarks/subareas. The extension of DTN-FLOW to forwarding packets to mobile nodes will be discussed in Section IV-E.4. As most previous routing algorithms, we consider a single-copy packet forwarding scenario in routing in the network.

3) Differences With Infrastructure Networks: Though DTN-FLOW presents similar overlay as the infrastructure network (i.e., subareas covered by landmarks), they have significant differences. First, DTN-FLOW does not require landmarks to be interconnected with fixed links. Rather, landmarks rely on the mobile nodes moving between them to relay packets. Second, as shown later, a landmark only functions as a special relay node in the packet routing. Therefore, landmarks do not bring about server–client structure but keep the *ad hoc* nature of DTNs. Unlike base stations, landmarks are just static nodes in DTNs with higher processing capacity.

4) Purpose of Landmarks: In DTN-FLOW, landmarks function as "routers" in the network. Each landmark decides the neighbor landmark to forward its received packets. Neighbor landmarks are connected by "links" that take mobile nodes as the transfer media to carry packets. Without landmarks, packets are relayed purely through mobile nodes when they meet with each other, which may suffer from the uncertainty of node mobility. Therefore, landmarks make the DTN routing more structured (i.e., along landmarks) in the network dynamism in DTNs. In summary, the use of landmarks in DTN-FLOW can better utilize node mobility in DTNs for efficient packet routing with a low extra cost.

B. Trace Analysis

In order to better understand how nodes transit among different landmarks in DTNs, we analyzed two real DTN traces collected from two different scenarios: students on campus and buses in the downtown area of a college town.

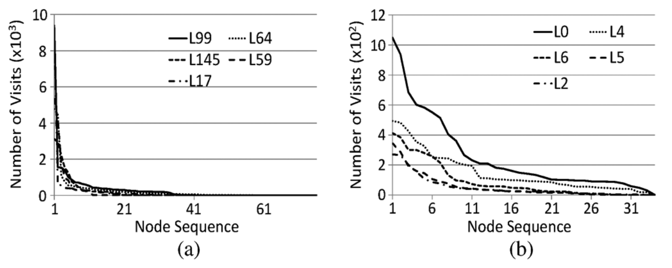


Fig. 2. Visiting distribution of top five most visited landmarks. (a) DART. (b) DNET.

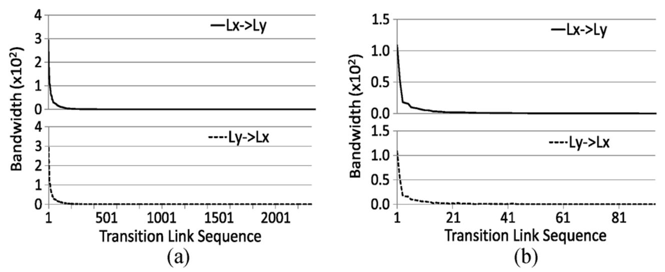


Fig. 3. Bandwidth distribution of transit links. (a) DART. (b) DNET.

TABLE I
CHARACTERISTICS OF MOBILITY TRACES

	DART Campus (DART)	DieselNet AP (DNET)
# Nodes	320	34
# Landmarks	159	18
Duration	119 days	20 days
# Transits	477803	25193
# Transits per day	2401	1257

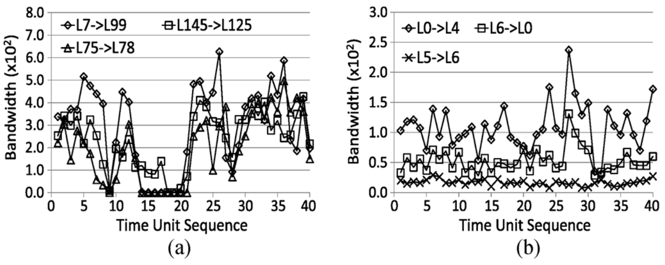


Fig. 4. Transit distribution of top three highest-bandwidth transit links. (a) DART. (b) DNET.

1) Empirical Datasets: Dartmouth Campus Trace (DART) [23]: DART recorded the WLAN access point (AP) association with digital devices carried by students in the Dartmouth campus between November 2, 2003 and February 28, 2004. We preprocessed the trace to fit our investigation. We regarded each building as a landmark and merged neighboring records referring to the same node (mobile device) and the same landmark. We also removed short connections ($<$200 s) and nodes with fewer records ($<$500). Finally, we obtained 320 nodes and 159 landmarks.

DieselNet AP Trace (DNET) [24]: DNET collected the AP association records from 34 buses in UMass Transit from October 22 to November 16, 2007, in Amherst, MA, USA. Each bus carried a Diesel Brick that constantly scanned the surrounding area for open AP connections and a GPS to record its GPS coordinators. Since there are many APs in the outdoor testing environment, some of which are not from the experiment, we removed APs that did not appear frequently ($<$50) from the trace. We mapped APs that are within a certain distance ($<$1.5 km) into one landmark. Similar to the processing of the DART trace, neighboring records referring to the same node (bus) and the same landmark were merged. Finally, we obtained 34 nodes and 18 landmarks.

The key characteristics of the two traces are summarized in Table I. We then measured the landmark visiting distribution and the transits of mobile nodes among landmarks.

2) Landmark Visiting Distribution: We first measured how landmarks are visited by mobile nodes. Due to page limit, we only show the visiting distribution of the five most visited landmarks in the two traces in Fig. 2(a) and (b), respectively. We see that in both traces, for each of the top five landmarks, only a small portion of nodes visits it frequently. For example, in the DART trace, less than 15 out of 320 nodes visit landmarks frequently. Though not shown in the figures, such a finding holds for almost all landmarks in the two traces. Thus, we obtain the first observation (O).

O1: For each subarea, only a small portion of nodes visit it frequently.

This observation matches our daily experience that a department building on a campus usually is mainly visited by students in the department, and a bus station may only be visited by buses that stop at it. Such a finding validates our claim in Section I that the number of nodes frequently visiting the destination area is limited, which leads to degraded throughput in previous routing algorithms that only rely on such nodes for packet forwarding.

3) Transits Among Landmarks: We refer to two transit links containing the same landmarks but have different directions (e.g., T_{ij} and T_{ji}) as *matching transit links*. We then measured the bandwidths of all transit links in the two traces and ordered them in decreasing order. We label two matching transit links with the same sequence number and plot them in two separate subfigures, as shown in Fig. 3(a) and (b). Transit links with 0 bandwidth were omitted. From the two subfigures, we make the following two observations.

O2: A small portion of transit links have high bandwidth.

O3: The matching transit links are symmetric in bandwidth.

We also measured the bandwidth of all transit links in the two traces along time. The time unit was set to 3 days and 0.5 day in the two traces, respectively. This results in a total of 40 time units for both traces. Due to page limit, we only present the results of the three highest-bandwidth transit links in Fig. 4(a) and (b). Fig. 4(a) shows that except for two time periods [7, 10] and [14, 21], the measured bandwidth of each transit link fluctuates around its average value slightly. We checked the calendar and found that the two periods are the Thanksgiving and Christmas holidays, which means that few students moved around on the campus. In Fig. 4(b), we see that the measured bandwidth of each transit link is more stable around its average bandwidth than in the DART trace. This is because: 1) the DNET trace excludes holidays and weekends; and 2) bus mobility is more repetitive over time than human mobility. Also, both figures show that though there are some fluctuations, the bandwidth relationship of the three transit links remains stable. We then derive the following observation.

O4: The bandwidth of a transit link measured in a certain time period can reflect its overall bandwidth.

IV. System Design

In this section, we introduce the detailed architecture of our DTN-FLOW system based on above observations. It has four main components: 1) landmark selection and subarea division; 2) node transit prediction; 3) routing table construction; and 4) packet routing algorithm. Component 1 provides general guidelines to select the locations of landmarks and split the DTN into subareas. Component 2 predicts the next landmark a node is going to visit based on its previous visiting records. Such predictions are used to forward packets and exchange routing tables among landmarks. Component 3 measures the data transfer capacity between each pair of landmarks, based on which the routing table is built to indicate the next-hop landmark for each destination landmark and associated estimated delay. With the support of the first two components, component 4 determines the next-hop landmark and the forwarding node in packet routing.

A. Landmark Selection and Subarea Division

The landmark selection determines the places to install landmarks. Subarea division assigns each landmark a subarea. Both landmark selection and subarea division are conducted by the network administrator or planner who hopes to utilize the DTN for a certain application.

1) Landmark Selection: As aforementioned, we select popular places that are frequently visited by mobile nodes as landmarks. To identify popular places, a simple way is to collect node visiting history and take the top N_v most frequently visited places as popular places. Popular places in DTNs with social network structures can also be predetermined based on node mobility pattern. For example, in the DART network, we can easily find popular buildings that students visit frequently: library, department buildings, and dorms. In DTNs in rural areas, villages are naturally popular places. In the DTNs using animals as mobile nodes for environment monitoring in mountain areas, places with water/food are frequently visited.

The resulted popular places form a candidate landmark list. There may be several popular places in a small area. Thus, not every popular place needs to be a landmark. Then, for every two candidate landmarks with distance less than D_v meters, the one with less visit frequency is removed from the candidate list. Finally, the distance between every two candidate landmarks is larger than D_v meters.

2) Subarea Division: With the landmarks, we split the entire network into subareas. Since the subarea division only serves the purpose of routing among landmarks, we do not need a method to precisely define the size of each subarea. Therefore, we follow the rules below to generate subareas.

- Each subarea contains only one landmark.
- The area between two landmarks is evenly split to the two subareas containing the two landmarks.
- There is no overlap among subareas.

Note that the split of area between landmarks does not affect how nodes move between landmarks. Nodes can transit among landmarks through any routes. Fig. 5 gives an example of the subarea division in our campus deployment of DTN-FLOW, which is introduced in Section V-C.

3) Influence of Landmark Selection and Subarea Division: In the above algorithms, N_v and D_v determine the number

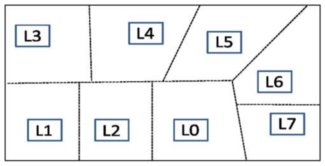


Fig. 5. Subarea division in our campus deployment.

of landmarks and the subarea sizes. With more landmarks, a node's transits between landmarks may present higher diversity and may not exhibit a stable pattern for prediction, thereby degrading the routing performance. The maintenance cost of landmarks also increases in this case. With fewer landmarks, the average subarea size increases, which makes it difficult to provide fine-grained destinations. Therefore, the values of N_v and D_v should be determined so that necessary popular places are represented by landmarks and the patterns of node transits between landmarks can be stably summarized. This objective can be achieved by simply following the above landmark selection and subarea division process. Recall that landmarks are selected from popular places. Then, a resultant large subarea with a single landmark is caused by the fact that there are no other popular places in this area. This means even if we place extra landmarks in this area, packets cannot quickly reach them since they are in unpopular places with few node visits. Therefore, N_v and D_v are determined by the popularity of areas, i.e., the node mobility patterns. In our real trace driven test in Section V, the step of subarea division is completed quickly based on node mobility patterns.

4) Real-World Scenarios and Limitations: The above landmark selection and subarea division procedures require certain administration input. However, as previously introduced, this step is quite intuitive and requires slight effort. With the design of landmarks, we can see that DTN-FLOW is suitable for DTNs with distributed popular places. In a real-world DTN, popular places usually are distributed over an area. For example, mobile device carriers (i.e., people or animals) usually belong to certain social structures and have skewed and repeated visiting patterns [18]. Therefore, the proposed DTN-FLOW is applicable to most realistic DTN scenarios.

5) Cost of Landmarks: As mentioned previously, a landmark can be regarded as a static autonomous node. Each landmark only needs to communicate with nodes in its subarea. Therefore, landmarks do not need network connection or to be interconnected. This means that the import of landmarks only needs to build some fixed nodes in the network. Although landmarks require higher capacity in storage and computing compared to normal mobile nodes, these requirements can be easily satisfied on fixed nodes. It is also easy to maintain landmarks. When a landmark malfunctions, we can simply replace it without network-level reconfiguration or merge its subarea with that of a neighboring landmark. Since the number of landmarks often is very limited, the total cost is limited. In summary, the cost of landmark deployment is acceptable, especially when considering the significant improvement on the routing efficiency and the reduction of the overhead on the mobile nodes as shown in Section V.

TABLE II
LANDMARK VISITING HISTORY TABLE ON A NODE

Landmark ID	Start time (s)	End time (s)
8	8500	9000
1	7100	8450
7	2000	7000
...	...	...

B. Node Transit Prediction

Since DTN-FLOW relies on node transit for packet forwarding, accurate prediction of node transit is a key component. DTN-FLOW predicts each node's next transit by maintaining a landmark visiting history table on each node, as shown in Table II. The "Start time" and "End time" denote the time when a node connects and disconnects to the central station in the corresponding landmark, respectively. Note that the "End time" in a previous landmark may differ with the "Start time" in a current landmark since a node may not always connect to a landmark during its movement.

1) Order-k Markov Predictor: To predict node transits among landmarks, we adopt the order-k ($O(k)$, $k = 0, 1, 2, \cdots$) Markov predictor [25], which assumes that the next transit is only related to the past k transits. A node's landmark transit history can be represented by $T_H = T_{x_1,x_2}T_{x_2,x_3}\ldots T_{x_{j-1},x_j}\ldots T_{x_{n-1},x_n}$, in which T_{x_{j-1},x_j} ($x_{j-1} \neq x_j$ and $x_{j-1}, x_j \in [1, M]$) represents a transit from $L_{x_{j-1}}$ to L_{x_j}. We let $X(n-k,n) = T_{x_{n-k},x_{n-k+1}} \cdots T_{x_{n-1},x_n}$ represent the past k consecutive transits. When $k = 0$, $X(n-k,n) = T_{x_n,x_n}$, representing the visiting of landmark L_{x_n}. Then, the probability for each possible next transit $T_{x_n,x_{n+1}}$ of a node is calculated by

$$\Pr\left(T_{x_n,x_{n+1}} | X(n-k,n)\right) = \frac{\Pr\left(X(n-k,n), T_{x_n,x_{n+1}}\right)}{\Pr\left(X(n-k,n)\right)} \tag{1}$$

where

$$\Pr\left(X(n-k,n), T_{x_n,x_{n+1}}\right) = \Pr\left(X(n-k,n+1)\right) \tag{2}$$

and

$$\Pr\left(X(n-k,n)\right) = \frac{N\left(X(n-k,n)\right)}{N(All_k)} \tag{3}$$

where $N(X(\cdot))$ and $N(All_k)$ denote the number of $X(\cdot)$ and k consecutive transits in T_H, respectively. Note that $N(All_0)$ denotes the total number of landmark visits of the node. Then, the transit that leads to the maximal probability based on (1) is selected as the predicted transit. For example, suppose we use an order-1 Markov predictor on a system with five landmarks ($M = 5$), and the landmark transit history of a node is $T_H = T_{0,1}T_{1,3}T_{3,4}T_{4,2}T_{2,0}T_{0,1}$. Then, based on (1), the probability for each possible next landmark L_a ($a \in [1,5]$) is calculated as $\Pr(T_{0,1}T_{1,a})/\Pr(T_{0,1})$. Based on (3), $\Pr(T_{0,1}T_{1,3}) = 1/5$ since $T_{0,1}T_{1,3}$ appears once in T_H and the total number of two consecutive transits is 5. Similarly, $\Pr(T_{0,1}) = 2/6$. Then, the transit probability is 0.6.

2) Determination of k: In general, the prediction accuracy of the order-k Markov increases as k increases until a certain value due to insufficient position records [26]. The order-k Markov predictor actually exploits the $(k+1)$-hop transit pattern for prediction. Therefore, when k increases, more information is considered to classify node mobility, thus increasing the prediction accuracy. However, when k increases, the possibility that at least one transit in a $(k+1)$-hop transit pattern cannot be collected (i.e., a missed $(k+1)$-hop transit pattern) also increases. When k increases to a large value, too many $(k+1)$-hop transit patterns may be missed, leading to a low prediction accuracy. In other words, the completeness of the collected position information affects the k that can lead to the highest prediction accuracy. Therefore, the administrator needs to first collect nodes' historical visiting records and then check which k can lead to the highest prediction accuracy. The identified k then can be used for future prediction.

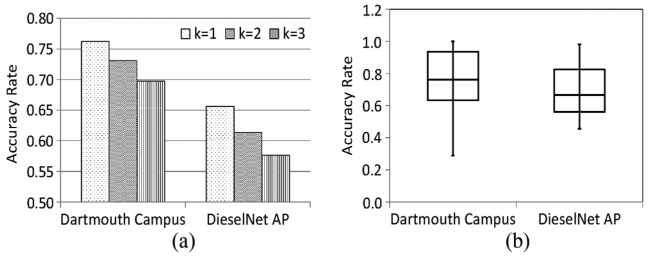


Fig. 6. Accuracy of the transit prediction. (a) Average prediction accuracy of order-k predictor. (b) Minimal, first quantile, average, third quantile, and maximal of the accuracy of the order-1 predictor.

3) Prediction Accuracy With the Two Traces: We applied the order-k Markov predictor to both the DART and the DNET traces with k equal to 1, 2, and 3 to check which k can lead to the best prediction accuracy. We calculated the *accuracy rate* of each node as the number of correct predictions over the number of predictions. The average accuracy rates of all nodes with different k are shown in Fig. 6(a). We see that $k = 1$ leads to the highest prediction accuracy. This is because many position records are absent in the two traces. In DART trace, a student's device cannot be logged unless he/she is using the device. In DNET trace, the APs are roadside APs owned by others and are not dedicated for the experiment, which means they may not appear constantly in the trace, leading to missing records. Based on such a result, we use the order-1 Markov predictor in the experiments in this paper.

We further show the minimal, first quantile, average, third quantile, and maximal of the accuracy rates of all nodes with the order-1 Markov predictor in Fig. 6(b). We see that in the DART trace, the accuracy rates of over 75% of nodes are higher than 64%, and the average accuracy rate of all nodes is about 77%. In the DNET trace, the accuracy rates of over 75% of nodes are higher than 59%, and the average accuracy rate of all nodes is about 66%. It is intriguing to see that the prediction accuracy in the bus network in DNET, which should have more repetitive moving patterns, is lower than that in the student network on campus in DART. We believe this is caused by the reason that we only predict one AP for the next transit, while a bus may associate with one of several neighboring APs after each transit in the trace. Though certain inaccurate predictions exist, the routing efficiency can be ensured with a method that will be explained in Section IV-D.

C. Routing Table Construction

In DTN-FLOW, each landmark dynamically measures the bandwidths of its transit links to each neighbor landmark.

TABLE III
BANDWIDTH TABLE ON A NODE

Landmark ID	Measured Bandwidth	Time Unit Sequence
2	20	9
6	6	9
1	15	9
...	...	...

The bandwidth of a transit link represents the expected delay of forwarding data through it. Based on the estimated delay, each landmark uses the distance-vector method [21] to build a routing table indicating the next-hop landmark for each destination landmark. Each landmark periodically transfers its routing table to its neighbor landmarks for routing table update. This step is realized through mobile nodes, i.e., a landmark, say L_i, chooses its node with the highest predicted probability of visiting L_j to forward its routing table to L_j. The detailed processes are introduced as follows.

1) Transit Link Bandwidth Measurement: Each landmark maintains a bandwidth table as shown in Table III to record the bandwidth from it to each of its neighbor landmarks. We let N_{ij}^t denote the number of nodes that have moved from L_i to L_j in the tth time unit. Each landmark, say L_i, periodically updates its bandwidth to landmark L_j by

$$B_{ij_{\text{new}}} = \alpha B_{ij_{\text{old}}} + (1-\alpha)N_{ij}^t \quad (4)$$

in which $B_{ij_{\text{new}}}$ and $B_{ij_{\text{old}}}$ represent the updated and previous bandwidth, respectively, and α is a weight factor.

It is easy for landmark L_i to calculate N_{ji}^t since mobile nodes moving to L_i can report their previous landmarks to L_i. However, it is difficult for L_i to calculate N_{ij}^t because after a mobile node moves from L_i to L_j, it cannot communicate with L_i. Recall that *O3* indicates that two matching transit links are symmetric in bandwidths. In this case, L_i can regard $N_{ij}^t \approx N_{ji}^t$ and calculate $B_{ij_{\text{new}}}$ using (4).

However, the symmetric property does not always hold true. For example, transit links connecting two stations in a one-way road can hardly be symmetric in bandwidth. To solve this problem, L_i relies on L_j to keep track of N_{ij}. When landmark L_j predicts that a node is going to leave it for L_i, it forwards N_{ij} to the node. When L_i receives N_{ij} from L_j, it checks whether the time unit sequence in it is larger than the current one. If yes, it updates its bandwidth to L_j accordingly based on (4). Otherwise, the packet is discarded.

2) Building Routing Tables: With the bandwidth table, each landmark can deduce the expected delay needed to transfer W bytes of data to each of its neighbor landmarks. Recall T denotes the time unit for B_{ij} measurement. Suppose each node has S bytes of memory, then the expected delay for forwarding a packet from L_i to L_j (D_{ij}) is $D_{ij} = (W/B_{ij}S)T$. Then, the routing table on each landmark is initialized with the delays to all neighbor landmarks. Each landmark, L_i, further uses the distance-vector protocol to construct the full routing table (as shown in Table IV) indicating the next-hop for every destination landmark (L_{d}) in the network and the overall delay from L_i to L_d, denoted by $D(L_i, L_{\text{d}})$.

In the distance-vector protocol, each landmark periodically forwards its routing table and associated time unit to all neighbor landmarks through mobile nodes. When a landmark, say L_i, receives the routing table from a neighbor landmark,

TABLE IV
ROUTING TABLE ON ONE NODE

Des. Landmark ID	Next Hop ID	Overall Delay
1	1	7
5	5	3
9	1	18
...	...	...

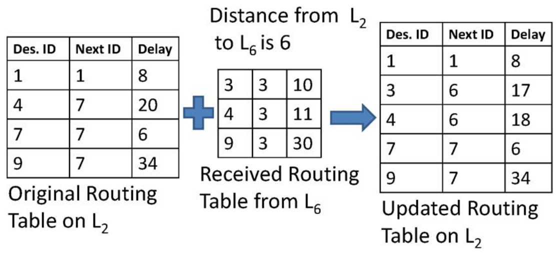

Des. ID | Next ID | Delay
1 | 1 | 8
4 | 7 | 20
7 | 7 | 6
9 | 7 | 34
Original Routing Table on L2
Distance from L2 to L6 is 6
3 | 3 | 10
4 | 3 | 11
9 | 3 | 30
Received Routing Table from L6
Des. ID | Next ID | Delay
1 | 1 | 8
3 | 6 | 17
4 | 6 | 18
7 | 7 | 6
9 | 7 | 34
Updated Routing Table on L2

Fig. 7. Demonstration of the routing table update.

say L_j, it first checks whether it is newer than the previously received one. If not, the table is discarded. Otherwise, the routing table is processed one entry by one entry. For each entry, if the destination landmark L_{d} does not exist in the routing table of L_i, it is added to the routing table by setting the "Next Hop ID" as L_j and the "Overall Delay" as $D_{ij} + D(L_j, L_{\text{d}})$. If L_{d} already exists, it checks whether $D(L_i, L_{\text{d}}) \leq D_{ij} + D(L_j, L_{\text{d}})$. If yes, no change is needed. Otherwise, the "Next Hop ID" is replaced as L_j, and the "Overall Delay" is updated with $D_{ij} + D(L_j, L_{\text{d}})$. This process repeats periodically, and each landmark finally learns the next hop to reach each other destination landmark with the minimum overall delay in its routing table.

For example, suppose the routing table on L_2 originally contains four entries—(1, 1, 8), (4, 7, 20), (7, 7, 6), and (9, 7, 34)—and it receives a routing table from L_6 with three entries: (3, 3, 10), (9, 3, 30), (4, 3, 11), and $D_{26} = 7$. Fig. 7 summarizes the routing table update. In detail, since the routing table has no entry for landmark L_3, entry (3, 6, 17) is inserted directly. Though L_9 already exists in the routing table, $D_{29} = 34$ is less than that of relaying through L_6 (i.e., 37), so no change is needed. L_4 already exists in the routing table, and $D_{24} = 20$ is larger than that of relaying through L_6 (i.e., 18), so the "Next-hop ID" is changed to 6, and the "Overall Delay" is set to 18. The final entries in the routing table are (1, 1, 8), (3, 6, 17), (4, 6, 18), (7, 7, 6), and (9, 7, 34).

3) Routing Table Coverage and Stability: We further measured the average coverage and the average stability of all landmarks' routing tables at 10 evenly distributed observation points in the two traces. A landmark's routing table coverage at the ith observation point is calculated as S_i/N_t, where S_i is the size of its routing table and N_t is the total number of landmarks. A landmark's stability at the ith observation point is calculated by $1 - E_i/N_t$, where E_i is the number of destination landmarks whose next-hop landmarks have changed since the previous observation point.

The measurement results are shown in Fig. 8. We see that after the first several observation points, each routing table can cover most destination landmarks, which demonstrates that the routing table is capable of providing packet routing guidance to

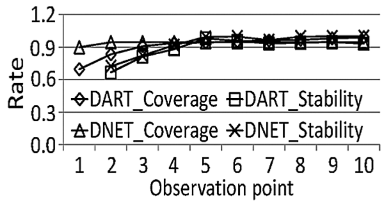


Fig. 8. Average routing table coverage and stability.

any destinations in our design. We also find that each routing table is quite stable after the first several observation points. This is because node movement presents stable pattern in the two traces, leading to stable bandwidth (i.e. delay) on each link. Therefore, the path to each destination landmark is also stable, leading to a stable routing table. This result can be utilized to save the routing table maintenance cost by reducing the routing table update frequency.

D. Packet Forwarding Algorithm

During the packet forwarding, a landmark refers to its routing table to select the next-hop landmark and forwards the packet to the mobile node that has the highest predicted probability to transit to the next-hop landmark. However, as mentioned in Section IV-B, node transit prediction may not always be accurate, which means a node may fail to carry a packet to the landmark indicated in the routing table. Also, there may be nodes that are moving to the packet's destination node directly, which can be utilized to enhance the routing performance. We first introduce our approaches to handle the two issues and then summarize the routing algorithm.

1) Handling Prediction Inaccuracy: To handle the inaccurate transit prediction, DTN-FLOW follows the principle that every forwarding must reduce the routing latency. Thus, when a node moves from L_i to a landmark L_k other than the predicted one L_j, the node checks whether the new landmark still reduces the expected delay to the destination L_d, that is, whether $D(L_k, L_d) < D(L_i, L_d)$. If yes, the node still forwards the packet to landmark L_k for further forwarding. Otherwise, the node holds the packet, waiting for next landmark that has shorter delay to the destination. This design aims to ensure that each transit, though may not be optimal due to node transit prediction inaccuracy, can always improve the probability of successful delivery.

2) Exploiting Direct Delivery Opportunities: Since nodes move opportunistically in a DTN, it is possible that a landmark can discover nodes that are predicted to visit the destination landmarks of some packets. Therefore, when a landmark receives a packet, it first checks whether any connected nodes are predicted to transit to its destination landmark. If yes, the packet is forwarded to the node directly. In case the node fails to forward the packet to its destination landmark, the node uses the scheme described in Section IV-D.1 to decide whether to forward the packet to the new landmark.

3) Routing Algorithm: We present the steps of the routing algorithm as follows.

1) When a node generates a packet for an area, it forwards the packet to the first landmark it meets.
2) When a landmark, say L_i, generates or receives a packet, it first checks whether any nodes are predicted to move to the destination landmark of the packet. If yes, the packet is forwarded to the node with the highest predicted probability along with the expected overall delay, which is used by the carrier node to determine whether to forward the packet to an encountered landmark other than the predicted one.
3) Otherwise, L_i checks its routing table to find the next-hop landmark for the packet and inserts the landmark ID and the expected overall delay into the packet.
4) L_i then checks all connected nodes and forwards the packet to the node that has available memory and has the highest predicted probability to transit to the next-hop landmark indicated by the routing table.
5) When a node moves to the area of a landmark, say L_j, it forwards L_j all packets that target L_j or have less overall delay from L_j to the destination than L_i. After this, it predicts its next transit based on the order-k Markov predictor and informs this to L_j.

4) Refining Transit Node Selection: In the fourth step of packet routing, the node with the highest predicted probability to transit to the next-hop landmark of a packet is selected as its carrier. However, as mentioned in Section IV-B, the prediction may not always be correct. We then integrate the prediction accuracy into the process of carrier selection.

Specifically, when selecting the carrier from L_i to L_j, instead of directly using each node's transit probability from L_i to L_j (i.e., $\Pr(T_{ij})$), we use an overall transit probability, denoted by $Pc(T_{ij})$; $Pc(T_{ij}) = \Pr(T_{ij}) * Ar(L_i)$, in which $Ar(L_i)$ is a node's prediction accuracy at landmark L_i. It denotes the probability that the node actually transits to the predicted landmark to which the node is going to transit. It is initiated as a medium value (e.g., 0.5) and is multiplied by β ($\beta > 1$) and γ ($\gamma < 1$) when a correct and an incorrect prediction occurs, respectively. Finally, the node with the highest overall transit probability is selected as the carrier.

As a result, the selected carrier should have both high transit probability and stable mobility pattern (i.e., high prediction accuracy) and can improve the probability of carrying the packet to the next-hop landmark indicated in the routing table.

5) Communication Scheduling: When a node connects to a landmark, it uses the uplink to upload packets to the landmark. Meanwhile, the landmark utilizes the downlink to forward packets on it to nodes. Both steps follow the packet routing algorithm introduced in Section IV-D.3. We assume that the landmark can only communicate with one node through either the uplink or the downlink at a moment. Though nodes usually are sparsely distributed in DTNs, a few landmarks may be congested in DTN-FLOW. We then design a scheduling algorithm to improve the overall throughput against the congestion.

1) The landmark scans its subarea to discover new nodes every T_{sc}, e.g., 1 min. If found, the landmark allows the new node to use the uplink to register immediately.
2) Other than the scanning period, the landmark decides to use the uplink or the downlink based on the ratio of the number of packets it holds (N_{pl}) to the number of packets on all nodes (N_{pn}): $R_{ud} = N_{pl}/N_{pn}$. When $R_{ud} < N_{sm}$, the landmark switches to packet uploading mode and selects nodes to use the uplink to upload packets. When $R_{ud} \geq N_{lg}$, it switches to packet forwarding mode again. N_{sm} and N_{lg} can be dynamically adjusted based on system needs, e.g., $N_{sm} = 1$ and $N_{lg} = 3$.

3) In the packet uploading mode, the uplink is assigned to the node with the most packets that their expected delays to destination landmarks are lower than their remaining TTLs. The node is allowed to upload at most N_{uu} (e.g., 50) packets each time. Such a process repeats until the landmark switches to the packet forwarding mode.
4) In the packet forwarding mode, the landmark first forwards the packet that: 1) has the minimal remaining TTL; and 2) its expected delay to its destination landmark is smaller than the remaining TTL. Such a process repeats until the landmark switches to the packet uploading mode.

Following this manner, packets that need to be handled first to ensure their successful delivery are assigned higher priority on the landmark, thereby improving the overall throughput.

E. Advanced Algorithm Extension and Discussion

In this section, we further propose three strategies to improve the efficiency and robustness of DTN-FLOW.

1) Dead-End Prevention: As mentioned previously, a node may carry a packet to an "unexpected" landmark, which means that it fails to transit to the predicted landmark but moves to a wrong landmark. In this case, based on our routing algorithm, this node still is responsible for carrying the packet to the next-hop landmark or a suitable landmark. However, this process may lead to a dead end, in which the packet carrier stays in a wrong landmark for a long time. For example, when moving out of landmark L_i, a bus may move to a parking lot or a garage for maintenance. In this case, the packets on the bus have to wait for a long time, leading to a dead end.

We propose a method to detect the dead end based on a node's historical average stay time in landmarks. In this method, each node calculates and stores the average time it stays in each landmark based on its historical movement records. When a node transits to a landmark, say L_e, it checks if a dead end occurs based on following conditions, where H_t is a determination factor and is usually set to a relatively large value to prevent false positives.

- If it has stayed in L_e for H_t times longer than its average stay time in a landmark.
- If it has stayed in L_e for H_t times longer than its average stay time in L_e.

The first condition means that the node encounters a dead end on its regular route, while the second condition means it encounters an abrupt dead end, i.e., unexpected maintenance. When a node observes a dead end when it moves to L_e, rather than keeping its packets, it forwards them to L_e directly. Then, L_e utilizes its routing table to decide the next-hop landmark for these packets and forwards them to the nodes that can carry them out of L_e. Note that in order to reduce false positives, dead-end detection is launched only when a node has accumulated enough historical records to calculate its average stay time in each landmark.

2) Routing Loop Detection and Correction: Recall that we use the distance-vector protocol to build the routing tables on each landmark to indicate the next-hop landmark to each destination landmark. Specifically, landmarks exchange their routing tables periodically or when necessary and update their routing tables accordingly, as explained in Section IV-C.2.

However, due to untimely routing table update, routing loop may happen. Fig. 9(a) and (b) demonstrates an example of the routing loop regarding the destination landmark $A5$. In Fig. 9,

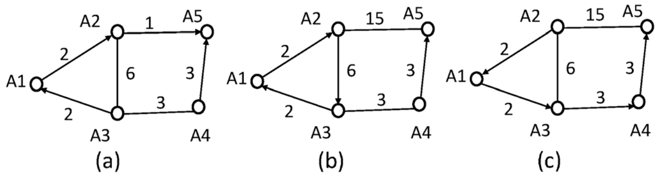


Fig. 9. Demonstration of the routing loop detection and correction. (a) Initial condition. (b) Routing loop. (c) Break the loop.

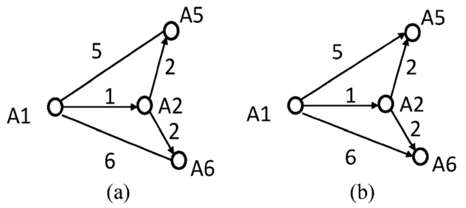


Fig. 10. Demonstration of an overloaded link and solution. (a) Packets from $A1$ destined to $A2$, $A5$, and $A6$ overload link T_{12}. (b) Links T_{15} and T_{16} alleviate the load on link T_{12}.

the number on each link denotes the expected delay of the link. As shown in Fig. 9(a), initially, the next-hop landmark for A_5 in the routing tables on $A1$, $A2$, $A3$, and $A4$ are $A2$, $A5$, $A1$, and $A5$, respectively. Then, as shown in Fig. 9(b), suppose the delay of the link connecting $A2$ and $A5$ changes to 15, and this information is only known by $A2$. Meanwhile, a distance vector from $A3$ arrives at $A2$ claiming $A3$'s estimated delay to $A5$ is 5 (i.e., 2 + 2 + 1) through $A1$. In this case, $A2$ will change the next-hop landmark for $A5$ from $A5$ to $A3$, which leads to a routing loop of $A2 \to A3 \to A1 \to A2$ for packets targeting $A5$.

In order to detect and correct routing loops, we let each packet record the IDs of the landmarks it has visited. When a packet finds that it has visited a landmark twice, it informs this landmark of the existence of a routing loop and the involved landmarks in the loop [e.g., $A1$, $A2$, and $A3$ in Fig. 9(b)]. Then, the landmark generates a loop correction packet, which includes the IDs of the involved landmarks and the destination landmark, and sends it to all involved landmarks. Upon receiving such a correction packet, these landmarks immediately send their updated distance vector on the destination landmark to all neighbor landmarks repeatedly until the next-hop landmark for the destination landmark remains unchanged for a certain period of time T_l. T_l should be large enough so that each landmark in the loop can collect the updated distance vector from all other landmarks in the loop. We then set T_l to the average time for a packet to traverse the loop.

3) Load Balancing: In the above design, DTN-FLOW decides the next-hop landmark by solely considering the delay of the links, i.e., choosing the link that leads to the destination landmark with the minimal expected delay. However, such a strategy may generate overloaded links because a link with a very low expected delay may be included in the optimal routes to multiple destination landmarks. Fig. 10(a) shows such an example, in which the number on each link denotes its expected delay. In the figure, the packets generated on $A1$ destined to $A2$, $A5$, and $A6$ will select the transit link T_{12} since it has a very low expected delay, (i.e., 1), thereby possibly overloading the link T_{12}.

When a link is overloaded, the packets may wait for a long time to be forwarded through the link. They may take much

TABLE V
EXPANDED ROUTING TABLE IN ONE NODE

Des. Landmark	Next Hop	Delay	Backup	Delay
1	1	7	14	20
5	5	3	11	8
9	1	18	7	30
...	...	...	...	...

longer time than the expected delay of the link to pass through the link, thereby degrading the routing efficiency. Actually, when many packets wait for the same link, other links can be utilized to offload the load. For example, in Fig. 10(b), the transit links T_{15} and $T16$ can take some packets from $A1$ destined to $A5$ and $A6$ to reduce the load on link T_{12}.

For this purpose, we first expand the routing table to provide a backup next-hop landmark for each destination landmark, which has the second lowest overall delay to the destination landmark, as shown in Table V. The backup next-hop landmark is updated concurrently with the update of the original routing table, which does not incur additional communication cost. Then, each landmark monitors the incoming rate and outgoing rate for each link. The former is calculated as the average number of received packets that need to be forwarded through the link in a time unit, while the latter is calculated as the average number of packets that are carried by mobile nodes to pass through the link in a time unit. When the incoming packet rate is T_b times larger than the outgoing rate, it means that the number of received packets increases faster than the number of packets being forwarded out through the link, which leads to link overloaded. Then, the landmark forwards the packets to the backup next-hop landmark through another link.

4) Routing Packets to Mobile Nodes: Recall that DTN-FLOW is mainly designed to realize packet routing between different subareas/landmarks. In certain scenarios, it is also desirable to route a packet to a certain mobile node. DTN-FLOW can be adapted to realize this objective. In DTNs, mobile nodes usually have skewed visiting preferences [18], which means that they visit certain landmarks frequently. This property can be utilized to forward a packet to a mobile node efficiently. Nodes can summarize their most frequently visited landmarks and report such information to landmarks in the network. Thus, the sender of a packet destined to a destination node can first learn the destination's frequently visited landmarks and forward/copy the packet to them. Since the destination node visits these landmarks frequently, the packet is unlikely to stay in the landmarks for a long time before being forwarded to the destination. This scheme avoids chasing the mobile nodes continuously or the requirement of knowing the position of the destination node beforehand.

V. PERFORMANCE EVALUATION

We first conducted trace-driven experiments with both the DART and the DNET traces and then evaluated the extensions introduced in Section IV-E. A small DTN-FLOW system is also deployed on our campus.

A. Trace-Driven Experiments

1) Experiment Settings: We used the first 1/4 part of the two traces as the initialization phase, in which nodes construct routing tables. Then, packets were generated at the rate of R_p packets per landmark per day. R_p was set to 500 by default. The destination landmark of each packet is randomly selected. We set the TTL of packets to 20 days in the DART trace and 4 days in the DNET trace. A packet was dropped after TTL. The time unit T for bandwidth evaluation and routing table update was set to 3 days. The size of each packet was set to 1 kB, and each node's memory was set to 2000 kB by default. The memory of the landmark was not limited. We used the order-1 Markov predictor in the experiments. We set the confidence interval to 95%.

We compared DTN-FLOW to five state-of-the-art routing algorithms: SimBet [11], PROPHET [5], PGR [16], GeoComm [19], and PER [20]. They were originally proposed for node-to-node routing or packet dissemination in DTNs. We adapted them to fit landmark-to-landmark routing to make them comparable to DTN-FLOW. We use SimBet to represent the social-network-based routing methods. It combines centrality and similarity to calculate the suitability of a node to carry packets to a given destination landmark. The similarity is derived from the frequency that the node visits the landmark. We use PROPHET to represent the probabilistic routing methods. It simply employs the visiting records with landmarks to calculate the future meeting probability to guide the packet forwarding. PGR, GeoComm, and PER exploit geographical information for DTN routing. PGR uses observed node mobility routes, i.e., a sequence of locations, to check whether the destination landmark is on a node's route. GeoComm measures each node's contact probability per unit time with each geocommunity, i.e., landmark, to guide the packet routing. In PER, a node's past mobility and sojourn among different landmarks are summarized to provide prediction a node's probability to visit a landmark before a certain deadline.

We measured following metrics:

- *Success rate*: the percentage of packets that successfully arrive at their destination landmarks;
- *Average delay*: the average time per successfully delivered packet needed to reach the destination landmark;
- *Forwarding cost*: the number of packet forwarding operations occurred during the experiment;
- *Overall cost*: the total number of packet and routing information forwarding operations during the experiment. Forwarding a routing table or a meeting probability table with m entries is counted as m.

Recall that in DTN-FLOW, landmark deployment is completed offline before the system starts. Thus, DTN-FLOW incurs additional cost for landmark deployment compared to other algorithms. However, this small additional cost can bring about significant improvement on routing efficiency and reduction on forwarding costs of energy-constraint mobile nodes (as shown in the experimental results later on), which is the key advantage of DTN-FLOW.

2) Performance With Different Memory Sizes: We first evaluated the performance of the six methods when the size of memory in each node was varied from 1200 to 3000 kB with a 200-kB increase in each step.

Success Rate: Figs. 11(a) and 12(a) present the success rates of the six methods with the DART and the DNET traces, respectively. We see that when the memory in each node increases, the success rates always follow $\mathrm{DTN-FLOW} > \mathrm{PER} > \mathrm{SimBet} \approx \mathrm{PROPHET} > \mathrm{GeoComm} > \mathrm{PGR}$. DTN-FLOW

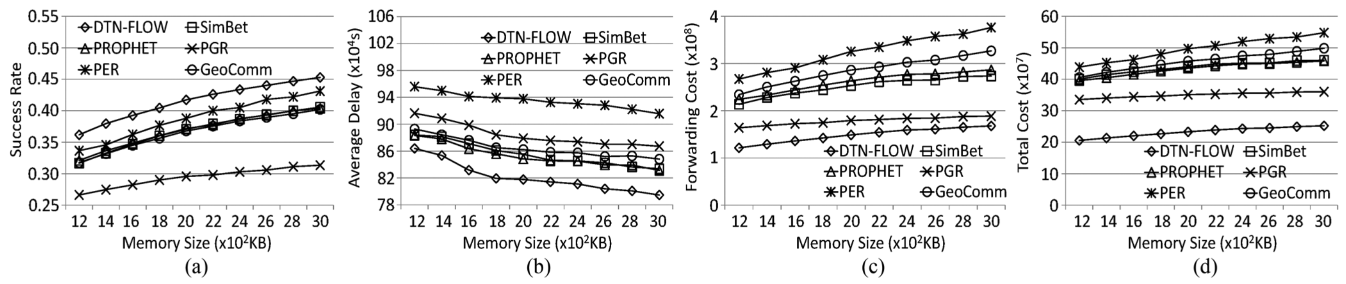


Fig. 11. Performance with different memory sizes using the DART trace. (a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Total cost.

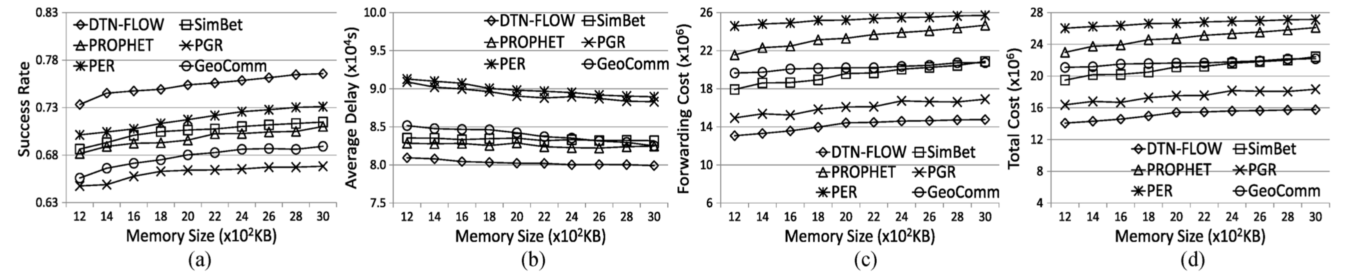


Fig. 12. Performance with different memory sizes using the DNET trace. (a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Total cost.

has the highest success rate because it fully utilizes node movements to forward packets one landmark by one landmark to their destination landmarks, even though some nodes rarely or may not visit these destinations. On the contrary, other methods only rely on nodes that visit destinations frequently for packet forwarding. Limited number of such nodes prevents them from achieving high success rate.

PER leads to the second highest success rate because it considers both transit probability distribution and sojourn time probability distribution to predict a node's probability to move to a landmark within a time limit. SimBet and PROPHET present similar success rates. SimBet exploits social properties, i.e., centrality and similarity, to rank a node's suitability to carry packets to a landmark. PROPHET uses previous encountering records to predict a node's probability of visiting a landmark. Both metrics can indirectly reflect a node's ability to visit a landmark, leading to high and similar success rates.

GeoComm has similar success rate with and lower success rate than SimBet and PROPHET in the tests with DART trace and the DNET trace, respectively. This is because in the DNET trace, each node, i.e., bus, has even contact probability with landmarks on its route since it stays on these landmarks with equal amount of times. Then, the contact probability cannot reflect a node's probability to visit landmarks as accurate as in SimBet and PROPHET, leading to lower success rate.

PGR tries to predict the entire route of a node (with multiple landmarks) for packet forwarding. However, such a prediction has a low accuracy. As shown in Fig. 6(b), the average accuracy of the prediction of only one location is already below 80%. Therefore, PGR has the lowest success rate.

In summary, the experimental results verify the high throughput of DTN-FLOW in transferring data among landmarks with difference memory sizes on each node.

Average Delay: Figs. 11(b) and 12(b) show the average delays of successfully delivered packets in the six methods with the DART and the DNET traces, respectively. We see that the average delays follow $\mathrm{DTN-FLOW} < \mathrm{SimBet} \approx \mathrm{PROPHET} < \mathrm{GeoComm} < \mathrm{PER} < \mathrm{PGR}$. DTN-FLOW has the lowest average delay because the designed routing tables in landmarks guide packets to be forwarded along the fastest paths to their destinations. In SimBet and PROPHET, packets may be generated in or carried to areas where very few nodes move to their destinations regularly. Therefore, packets have to wait for a certain period of time before meeting nodes that visit their destinations frequently, leading to a moderate average delay. Moreover, since nodes with high centrality (i.e., connecting many landmarks) may not visit the specific destination landmark as frequently, SimBet has slightly higher average delay than PROPHET.

For GeoComm, the contact probability between a landmark and a node cannot reflect the node's future probability to visit a landmark as accurate as that in SimBet and PROPHET. Therefore, it has larger average delay than SimBet and PROPHET. PER further has larger delay than GeoComm because it only chooses the node that has the highest probability to visit the destination landmark before a deadline as the forwarder for a packet, rather than the node that can carry the packet to the destination landmark as soon as possible. For PGR, as explained previously, it is difficult to accurately predict long paths with multiple locations, thus leading to inaccurate forwarder selection and the long delay.

These experimental results show the high efficiency of DTN-FLOW in transferring data among landmarks with difference sizes of memory in each node.

Forwarding Cost: Figs. 11(c) and 12(c) plot the forwarding costs of the six methods with the DART and the DNET traces, respectively. We find that the forwarding costs follow $\mathrm{DTN-FLOW} < \mathrm{PGR} < \mathrm{SimBet} < \mathrm{PROPHET} < \mathrm{PER}$ with both traces and GeoComm has the second and the third largest forwarding cost with the DART and the DNET trace, respectively. DTN-FLOW refers to the routing table to forward packets along fastest landmark paths to reach their destinations, which usually takes several forwarding operations.

PGR has the second lowest forwarding cost because nodes tend to show similar ability to visit a landmark. Therefore, a packet holder cannot easily find another node that has higher

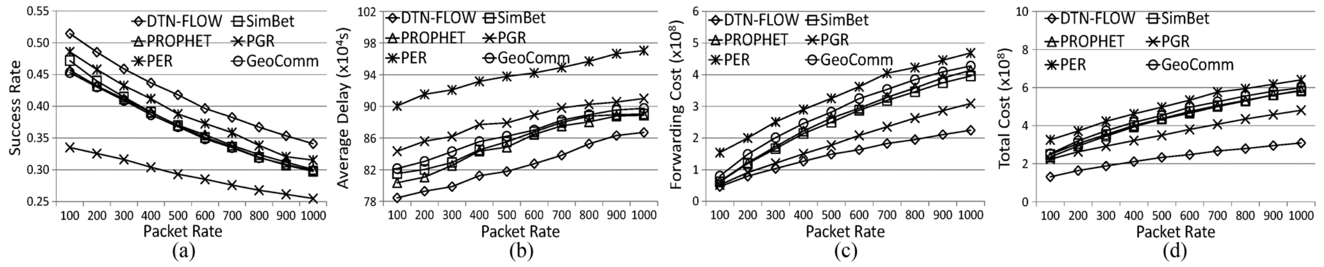


Fig. 13. Performance with different packet rates using the DART trace. (a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Total cost.

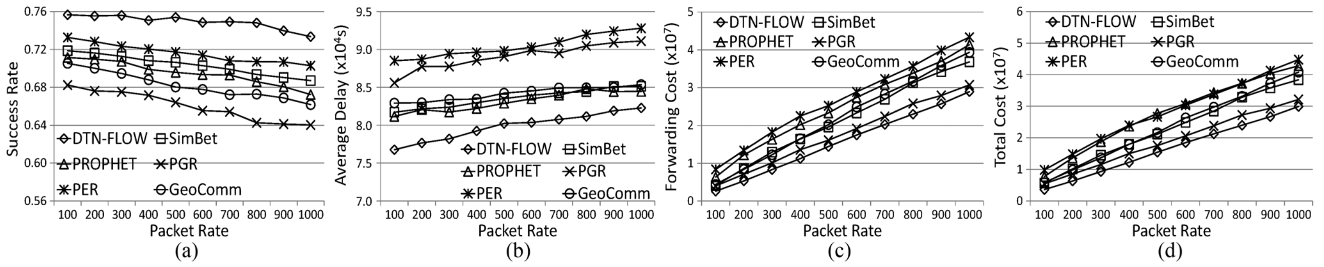


Fig. 14. Performance with different packet rates using the DNET trace. (a) Success rate. (b) Average delay. (c) Forwarding cost. (d) Total cost.

probability of meeting the destination node. Then, packets are not forwarded frequently. However, the low forwarding cost in PGR also results in a low efficiency.

SimBet, PROPHET, GeoComm, and PER use a metric to rank the suitability of nodes for carrying packets and forward packets to high rank nodes. Then, packets are frequently forwarded to nodes with higher suitability, leading to high forwarding cost. More specifically, the easiness of finding a node with higher rank determines the actual forwarding costs of the four methods. In SimBet, since high-centrality nodes usually are limited in the network, packets are gathered on these nodes without further forwarding, leading to the lowest forwarding cost among the four methods.

GeoComm has higher and lower forwarding cost than PROPHET in the test with the DART trace and the DNET trace, respectively. This is because in GeoComm, a node's contact probabilities with each landmark vary greatly due to people's mobility in the DART trace and remain stable in the DNET trace. Therefore, packets are frequently forwarded in the test with DART trace. On the contrary, PROPHET forwards packets greedily by only considering meeting frequency, leading to high forwarding cost in both traces. PER leads to the highest forwarding cost in the tests with both traces. This is because whenever a node moves to a new landmark, its probability of visiting a certain landmark before a deadline changes. In other words, such probabilities vary significantly with node movement. As a result, packets are forwarded for the most frequently in the network.

Total Cost: Figs. 11(d) and 12(d) plot the total costs of the six methods with the DART and the DNET traces, respectively. We see that the total costs follow $\mathrm{DTN-FLOW} < \mathrm{PGR} < \mathrm{SimBet} \approx \mathrm{PROPHET} < \mathrm{GeoComm} < \mathrm{PGR}$ and $\mathrm{DTN-FLOW} < \mathrm{PGR} < \mathrm{SimBet} \approx \mathrm{GeoComm} < \mathrm{PROPHET} < \mathrm{PGR}$ in the tests with the DART and the DNET traces, respectively. Recall that the total cost includes packet forwarding cost and maintenance cost, which is incurred by routing information forwarding. In DTN-FLOW, the maintenance cost comes from routing table updates. When a node connects to a new landmark, it forwards the routing table of its previously connected landmark to the new landmark and receives the routing table of the new landmark. In other methods, two encountering nodes exchange their calculated suitability/rank for each destination landmark and then decide whether to forward packets to the other node. Since a node's probability of meeting a landmark is lower than that of meeting another node, maintenance cost in DTN-FLOW is lower than that in other methods. Therefore, DTN-FLOW produces the lowest total cost.

Comparing Figs. 11(d) and 12(d) to Figs. 11(c) and 12(c), we notice the maintenance cost only counts a small part of the total cost. Therefore, the relationship on total cost remains the same as that on the forwarding cost. We also see that when the memory size on each node increases, the total costs of all methods increase, though the maintenance costs of each method actually remain stable. This is because the forwarding cost is much higher than the maintenance cost. The results on forwarding cost and total cost verify the high efficiency of DTN-FLOW in terms of cost with different memory sizes on each node.

3) Performance With Different Packet Rates: We also evaluated the performance of the six methods with different packet generation rates. We varied the packet rate from 100 to 1000 with 100 increase in each step.

Success Rate: Figs. 13(a) and 14(a) show the success rates of the six methods in the tests using the DART and the DNET traces, respectively. We see that the success rates follow $\mathrm{DTN-FLOW} > \mathrm{PER} > \mathrm{SimBet} \approx \mathrm{PROPHET} > \mathrm{GeoComm} > \mathrm{PGR}$. Such results match those in Figs. 11(a) and 12(a) for the same reasons. We also see that when the packet rate increases, the success rates of the six methods decrease. The forwarding opportunities in the system are determined by node memory and encountering opportunities, which are independent with the number of packets. When the number of packets increases, the number of packets that can be delivered successfully does not increase accordingly, leading to a degraded success rate. The high success rate of DTN-FLOW with different packet rates verifies the high throughput performance of DTN-FLOW.

Average Delay: Figs. 13(b) and 14(b) illustrate the average delays of the six methods in the tests using the DART and the DNET traces, respectively. We see that the average delays follow $\mathrm{DTN-FLOW} < \mathrm{SimBet} \approx \mathrm{PROPHET} < \mathrm{GeoComm} < \mathrm{PGR} < \mathrm{PER}$. This relationship remains the same as in Figs. 11(b) and 12(b) for the same reasons. Moreover, we find that when the packet rate increases, the average delays of the four methods increase. This is caused by the limited forwarding opportunities in the system. When there are more packets in the system, the average time a packet needs to wait before being forwarded increases, resulting in higher total delay. DTN-FLOW always generates the lowest average delay at all packets rates, which demonstrates the high efficiency of DTN-FLOW in terms of routing delay.

Forwarding Cost: Figs. 13(c) and 14(c) show the forwarding costs of the four methods in the tests using the DART and the DNET traces, respectively. We see that the forwarding costs $\mathrm{DTN-FLOW} < \mathrm{PGR} < \mathrm{SimBet} < \mathrm{PROPHET} < \mathrm{PER}$ with both traces and GeoComm has the second and the roughly third largest forwarding cost with the DART and the DNET trace, respectively. Again, this relationship is the same as in Figs. 11(c) and 12(c) due to the same reasons. We also see that the forwarding costs of the four methods increase when the packet rate increases. When there are more packets generated in the system, more forwarding opportunities are utilized, resulting in more packets forwarding operations. This is why the forwarding costs in Fig. 14(c) remain relatively stable when the packet rate is larger than 40.

Total Cost: Figs. 13(d) and 14(d) show the total costs of the four methods in the tests using the DART and the DNET traces, respectively. We see that their total costs again follow $\mathrm{DTN-FLOW} < \mathrm{PGR} < \mathrm{SimBet} \approx \mathrm{PROPHET} < \mathrm{GeoComm} < \mathrm{PGR}$ and $\mathrm{DTN-FLOW} < \mathrm{PGR} < \mathrm{SimBet} \approx \mathrm{GeoComm} < \mathrm{PROPHET} < \mathrm{PGR}$ in the tests with the DART and the DNET traces, respectively. This result matches that in Figs. 11(d) and 12(d) for the same reasons. We also find that the total costs of the four methods increase when the packet rates increase. This is because the maintenance costs of the four methods, which are irrelevant to the packet rate, only account for a small part of the total costs. Such results further confirm the high efficiency of DTN-FLOW in terms of cost with difference packet rates.

Combining all above results obtained with various memory sizes and packet rates, we conclude that DTN-FLOW has superior performance in achieving high throughput, low average delay, and low cost data transmission between landmarks than previous routing algorithms in DTNs.

B. Evaluation of Advanced Algorithm Extensions

In this section, we evaluate the performance of the extensions proposed in Section IV-E. We set the packet rate to 500 and the memory size on each node to 2000 kB. Other settings are the same as in the trace-driven experiments.

1) Dead-End Prevention: We evaluated the performance of our proposed dead-end prevention method. We varied H_t from 2 to 5 in the test. Table VI shows the hit rates and average delays of each test. Note the "ORG" represents the original DTN-FLOW without the proposed dead-end prevention method.

TABLE VI
EXPERIMENTAL RESULTS ON DEAD-END PREVENTION

Trace \ H_t		ORG	2	3	4	5
DART	Success Rate	0.410	0.433	0.431	0.429	0.428
	Delay ($\times 10^5 S$)	8.19	7.89	7.94	7.99	8.01
DNET	Success Rate	0.747	0.761	0.760	0.751	0.749
	Delay ($\times 10^4 S$)	8.02	7.46	7.49	7.55	7.58

TABLE VII
EXPERIMENTAL RESULTS ON LOOP DETECTION AND CORRECTION

Trace \ L_s		ORG-2	W-2	ORG-3	W-3
DART	Success Rate	0.379	0.404	0.362	0.373
	O. Delay ($\times 10^6 S$)	6.44	6.07	6.49	6.02
DNET	Success Rate	0.738	0.745	0.732	0.740
	O. Delay ($\times 10^5 S$)	5.47	5.12	5.51	5.18

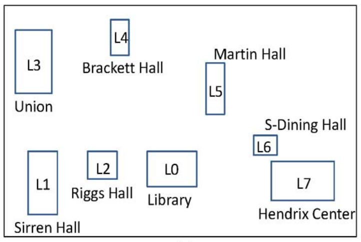
We see from the table that in the tests with both traces, when the dead-end prevention method is used, the success rate is increased and the average delay is decreased. This result confirms that our proposed method enables nodes to effectively detect dead ends and transfer their packets to other nodes through landmarks. We also find that the best performance is achieved when H_t equals 2 in the tests with both traces. This shows that $H_t = 2$ is sufficient to detect most dead ends. When H_t is larger than 2, it requires more time to identify a dead end. Then, some packets may be dropped due to TTL during the waiting, leading to more dropped packets (i.e., decreased success rate) and increased average delay.

2) Routing Loop Detection and Correction: We also evaluated the effectiveness of the loop detection and correction method proposed in Section IV-E.2. We purposely created L_s loops in this test and tested when L_s equals 2 and 3. The destination landmark of each created loop is randomly selected in the network. Table VII shows the experimental results with both traces, in which "ORG-x" and "W-x" ($x = 2, 3$) represent the DTN-FLOW without and with the proposed loop detection and prevention method when L_s equals x, respectively.

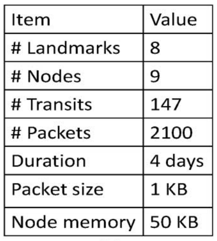
We see from the table that when two or three routing loops exist in the network, the hit rates decrease in the tests with both traces without the proposed loop detection and correction method. This is because some packets are continuously forwarded along the loop, failing to reach their destinations. We also find that the hit rates in W-2 and W-3 are only slightly lower than those when there are no routing loops as shown in Figs. 13(a) and 14(a). Such a result demonstrates that our proposed method can effectively detect and correct loops.

In order to compare the delay fairly, we measure the overall average delay, denoted O. Delay, in this test, which calculates the average delay of all packets (including the unsuccessful packets). We regard the delay of an unsuccessful packet as the experimental time, i.e., 10^7 s for the DART trace and 2×10^6 s for DNET trace. We see from the table that when the loop detection correction method is used, the overall average delay is decreased. This is because the proposed method can reduce the number of unsuccessful packets due to routing loops, thereby decreasing the overall average delay.

3) Load Balancing: We further evaluated the performance of the proposed load balancing method. For the success rate, in order to better demonstrate the effectiveness of our proposed



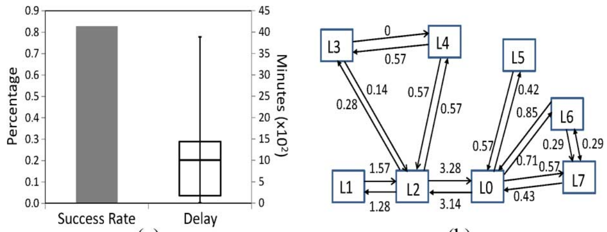
(a)



Item	Value
# Landmarks	8
# Nodes	9
# Transits	147
# Packets	2100
Duration	4 days
Packet size	1 KB
Node memory	50 KB

(b)

Fig. 15. Landmark map and configurations in the real deployment. (a) Map for landmark locations. (b) Configuration.



(a) (b)

Fig. 16. Experimental results in real deployment. (a) Success rate and delay. (b) Bandwidths of transit links.

TABLE VIII
EXPERIMENTAL RESULTS OF LOAD BALANCING ON SUCCESS RATE

Packet Rate ($\times 10^2$)		11	12	13	14	15
DART	W/O-Balance	0.315	0.307	0.295	0.283	0.260
	W-Balance	0.319	0.313	0.307	0.297	0.280
DNET	W/O-Balance	0.679	0.645	0.610	0.584	0.541
	W-Balance	0.696	0.663	0.628	0.603	0.568

TABLE IX
EXPERIMENTAL RESULTS OF LOAD BALANCING ON AVERAGE DELAY ($\times 10^4$ s)

Packet Rate ($\times 10^2$)		11	12	13	14	15
DART	W/O-Balance	89.4	89.7	89.8	90.3	90.8
	W-Balance	87.6	87.4	87.0	87.2	87.5
DNET	W/O-Balance	8.44	8.62	8.71	8.79	8.94
	W-Balance	8.36	8.43	8.66	8.71	8.81

method, we purposely enlarged the packet rate to the range of [1100, 1500] to create overloaded links in the network.

Tables VIII and IX show the experimental results on hit rates and average delays of DTN-FLOW with and without the load balancing method, denoted by "W-Balance" and "W/O-Balance," respectively. We see from the two tables that when the load balancing method is used, the success rate is increased and the average delay is decreased in the tests with both traces. This is because the backup next-hop landmark effectively offloads packets waiting for overloaded links, thereby reducing their waiting time. These results show that the proposed load balancing method can effectively offload packets on overloaded links to improve the overall routing efficiency.

C. Real Deployment

1) Settings: We deployed DTN-FLOW on our campus for real-world evaluation of its performance. We selected eight buildings as landmarks and labeled them as L_0–L_7. Their relative locations are shown in Fig. 15(a). Among the eight landmarks, L_0 is the library; L_1, L_2, L_4, and L_5 are department buildings; and L_3, L_6, and L_7 are the student center and dining halls. Each of nine students from four departments carried a Windows Mobile phone daily, each of which checks its GPS coordinator periodically to judge the landmark with which it associates.

In the test, each landmark generates 75 packets evenly in the daytime each day. We simulated a scenario in which L_0 (Library) needs to collect information from other buildings, i.e., all packets were targeted to L_0. The packet TTL was set to 3 days. We set the packet size to 1 kB and the memory on each node to 50 kB. The time unit T was set to 12 h. The deployment configuration is summarized in Fig. 15(b).

TABLE X
ROUTING TABLES IN L_2, L_4, AND L_6

Landmark ID	Destination Landmark	Next-hop
L_2	L_0, L_5, L_6, L_7	L_0
	L_1	L_1
	L_3	L_3
	L_4	L_4
L_4	L_0, L_1, L_3, L_6, L_7	L_3
	L_2, L_5	L_2
L_6	$L_0, L_1, L_2, L_3, L_4, L_5, L_6$	L_0
	L_7	L_7

2) Experimental Results: Fig. 16(a) demonstrates the success rate and the minimal, first quantile, average, third quantile, and maximal of the delays of successfully delivered packets. We see that more than 82% of packets were successfully delivered to the destination. Also, more than 75% of packets were delivered within 1400 min, and the average delay is about 1000 min. Note that the entire deployment only employed nine mobile nodes with 147 transits to forward packets. A larger deployment with more nodes would increase the success rate and reduce the delay. These experimental results demonstrate the high efficiency of the DTN-FLOW in transferring data among landmarks.

We also obtained the bandwidth of each transit link at the end of the deployment, as shown in Fig. 16(b). We omit transit links with bandwidth lower than 0.14 to show the major routing paths. The bandwidth on different transit links are within our expectation. For example, the links between L_0 and L_2 have very high bandwidth. This is because most students who attended the test are from departments located in L_2 and L_1, and they usually study in the library (L_0) and go to classes in both department buildings (L_1 and L_2). Such results justify that the DTN-FLOW can accurately measure the amount of transits among landmarks.

We further recorded the routing table on each landmark. Due to page limit, we only show those of L_2, L_4, and L_6 in Table X. We see that the routing tables match the fastest path based on transit link bandwidths shown in Fig. 16(b). For L_2, it needs to go through L_0 to reach L_0, L_5, L_6, L_7. For L_4, it relies on L_3 and L_2 to reach other landmarks. For L_6, except for L_7, it has to go through L_0 to reach other landmarks. Such results verify that the routing table update in DTN-FLOW, which relies on mobile nodes, is reliable and can reflect the suitable paths to each destination.

VI. CONCLUSION

In this paper, we propose DTN-FLOW, an efficient routing algorithm to transfer data among landmarks with high throughput in DTNs. DTN-FLOW splits the entire DTN area into subareas with different landmarks and uses node transits between landmarks to forward packets one landmark by one landmark to reach their destinations. Specifically, DTN-FLOW consists of four components: landmark selection and subarea division, node transit prediction, routing table construction, and packet routing algorithm. The first component selects landmarks from places that are frequently visited by nodes and splits the network into subareas accordingly. The second component predicts node transits among landmarks based on previous movements using the order-k Markov predictor. The third component measures the transmission capability between each pair of landmarks and uses such information to build routing tables on each landmark. In the fourth component, each landmark decides the next-hop landmark for each packet by checking its routing table and forwards the packet to the node that is most likely to transit to the next-hop landmark. Extensive analysis, experiments, and real deployment on our campus demonstrate the effectiveness of DTN-FLOW. In the future, we plan to investigate how to combine node-to-node communication to further enhance the packet routing efficiency.

References

[1] S. Jain, K. R. Fall, and R. K. Patra, "Routing in a delay tolerant network," in *Proc. SIGCOMM*, 2004, pp. 145–158.

[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. SIGCOMM*, 2003, pp. 27–34.

[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. ASPLOS-X*, 2002, pp. 96–107.

[4] M. Grossglauser and D. N. C. Tse, "Mobility increases the capacity of *ad hoc* wireless networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 477–486, Aug. 2002.

[5] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *Mobile Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[6] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–11.

[7] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. SIGCOMM*, 2007, pp. 373–384.

[8] 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. IEEE INFOCOM*, 2010, pp. 1–9.

[9] F. Li and J. Wu, "MOPS: Providing content-based service in disruption-tolerant networks," in *Proc. IEEE ICDCS*, 2009, pp. 526–533.

[10] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay tolerant networks," in *Proc. ACM MobiHoc*, 2008, pp. 241–250.

[11] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *Proc. ACM MobiHoc*, 2007, pp. 32–40.

[12] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft, "A socio-aware overlay for publish/subscribe communication in delay tolerant networks," in *Proc. ACM MSWiM*, 2007, pp. 225–234.

[13] 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. Sel. Areas Commun.*, vol. 26, no. 5, pp. 748–760, Jun. 2008.

[14] B. Chiara, M. Conti, J. Jacopini, and A. Passarella, "HiBOp: A history based routing protocol for opportunistic networks," in *Proc. WoWMoM*, 2007, pp. 1–12.

[15] J. Link, D. Schmitz, and K. Wehrle, "GeoDTN: Geographic routing in disruption tolerant networks," in *Proc. IEEE GLOBECOM*, 2011, pp. 1–5.

[16] J. Kurhinen and J. Janatuinen, "Geographical routing for delay tolerant encounter networks," in *Proc. IEEE ISCC*, 2007, pp. 463–467.

[17] I. Leontiadis and C. Mascolo, "GeOpps: Geographical opportunistic routing for vehicular networks," in *Proc. IEEE WoWMoM*, 2007, pp. 1–6.

[18] J. Leguay, T. Friedman, and V. Conan, "DTN routing in a mobility pattern space," in *Proc. ACM SIGCOMM WDTN*, 2005, pp. 276–283.

[19] J. Fan, J. Chen, Y. Du, W. Gao, J. Wu, and Y. Sun, "Geocommunity-based broadcasting for data dissemination in mobile social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 734–743, Apr. 2013.

[20] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: An efficient routing in disruption-tolerant networks," in *Proc. ACM MobiHoc*, 2009, pp. 95–104.

[21] C. Hedrick, "RFC1058—Routing information protocol," 1988.

[22] K. C. Lee, M. Le, J. Härri, and M. Gerla, "LOUVRE: Landmark overlays for urban vehicular routing environments," in *Proc. IEEE VTC-Fall*, 2008, pp. 1–5.

[23] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proc. MobiCom*, 2004, pp. 187–201.

[24] A. Balasubramanian, B. Levine, and A. Venkataramani, "Enhancing interactive web applications in hybrid networks," in *Proc. ACM MobiCom*, 2008, pp. 70–80.

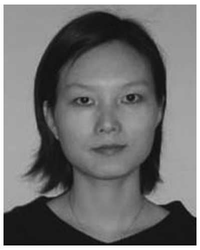
[25] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive Wi-Fi mobility data," in *Proc. IEEE INFOCOM*, 2004, pp. 1414–1424.

[26] M. Lin, W.-J. Hsu, and Z. Q. Lee, "Predictability of individuals mobility with high-resolution positioning data," in *Proc. UbiComp*, 2012, pp. 381–390.

[27] K. Chen and H. Shen, "DTN-FLOW: Inter-landmark data flow for high-throughput routing in DTNs," in *Proc. IEEE IPDPS*, 2013, pp. 726–737.

Kang Chen (S'13) received the B.S. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2005, and the M.S. degree in communication and information systems from the Graduate University of Chinese Academy of Sciences, Beijing, China, in 2008, and is currently pursuing the Ph.D. degree in computer engineering at Clemson University, Clemson, SC, USA.

His research interests include distributed mobile systems and computer networks.



Haiying Shen (M'07–SM'13) received the B.S. degree in computer science and engineering from Tongji University, Shanghai, China, in 2000, and the M.S. and Ph.D. degrees in computer engineering from Wayne State University, Detroit, MI, USA, in 2004 and 2006, respectively.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA. Her research interests include distributed computer systems and networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing.

Dr. Shen is a Microsoft Faculty Fellow of 2010 and a member of the Association for Computing Machinery (ACM).