TSearch: Target-Oriented Low-Delay Node Searching in DTNs with Social Network Properties

Li Yan, *Student Member, IEEE,* Haiying Shen, *Senior Member, IEEE,* and Kang Chen, *Student Member, IEEE*

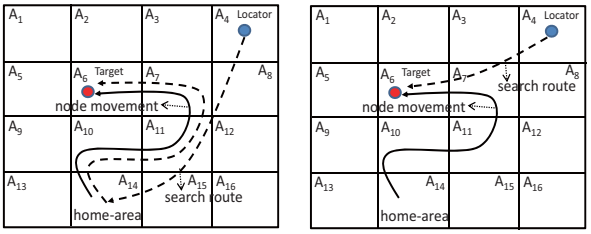
***Abstract*—Node searching in delay tolerant networks (DTNs) is of great importance for different applications, in which a locator node finds a target node in person. In the previous distributed node searching method, a locator traces the target along its movement path from its most frequently visited location. For this purpose, nodes leave traces during their movements and also store their long-term movement patterns in their frequently visited locations (i.e., preferred locations). However, such tracing leads to a long delay and high overhead on the locator by long-distance moving. Our trace data study confirms these problems and provides foundation of our design of a new node searching method, called target-oriented method (TSearch). By leveraging social network properties, TSearch aims to enable a locator to directly move towards the target. Nodes create encounter records (ERs) indicating the locations and times of their encounters and make the ERs easily accessible by locators through message exchanges or a hierarchical structure. In node searching, a locator follows the target's latest ER, the latest ERs of its friends (i.e., frequently meeting nodes), its preferred locations and the target's possible locations deduced from additional information for node searching. Extensive trace-driven and real-world experiments show that TSearch achieves significantly higher success rate and lower delay in node searching compared with previous methods.**

***Index Terms*—Delay-tolerant networks, node searching, social network properties.**

I. Introduction

In recent few years, Delay Tolerant Networks (DTNs) attract significant attention from researchers. In such sparsely distributed networks, node searching, in which a *locator* node finds a *target* node in person, is of great value in node management and many applications. For example, in a DTN formed by mobile device holders in a hospital, a campus, a disaster area or a national park, a user needs to find another user. In a DTN in battlefield, a node needs to find a node carrying a malfunctioning device for repair. In a DTN formed by vehicles [1], a vehicle may need to find another vehicle to communicate. The DTN condition without infrastructure or continuous network connectivity poses a challenge for designing an efficient distributed node searching algorithm.

Some previous object tracking systems [2]–[6] in wireless networks provide high localization accuracy or search efficiency based on the geographical information provided by central base stations or other infrastructures. However, the extra infrastructure requirement is costly and impractical for DTNs (e.g., in battlefields). DTN routing algorithms [7]–[14] can be indirectly used for node searching. In routing, a node forwards the message to the node with a higher probability of meeting the destination. Then, to find a target, a locator can move with the selected message carriers by regarding the target as the message destination. However, since the locator must follow multiple nodes in routing and each node has its own movement path rather than moving directly towards the target, such a node searching method generates a high delay and overhead on the locator by long-distance moving.



(a) Node searching in DSearching. (b) Node searching in TSearch.

Fig. 1: Node searching in DSearching and TSearch.

Recently, a distributed node searching algorithm (DSearching) has been proposed [15]. It divides the entire DTN area to sub-areas. During a node's movement, it tells several nodes in its current sub-area its next sub-area (called transient visiting record (VR)) before moving out. Each node also deduces its long-term mobility pattern (MP), which indicates the sub-areas it has high probabilities to move to from each of its frequently visited sub-area (i.e., preferred location). It distributes its MP from sub-area A_i to long-staying nodes in A_i. A node's home-area is the sub-area it has the highest staying probability, and this information is stored in all sub-areas. As shown in Figure 1(a), a locator starts from the target's home-area and follows the VRs. When these records are absent in searching, the locator moves to the next sub-area with the highest probability based on the MP. If the MP is not available, the locator searches nearby sub-areas for VR and MP.

However, both moving to the target's home-area and tracing along the target's movement path may take a long time. First, as Figures 1(a) and 1(b) show, this tracing process ($A_4 \rightarrow A_{14} \rightarrow A_{10} \rightarrow A_{11} \rightarrow A_7 \rightarrow A_6$) generates high delay. A locator can take a shortcut to directly move towards the target ($A_4 \rightarrow A_7 \rightarrow A_6$). Second, when a locator in a sub-area (say A_{11} in Figure 1(a)) loses trace (i.e., VR), it moves to the predicted next sub-area from A_{11} (i.e., A_7), which generates extra search path length. Instead, directly moving to the sub-area that the target frequently visits (i.e., $A_{11} \rightarrow A_6$) generates shorter searching delay and overhead. Also, in this step, the next sub-area with the highest probability may not be the one that the target actually moves to, which leads to high searching delay and even searching failure. Further, storing the target's MP in a limited number of sub-areas may prevent it from locators, which may also increase searching delay.

In this paper, we have conducted trace data [16], [17] study, which confirms the above problems of DSearching and

also lays a foundation of our proposed target-oriented method (called TSearch). As DSearching, TSearch is also designed for DTNs with social network properties such as mobility range stability, certain mobility patterns and certain frequently meeting nodes (i.e., friends), and skewed visiting places (i.e., preferred locations) shown in previous works [8], [18], [19]. By leveraging these social network properties, TSearch aims to enable a locator to directly move towards the target.

In TSearch, the information for node searching consists of encounter records (ERs), friends, and preferred locations. Nodes record and disseminate ERs that indicate the locations and times of their encounters. A locator (N_i) always directly moves to the sub-area in the latest ER of the target (N_j) known by itself (Figure 2). In the absence of N_j's newer ER, N_i relies on the ERs of N_j's friends. In the absence of the friends' ERs, N_i directly moves to the nearest preferred location of N_j, and also requests the nodes sharing the common preferred locations with N_j to search N_j simultaneously. If the locator does not have the above information of its target, it relies on additional information (i.e., relation graph, which represents nodes' social relationship based on their mutual friends) to deduce a proper search area. To make the information for node searching globally accessible, TSearch adopts the hierarchical structure from [20], [21], in which each sub-area has a long-staying node (called anchor) to collect the information from nodes, and nodes that frequently transit between two sub-areas (called ambassadors) are responsible for the information updates between anchors. TSearch provides an option for nodes to piggyback ERs on the information exchanged between neighbors to expedite the information dissemination. Our contributions are threefold:

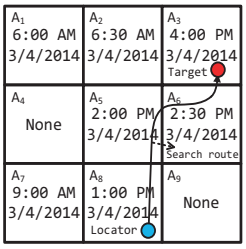


Fig. 2: ER-based node search.

(1) Extensive study on two real traces [16], [17] confirms the drawbacks of DSearching and lays the foundation of TSearch.
(2) We propose TSearch, which is the first work (to our best knowledge) that aims to enable locators directly move towards the targets with easily accessible information to reduce node searching delay by utilizing social network properties.
(3) We have conducted trace-driven and field experiments, which verify the superiority of TSearch over previous methods.

The paper is organized as follows. Section II presents an overview of related work. Section III presents our trace analysis results. Section IV presents the detailed design of TSearch. Section V presents the experimental results of TSearch. Section VI concludes this paper with remarks on our future work.

II. Related Work

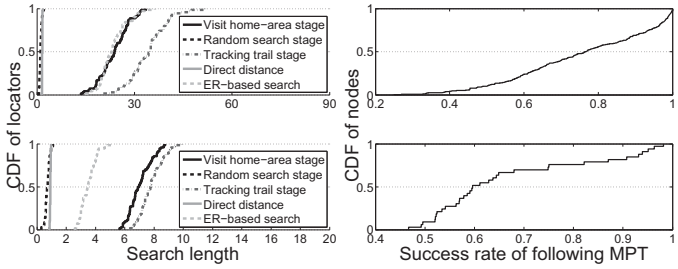
Object searching in mobile networks has attracted much attention. Juang *et al.* [2] proposed a method that sends the positions of animals to the central station through hop-by-hop broadcasting. By utilizing the flock behavior of sheep, Thorstensen *et al.* [3] proposed a system monitoring and report the positions of other sheep to the server through wireless communication. Cenwits [4] and SenSearch [5] provide object searching services in wilderness areas. They utilize opportunistic encounters among nodes to forward information to infrastructures. However, these methods need extra infrastructures, which is not practical for DTNs. Symington *et al.* [6] models the encounters of sensors as a connected graph and sensors' estimated trajectories. Then, the tracking process is to find a forwarding path to the target that meets its estimated trajectory. However, since the connected graph is generated centrally, it is not suitable for decentralized DTNs. DSearching [15] was proposed specifically for node searching in DTNs. As indicated previously, it provides insufficiently efficient node search by aiming to enable a locator to trace the target along its movement path from its home-area. Routing algorithms [7]–[14] can be indirectly applied for node searching, but the hop-by-hop routing is not efficient for node searching in DTNs. In contrast, TSearch does not need infrastructure or central server, and is the first work that enables locators to directly move towards targets to achieve low search delay and overhead.

III. Rationale of TSearch Design

In this section, we present the rationale of TSearch through trace analysis. We used the DART trace [16] (DART) and the DieselNet AP trace (DNET) [17]. DART is a 119-day record of wireless devices carried by students on Dartmouth College campus. DNET is a 20-day record of WiFi nodes attached to the buses of UMass college town. We filtered out nodes with few occurrences and merged access points (APs) within short ranges to one sub-area. Finally, DART has 320 nodes and 159 sub-areas, DNET has 34 buses and 18 sub-areas. We set the initial period to 30 days for DART and 2.5 days for DNET, during which nodes collect information. We randomly selected 70 locators and each locator randomly chose a target to search periodically for 90 times and the average experimental result of each locator is reported. The periodical time was set to 1 day in DART and 4 hours in DNET. Considering students move less frequently than buses, the search TTL (Time-To-Live) was set to 24 hours in DART and 4 hours in DNET. Node searches using more than TTL are considered as unsuccessful. In each of the following figures, the top figure is for DART and the bottom figure is for DNET. The key findings are:

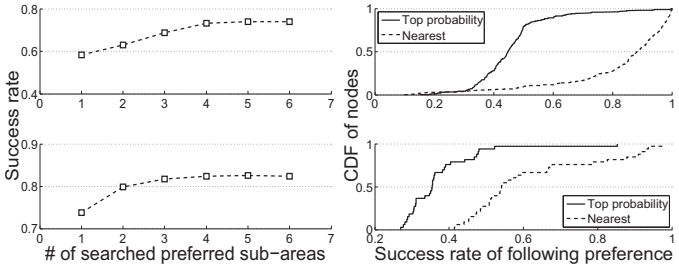
(1) DSearch is not efficient in node searching and ER is effective in enhancing node searching speed.
(2) Node searching needs to consider multiple preferred locations of the target node and searching the preferred sub-area nearest to the latest location of the target node is more accurate than searching in the target node's most frequently visited sub-area.
(3) The target's frequently met nodes are useful for searching.
(4) ER limits the range of node searching.
(5) Anchor is effective in maintaining mobility information in its sub-area, and ambassador is effective in maintaining information consistency among anchors.
(6) Long staying nodes (anchors) and frequently transiting nodes (ambassadors) exist most of the time.
(7) The preferred locations of the target node's friends are also useful in node searching.

1) Leveraging Encounter Records (ERs): We define *search length* as the number of sub-areas the locator transited in searching. DSearching has three stages: i) a locator moves to the target's home-area, ii) tracks along its moving trail,



(a) The number of searched sub-areas. (b) Choosing the top next sub-area.

Fig. 3: Drawbacks of DSearching.



(a) Searching top preferred locations. (b) Preferred locations to search.

Fig. 4: Node searching based on preferred locations.

iii) and may randomly search in neighbor areas. As shown in Figure 1, such searching may generate a long search length. To confirm this drawback, we measured the cumulative distribution function (CDF) of the search lengths of these three searching stages as shown in Figure 3(a). It also includes the locator-target initial direct distance. We see that the direct distance is very short (within 3 sub-areas in DART and 2 sub-areas in DNET). However, 50% of locators need to travel more than 24 and 6 sub-areas to reach the target's home-area, and also travel more than 35 and 8 sub-areas in the tracking stage in DART and DNET, respectively. These results demonstrate that locators must travel many sub-areas in the first two stages in DSearch. To avoid this, we propose the concept of encounter record (ER), which records the location and time of a node. The ERs of nodes are disseminated among nodes for locators to access, so they can move directly to the most recent locations of the targets. We measured the search length of this method as shown in Figure 3(a) when we temporarily let nodes piggyback ERs on the messages exchanged between neighbors for dissemination. The result shows that ERs are effective in accelerating node searching.

2) Leveraging Preferred Locations: A node's *preferred locations* are defined as the sub-areas the node frequently visits. In DSearching, by referring the target's Mobility Pattern Table (MPT), the locator always moves to the target's preferred sub-area with the highest probability. To verify the effectiveness of MPT, for each node, we used it to search the node's next sub-area from its previous sub-area in its entire movement path, and calculated the success rate. Figure 3(b) shows the CDF of the success rate. We see that 20% of the nodes have success rate less than 60% and 50% of the nodes have success rate less than 75% in DART, while 25% of the nodes have success rate less than 55% and 75% of the nodes have success rate less than 70% in DNET. Therefore, a locator should not ignore the other preferred locations that a target has a high probability (though not the highest probability) to move to. When a target leaves a sub-area, it may be moving to a nearby preferred location. Then, searching the preferred location nearest to the target node's most recent location may lead to a higher success rate. To verify these, we draw Figures 4(a) and 4(b). We ranked each node's visited sub-areas based on the visiting frequency and consider the top sub-areas that constitute 60% of visiting frequency as its preferred locations.

Figure 4(a) shows the average success rate of searching different numbers of preferred locations. We see that searching the top preferred location only leads to 59% and 74% success rates in DART and DNET, respectively. Searching top 4 (in DART) and 3 (in DNET) preferred locations can achieve 73% and 82% success rate, respectively, and then searching additional 1 or 2 preferred locations only generates limited improvement. Figure 4(b) shows the success rates of searching the top and nearest preferred location, respectively. We can see that selecting the nearest preferred location is more accurate than selecting the top preferred location.

3) Leveraging Frequently Met Nodes (i.e., Friends): We define nodes N_i and N_j are friends if their encounter frequency is higher than a threshold. Since each node has certain friends [8], [18], [19], a locator is likely to meet the target through finding its friend. To verify this, we draw Figure 6 that shows the CDF of success rate of following the ERs of the target and the target's friends, respectively. To offer a baseline, we also draw the success rate of finding the target node through following the ERs of randomly selected nodes. We regard a node's friends as the nodes taking up the top 60% of all contacts with the node. Each node has 12 friends in average in DART and DNET. We see that following the targets' ERs, about 60% of the locators have success rate higher than 92% in DART and 91% in DNET. Following the ERs of the target's friends, about 60% of the locators have success rate higher than 70% in DART and 80% in DNET. This shows the ER of target's friend is useful in node searching.

4) Search Range Constraint: Based on node velocity V and the time and location in the latest ER of a node, the range that the node possibly stays (called coverage area) can be determined. It is a circle with VT_e as the radius and the ER location as the center, where T_e is the elapsed time since the time in the ER. For example, in Figure 5, based on the T_e indicated in the ER corresponding to sub-area A_5 and the velocity V, the possible positions of the target are included in the circle determined by the radius VT_e. For each node, at each of its locations, we checked whether it is within its coverage area based on its previous location. Figure 7 shows the CDF of the *coverage ratio* defined as the ratio of the number of locations in the coverage areas. We see that when the TTL of ER is set to 4 hours in DART and 2 hours in DNET, 80% of nodes have coverage ratio higher than 70% in DART and DNET. The result shows that ERs with a proper TTL can be used to limit the searching areas of the locators.

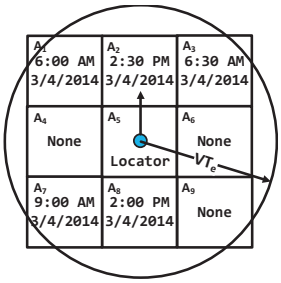


Fig. 5: Constrained search area.

5) Information Dissemination: Nodes may move locally in only a few sub-areas [8], [18], [19]. To make ERs globally accessible, we adopt a hierarchical structure in previous

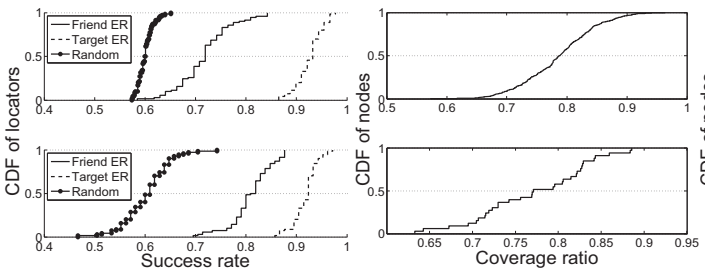
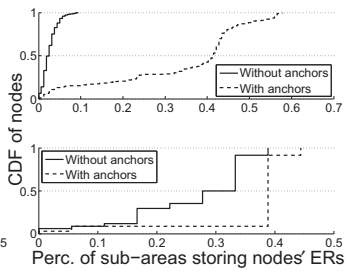
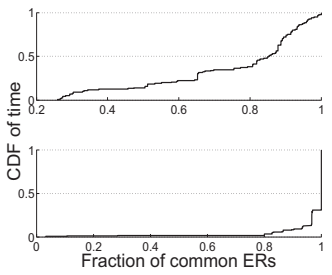


Fig. 6: Following the ERs of the target or its friends.

Fig. 7: Constraining searching range.



(a) Effectiveness of anchors.



(b) Effectiveness of ambassadors.

Fig. 8: Role-based information dissemination.

works [20], [21], which verified the existence of long-staying nodes (i.e., anchors) in each sub-area and nodes that frequently transit between two sub-areas (i.e., ambassadors). In our method, nodes report information to anchors and ambassadors are responsible for the record updates between anchors.

In order to see the effectiveness of anchor, we measured the percent of sub-areas that have the ERs of certain nodes at the time point of 120 hours and 20 hours in DART and DNET, respectively. From Figure 8(a), we see that 50% of nodes have their ERs disseminated to less than 2% and less than 40% of all the sub-areas without and with anchors in DART, and to less than 27% and less than 39% of all the sub-areas without and with anchors in DNET. This confirms our expectation.

In order to see the effectiveness of ambassadors, in each hour during the 120 hours and 20 hours in DART and DNET, respectively, we measured the ratio of common ERs among all anchors, as shown in Figure 8(b). We see that about 80% of the time, the ratio of common ERs among anchors is higher than 40% in DART, and higher than 90% in DNET. It confirms that the ambassadors can help maintain a high degree of consistency among anchors.

6) Stability of the Number of Nodes with Different Roles: We draw Figure 9 to show the number of nodes with each role throughout the time of the two traces. We sampled the number of nodes with each role every 3 days and 12 hours for DART and DNET, respectively. We see that after an initial time period, the number of nodes with each role stabilizes though with some fluctuation. The number of anchors is around 80 and 9 with fluctuating range of [10, 15] and [2, 3] in DART and DNET, respectively. The number of ambassadors is around 200 and 15 with fluctuating range of [10, 60] and [3, 5] in DART and DNET, respectively. The number of ambassadors fluctuates more than that of anchors. This is because some previously qualified ambassadors may change mobility patterns. Compared with DART, the number of nodes with different roles in DNET fluctuates more. This is because buses (DNET) move more scheduled than pedestrians (DART). The results verify the constant existence of nodes suitable for anchors and ambassadors. Although the numbers fluctuate, the range is only within 35%.

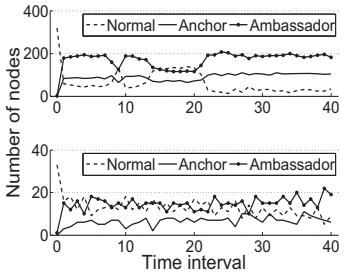


Fig. 9: Stability of number of nodes with different roles.

7) Leveraging Friends' Preferred Locations: In a DTN with social properties, nodes usually meet their friends. On the other side, nodes also have preferred sub-areas (Section III-2). Therefore, we may find the preferred locations of nodes' friends are also useful for indicating the nodes' location. To confirm this, we firstly identify each node's friends with the method introduced in Section III-3. We call a node's preferred location with the highest visiting frequency its *most preferred location*. We measure the ratio of the total time a node stays in the most preferred locations of its friends over the node's total activity time. It is calculated by $\sum_j T_{p_j}/T_t$, where T_{p_j} is the time the node N_i stays in the most preferred location of its friend N_j, and T_t is N_i's total activity time. Figure 10 shows the CDF of this ratio on all nodes. We see that about 75% of nodes spend more than 60% of their time, and about 80% of nodes spend more than 40% of their time staying in their friends' most preferred locations in DART and DNET, respectively. The results confirm that most nodes spend much of their time in the most preferred locations of their friends.

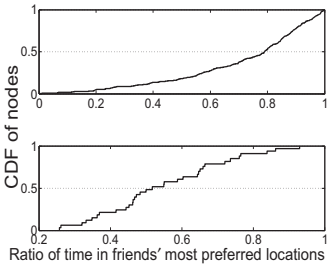


Fig. 10: Ratio of time in friends' most preferred locations.

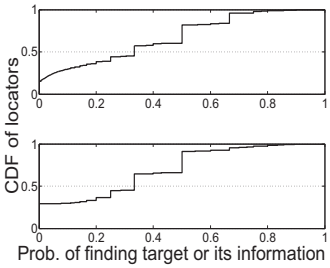


Fig. 11: Following the most preferred sub-areas of target's friends.

Thus, we conjecture that searching in the most preferred locations of a target's friends can help the locator find the target or its information for node searching. To confirm our conjecture, we draw Figure 11 that shows the CDF of the probability of finding the target or its information through only following the most preferred locations of its friends in node searching. We see that about 40% of locators achieve the probability of more than 50%, and about 30% of locators achieve the probability of more than 50% in finding the target or its information. The result confirms that using the most preferred locations of the target's friends as a complementary approach in node searching is effective.

IV. Design of TSearch

As in DSearching, we partition the network area into several sub-areas (denoted by A_i) to represent node positions (Figure 1). Note that more partitioned sub-areas leads to more accurate node positions but also higher maintenance overhead. To achieve a balance for this tradeoff in deciding sub-areas, we consider the fact that nodes usually have "gathering"

preference in popular places that are frequently visited by all nodes (e.g., libraries, dorms, departments) [9], [22]–[24]. Specifically, the DTN area is partitioned as below:

- Each sub-area contains one popular place.
- The area between two popular places is evenly split to the two sub-areas containing the two places.
- There is no overlap among sub-areas.

The sub-area partition is completed off-line. Each node is configured with the area map when it joins the DTN. Each node in the network has a positioning device (e.g. GPS) that can be used to learn its current sub-area. TSearch is designed for DTNs with social properties [8], [18], [19] and it leverages these properties for efficient node searching.

- Mobility range stability means that the movement area of each user is significantly smaller than the whole area, and the change of its mobility range is small over time [18], [20]. Therefore, ERs can be used to search targets. Even though a target is no longer in an ER's location, it is very likely to stay nearby (Figure 3(a)).
- Following the ERs of a target's friends (i.e., frequently meeting nodes) can be an auxiliary method (Figure 6).
- As each node has preferred locations, moving towards a target's preferred location has a high probability of finding it on the way or at the destination (Figure 4).
- Mobility pattern feature indicates that some nodes are relatively stable while some nodes transit frequently between sub-areas. As previous works [20], [21], we assign different roles (i.e., anchor, ambassador) to nodes with certain mobility features (Figure 8).
- Since most nodes spend much time in the most preferred locations of their friends (Figure 10), searching in the most preferred locations of a target's friends is effective in finding the target or its information (Figure 11).

Accordingly, TSearch has three types of location information of nodes: ERs, friends' ERs and preferred locations, along with additional information for search area determination: relation graph. Due to privacy concerns, some nodes may be unwilling to share their mobility information. We can use previous schemes [25]–[27] to motivate nodes to share their mobility information. In following sections, we first introduce the location information and the additional information in Section IV-A and Section IV-B, respectively. We then present the node searching algorithm in Section IV-C, and finally explain the information dissemination in Section IV-D.

A. Information for Node Searching

A node generates an ER for each of its neighbors (i.e., the nodes in its transmission range) upon encountering. Node N_i generates ER for neighbor N_j in the form of $< N_i, N_j, L_{ij}, T_{ij} >$, where L_{ij} and T_{ij} denote the current sub-area and current time. L_{ij} is attached with a GPS position within the current sub-area. If N_i already has N_j's ER, it only needs to update the L_{ij} and T_{ij} in the existing ER. Finally, each node maintains its ER table based on its encounters with other nodes as shown in Table I. To constrain the storage overhead for ERs and ensure their validity in guiding node searching, TSearch sets a TTL for ERs. Each node deletes ERs after TTL upon their creation. Due to the mobility range stability, the number of nodes that node N_i encounters is limited [8], [18], [19], which means that N_i's ERs are created in a limited number of nodes. In Section IV-D, we will introduce methods to enable a locator of N_i to access its ERs.

TABLE I: An encounter record (ER) table.

ER creator	Node ID	Sub-area	Time
N_2	N_3	A_1	1:00pm, 3/4/2014
N_1	N_7	A_2	2:00pm, 3/4/2014
...	...	...	...

After a node joins in the system, it calculates its friends and preferred locations from movement records as shown in Table II. Because each node has mobility patterns, such information do not update frequently. Each node ranks its encountered peers by their encounter frequency in descending order. TSearch only considers the mobility information of frequently encountered nodes, so the encounter frequency threshold for determining friendship was set to 60%. Table II shows node N_1 has the probability of 0.3 to meet N_3, probability of 0.2 to meet N_4 and probability of 0.1 to meet N_7. Meanwhile, it also knows that N_1 has the probability of 0.25 to appear in A_3, probability of 0.15 to appear in A_4 and probability of 0.1 to appear in A_5.

TABLE II: Friends and preferred locations of N_1.

Node	Friends	Meeting prob.	Preferred locations	Visiting prob.
N_1	N_3	0.3	A_3	0.25
	N_4	0.2	A_4	0.15
	N_7	0.1	A_5	0.1

The TTL should be properly set so that the ERs can reflect the most recent position of corresponding nodes. The TTL is determined based on many factors (average encounter frequency, average encounter duration, number of nodes in the network). In this paper, we determine the TTLs heuristically based on node encounters in DART and DNET. We leave the method to accurately determine TTL value as our future work.

B. Additional Information for Node Searching

In this section, we handle the problem of how a locator uses existing information in an anchor to deduce the next search area when the direct information of the target (i.e., ER, friends and preferred sub-areas) is not available. Section III shows that searching in the most preferred locations of a target's friends is effective in finding the target or its information for node searching. Therefore, we additionally consider this auxiliary information in node searching under two cases. First, the locator knows the most preferred locations of the target's friends. Second, the locator does not know the most preferred locations of the target's friends.

1) Possible Locations of Nodes: After anchors and ambassadors collect and disseminate nodes' information, anchors will have a general view of the friends and preferred locations of many nodes. For each friend of N_i, the anchor may know its most preferred location. Among the most preferred locations of N_i's friends, the locator needs to choose one location to search N_i, which should be the location that N_i has the highest probability to visit among these most preferred locations. We use *weight* to represent the probability that a target visits its friend's most preferred location. It can be calculated by the product of their meeting probability (P_m) and the visiting probability of the friend on this location (P_v). For each node,

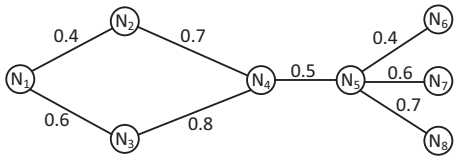


Fig. 12: Relation graph of nodes.

an anchor builds a table recording its friends, their most preferred locations, their visiting probabilities to their most preferred locations and the corresponding weights. Finally, the locator chooses the most preferred location with the highest weight to search the target.

TABLE III: Possible locations of N_1.

Node	N_1			
Friends	Meeting prob.	Most preferred locations	Visiting prob.	Weight
N_3	0.2	A_1	0.5	0.1
N_4	0.1	A_9	0.4	0.04
N_7	0.3	A_3	0.1	0.03

Table III shows an example for such a table for N_1. From the table, the locator knows that N_3's most preferred location is A_1 with corresponding visiting probability of 0.5 and weight of 0.1, N_4's most preferred location is A_9 with corresponding visiting probability of 0.4 and weight of 0.04, and N_7's most preferred location is A_3 with corresponding visiting probability of 0.1 and weight of 0.03. Then, the locator chooses A_1 to search N_1, which has the highest weight (i.e., the highest probability that N_1 stays among these most preferred locations). If the locator cannot find N_1 in A_1, it is very likely to find N_1's information for node searching (i.e., ER, friends and preferred sub-areas) from the anchor of A_1 since N_1 has a high probability of staying in A_1. Using this information, the locator can find N_1.

2) Relation Graph Based Search Area Determination: Searching the most preferred location of the target's friend is based on the assumption that the locator can learn the most preferred locations of the target's friends from the anchor or information exchange. However, the locator may not learn the information of the target's friends. In this case, as long as the locator can approach these possible locations, its probability of finding the target increases. Since nodes report the information to anchors, the locator may be able to find the information of the target's friend or even the information of the target by searching the most preferred location of the friend of the target's friend. Generally, for the nodes known by an anchor, considering their closenesses (i.e., meeting probability) to the target helps the locator determine which node's information is the most useful in helping the locator approach the target.

For example, N_i and N_j have some mutual friends. Suppose a locator wants to find N_j, but it cannot proceed the node searching because it does not know the ERs of N_j or its friends, or the preferred locations of N_j or its friends. However, by searching the most preferred location of N_i for the information of the mutual friends, the locator can have a high probability of finding N_j. Specifically, based on Figure 10 and Figure 11, in N_i's most preferred location, the locator is likely to find the mutual friends or their information (e.g., ER, friends, preferred locations). Since N_j also frequently meets with the mutual friends in their most preferred sub-areas, the locator is likely to find N_j or its information on the way approaching the preferred locations of the mutual friends.

To realize this method, each anchor builds a relation graph (Figure 12) that connects all nodes known by itself. In the relation graph, each vertex represents a node, each link is associated with a weight, which is the meeting probability of the two connected nodes. Each path connecting two nodes consists of several links. The weight of a path is calculated as the product of the weights of its composing links. The relation closeness between a pair of nodes equals to the maximum weight among the weights of the paths connecting them.

From the relation graph, the locator firstly determines and ranks each node's relation closeness to the target. Secondly, the locator learns the information of the nodes whose closeness to the target is higher than a threshold. Although each anchor's relation graph tries to include connections between as many nodes as possible, usually only the information of the target node's closely related nodes is useful for node searching. Therefore, we set the closeness threshold to 0.2. Then, the locator can deduce a collection of nodes that are close to the target. Next, for nodes in the collection, the locator multiplies the relation closeness of each node by the node's visiting probability on its most preferred location to be the node's weight. Finally, the locator chooses the most preferred location of the node with the highest weight to be the next destination. Since the relation graph is established from the long-term visiting patterns of nodes, it is less likely to be affected by the nodes' transient mobility information. On the other hand, if some nodes leave the system, the information related to the nodes should be abandoned. Therefore, we set the TTL of maintaining relation graphs to 24 hours in DART and 4 hours in DNET. That is, the anchor will recalculate the encounter frequency, visiting frequency and reconstruct the relation graph during every TTL.

For Figure 12, Table IV shows the most preferred locations of nodes known by the anchor. Suppose the locator presently stays in A_5, which is the most preferred location of N_1, and its target node is N_6. However, the anchor only knows the most preferred locations of the nodes shown in Table IV. According to Figure 12, the ranking of the nodes' closeness to the target is: $N_4(0.2) > N_3(0.16) > N_2(0.14) > N_1(0.096)$. Suppose the closeness threshold is 0.1. Thus the most preferred locations of N_4, N_3 and N_2, namely A_9, A_1 and A_2, are candidate destinations. According to Table IV, the weights of these locations are $A_9(0.12) > A_1(0.112) > A_2(0.07)$. Finally, the locator chooses A_9 as the next search destination.

TABLE IV: Most preferred locations of nodes.

Node	N_1	N_2	N_3	N_4
Preferred location	A_5	A_2	A_1	A_9
Visiting prob.	0.3	0.5	0.7	0.6

C. Target-Oriented Node Searching

The information is collected and disseminated by anchors and ambassadors, respectively. A DTN can also choose to piggyback ERs on packets exchanged between neighbors to expedite the information dissemination if it can afford this additional transmission overhead. We will introduce the details for the information dissemination in Section IV-D. In TSearch, the priorities of information is ordered by ERs>friends' ERs>preferred locations>additional information. Specifically, when a locator searches a target, if it has the ER of the target,

it moves towards the location in the ER. During the movement, if the locator receives a newer ER (with a more recent time), it moves towards the new location. In the absence of an ER, the locator finds the ER of the target's most frequently encountered friend, and then moves towards the location in this ER. If the ERs of the target's friends are not available, the locator moves towards the preferred location nearest to the target's most recent location indicated by ER. In order not to miss other frequently visited places of the target, we propose an agent-based simultaneous searching scheme, in which the locator requests a certain number of nodes sharing these preferred locations to search the target simultaneously. When above target's direct information is unavailable, it uses the additional information maintained by anchors to assist node searching. In the absence of all the information, the locator searches in the coverage area of the target. In the following, we present the details of each step.

1) Node Searching Based on ERs: If a locator has or can access the ER of its target, it directly moves to the location in the ER, say A_i. The ER may not provide the current location of the target and the target may move to a sub-area near A_i. Therefore, during searching, if the locator receives a newer ER which represents a more recent appearance place of the target, it moves to this new place.

2) Node Searching Based on Friends' ERs and Preferred Locations: It is possible that in the disconnected DTN with sparsely distributed nodes, the most recent ERs of the target are not transmitted to the locator or its local anchor in time. In the absence of the target's ER initially or when the locator arrives at the moving destination but cannot find the target, the locator queries the target's friends and their ERs, and the preferred locations of the target from its local anchor. The locator finds the ER of the friend that has the highest meeting probability with the target and moves towards the location in this ER. As the friend has a high probability of meeting the target, the locator has a high probability of finding the target.

In the case that no newer ER of the target or no ERs of the target's friends can be found, the locator can use the target's visiting preference for node searching. Based on our observations in Section III, the locator itself moves to the nearest preferred location of the target, and relies on M number of nodes (as agents) to search the target in top M preferred locations of the target. M is an empirical parameter determined by the node mobility, the network size and etc. For example, as shown in Figure 4(a), $M = 4$ for DART and $M = 3$ for DNET. The selected agents must have high probabilities of meeting the target and of moving to the M preferred locations. These agents then should be the nodes that have these common preferred locations with the target. The locator queries the local anchor for such nodes in the current sub-area. For each of the M top preferred location A_k, the locator queries the nodes that have A_k as their preferred locations about the time they will move to A_k, and then chooses the node with the earliest time to search the target. If an agent finds the target, it uses a routing algorithm [7]–[13] to send a notification message with the latest ER to the locator. Then, the locator moves to the new destination in the ER.

Within each sub-area, nodes may not be in the transmission ranges of each other. That is, even if an agent arrives at the target's sub-area, it may not find the target quickly. In order to increase the success rate, multiple agents can be sent to each selected preferred location. The overhead of failing to find the target in a sub-area equals $(1 - P)n_a$, where P is the target's probability of visiting the sub-area and n_a is the number of agents for the sub-area. In order to quickly find the target in a sub-area while constraining the searching overhead, for each selected sub-area, the number of agents should be set to a value proportional to the target's visiting probability in the sub-area. That is, if the target has a higher probability of visiting a sub-area, more agents moving to that sub-area are designated and vice versa.

3) Node Searching Based on Additional Information: It is possible that only partial information of the target is available in some sub-areas distant to the target. For example, when the locator only knows the friends of the target, but can find neither the ERs of the friends nor the preferred locations of the target, the node searching based on friends' ERs or target's preferred locations cannot be conducted. In this case, the most preferred locations of the nodes close to the target in the relation graph are used as the next searching locations.

Specifically, when a locator enters a sub-area, it firstly accesses the relation graph from the anchor of the sub-area. For all the nodes known by the anchor, the locator ranks the nodes according to their closeness to the target. The locator then identifies a collection of nodes with closeness higher than the closeness threshold. For each node in this collection, the locator calculates the weight on the node's most preferred location and chooses the location with the maximum weight as the next searching location. Then, the locator itself moves to this location to search this target. For other candidate locations, the locator requests ambassadors that are moving to the candidate locations to assist the search.

For example, as shown in Figure 12, suppose the closeness threshold is 0.1, the nodes whose closeness to target N_6 is larger than this threshold are N_4(0.2), N_3(0.16) and N_2(0.14). According to Table IV, the weights of the nodes' most preferred locations are $A_9(0.12)$, $A_1(0.112)$ and $A_2(0.07)$, respectively. Thus the locator moves to A_9 (N_4's most preferred location). The other two assisting ambassadors move to A_1 (N_3's most preferred location) and A_2 (N_2's most preferred location), respectively. If an ambassador finds the target or its direct information, it uses a routing algorithm [7]–[13] to send the message to the locator. Then, the locator can switch to other searching methods or determine a better search sub-area.

4) Node Searching In Coverage Area: Section III finds the coverage area of a target where the target possibly stays currently. In the absence of all types of information for node searching, the locator then searches the target's coverage area rather than randomly searching the nearby sub-areas as in DSearching. If the locator moves around itself in searching, it generates high overhead. To handle this problem, the locator then uses the agent-based simultaneous searching scheme to search the coverage area.

5) Summary Of Node Searching: In summary, TSearch uses ERs, friends' ERs, preferred locations, and relation graph for node searching. Since ERs indicate the most recent movement of the target node; the target node's friends' ERs reflect the movement of the target node from the perspective of

its frequently meeting nodes; the preferred locations indicate the target node's frequently visited places; and additional information establishes the long-term relation of nodes, we use them with priorities as: ERs>friends' ERs>preferred locations>additional information. We use algorithm 1 to summarize the node searching process of a locator.

Algorithm 1 Node searching process of locator L_i.

while target node N_t is not found **do**
  if ER of N_t is available **then**
    Locator L_i moves to ER's indicated position;
  else if N_t's friends' ERs are available **then**
    L_i moves to the location indicated in the ER of N_t's most frequently encountered friend;
  else if N_t's preferred locations are available **then**
    L_i moves to the nearest N_t's preferred location;
    L_i asks M agents to search in N_t's top M preferred locations;
  else if Relation graph is available **then**
    L_i refers to the anchor for possible locations of N_t;
    L_i moves to the location with the maximum weight;
  else
    L_i deduces the target node's coverage area from N_t's latest ER;
    L_i uses the agent-based simultaneous searching scheme to search the coverage area;
  end if
end while

After the locator moves to the sub-area of the target node, it may not meet the target node immediately since the target node may be currently out of the locator's transmission range. In this case, the locator relies on the anchor to finish the last step to encounter the target. Since nodes periodically report ERs to the anchor of their sub-area, the anchor knows the exact location of the target node. Therefore, after the locator meets the anchor, it learns the target's current location and moves to this location in the sub-area to finally meet the target node.

D. Role-based Information Collection and Dissemination

Based on the hierarchical structure in previous works [20], [21], we design a role-based information collection and dissemination scheme to enable the information for node searching to be globally accessed. We assign different roles to nodes by their mobility characteristics. For example, stable nodes are likely to encounter many nodes moving in their sub-areas and suitable for collecting mobility information. While nodes frequently transiting among sub-areas are suitable for disseminating mobility information.

1) Role-based Scheme: The role-based scheme selects a relatively stable node with high storage and computing capacity in each sub-area to be "anchor", and selects a number of nodes frequently transiting between two sub-areas as their "ambassadors". Anchors are responsible for collecting the ERs, friends and preferred locations of nodes in different sub-areas. When a node moves into a sub-area, it reports its stored ERs, friends and preferred locations to the sub-area's anchor. An anchor only stores the latest ER of each node. Therefore, once a locator moves into a sub-area, it can quickly access the information of its target from the sub-area's anchor.

An ambassador for sub-areas A_i and A_j are responsible for maintaining the consistency of stored information in the anchors of A_i and A_j. When the ambassador moves from A_i to A_j, it carries the updated and new information (since the last update) in the anchor of A_i to the anchor of A_j. The anchor of A_j then adds the information not in its own storage, and updates the latest ERs. The same applies when the ambassador moves from A_j to A_i. Once a new encounter event happens in a sub-area, the ambassador will carry the new ER to other sub-areas. Thus, a locator can access the information of nodes in remote sub-areas from the local anchor for node searching. In the absence of the target's ER, the locator can use the preferred locations, and the ERs of the target's friends for node searching.

In a DTN, nodes always need to exchange packets with neighbors to identify their neighbors. For a DTN that can afford the overhead of transmitting a few more packets, nodes can piggyback ERs on the exchanged packets to expedite the information dissemination. Then, a locator can quickly receive the ERs of nodes in nearby sub-areas.

2) Role-based Node Selection: We next introduce how to select the anchors and ambassadors. We use the nodes' probability of staying in a certain sub-area to determine whether they can be the anchors of this sub-area. The staying probability of a node, say N_i, at sub-area A_k is defined as $P_{N_i}(A_k) = T_i/T_u$, where T_i is the total time that N_i has stayed in sub-area A_k during a unit time period T_u. If $P_{N_i}(A_k)$ is larger than a high threshold, N_i can be the anchor for A_k. Since the anchor's staying time in its anchoring sub-area should be long so that the node searching request in the sub-area can be fulfilled as much as possible, we set T_u to 1 hour and set the threshold of staying probability for being an anchor to 0.8. By exchanging messages, the nodes with staying probabilities higher than the threshold will become candidate anchors. Then the node with most available resources (e.g., memory, calculating capability) becomes the anchor of a sub-area, and all other candidate nodes become anchor backups. Before the current anchor moves out of sub-area (A_k), it chooses the anchor backup with the highest $P_{N_i}(A_k)$ as the new anchor, transfers all of its information to the new anchor and notifies the nodes in the sub-area about this new anchor.

The ambassadors for two sub-areas, say A_i and A_j, are the nodes that have high frequency of transiting between A_i and A_j. A node records the number of transits between two sub-areas during time period T_u. If this transit probability is larger than a threshold, this node can be an ambassador between these two sub-areas. To maximally stabilize the transfer of mobility information between pairs of sub-areas, we set the threshold of transit probability between two sub-areas for being an ambassador to 0.8. Then, it reports itself as an ambassador candidate to the anchors of the two sub-areas. The anchor only chooses the top 50% of the ambassador candidates with the most memory space and processing capacity as the final ambassadors for corresponding sub-areas.

Anchors and ambassadors may fail during node searching. If an anchor fails, a new anchor will be selected from the anchor backups in the next time period T_u. If an ambassador for two sub-areas fails, a new ambassador will be added by the anchor from the remaining ambassador candidates in the next time period. If there are no remaining nodes suitable for anchor/ambassador, the nodes that are about to move out of the current sub-area collects the mobility information information of neighboring nodes. When the node moves into another sub-area, it broadcasts the collected information to the anchor of

the sub-area.

Figure 13 illustrates how information consistency between sub-areas is maintained by the nodes with different roles. The nodes report their stored ERs, their own friends and preferred locations to their local anchors, which maintain the latest ER for each node. When an ambassador for A_i and A_j moves to A_j from A_i, it retrieves all the information from the anchor of A_i and sends the information to the anchor of A_j. Then, the anchor of A_j updates its information. When the ambassador moves to A_i from A_j, the anchor of A_i updates its information. Thus, the information of all nodes is collected and the information consistency is maintained in all sub-areas.

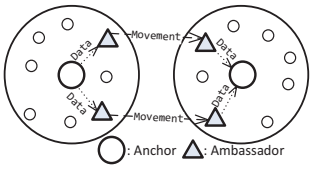


Fig. 13: Information consistency maintenance between sub-areas.

3) Information Collection without Anchors or Ambassadors: There may be no stable anchors or frequently transiting ambassadors. We use the following strategies to handle these situations. If no node is suitable for being an anchor, e.g., all nodes in a sub-area have low staying probabilities, ambassadors or nodes that will transit to other sub-areas will be responsible for collecting and disseminating the information. Specifically, the ambassadors will collect the stored ERs, friends and preferred locations of nearby nodes. Later, the ambassadors will broadcast the collected information to new sub-areas. If no ambassadors are available, that is, no nodes have very frequent transits between two sub-areas, other nodes are responsible for carrying information between sub-areas. Specifically, when a node is about to move out of its present sub-area, it collects information from neighbor nodes it encounters.When the node moves into another sub-area, it will report the collected information to the anchor of the sub-area.

V. Performance Evaluation

We conducted trace-driven experiments based on the DART [16] and DNET [17] traces introduced in Section III. Unless otherwise specified, the experiment setting is the same as that in Section III. Search rate is defined as the number of locators generated every 24 hours in DART and every 4 hours in DNET and was set to 40 by default. Since both traces do not provide map information, we assume a locator needs 10 minutes to move from one sub-area to another neighbor sub-area on average. The average radius of sub-areas is 244 meters and 300 meters in DART and DNET, respectively. The average probability of nodes meeting within the same sub-area is 0.35 and 0.11 in DART and DNET, respectively. ERs cannot reflect the nodes' real-time position after a certain time period. To maintain the freshness of ERs, we set a TTL for ERs and nodes remove ERs after TTL. The expiration TTL for ERs was set to 4 hours and 2 hours in DART and DNET, respectively. The encounter frequency threshold for determining friends was set to 0.6. The thresholds of staying probability and transit probability for determining anchors and ambassadors were set to 0.8. The threshold for determining node closeness is 0.2.

We evaluated TSearch in four varieties: TSearch that uses additional information and has ER exchange (*TS**+), TSearch that uses additional information and has no ER exchange (*TS*+), TSearch that doesn't use additional information and has ER exchange (*TS**), TSearch that doesn't use additional information and has no ER exchange (*TS*). The comparison methods are: the *DSearching* distributed node searching method (*DS* in short) [15], and a routing based method (denoted by *Routing*) [28] as explained in Section I. In order to show the effect of ERs in node searching in TSearch, we also evaluated TSearch that only uses ERs without anchors (denoted by *ER*). That is, nodes record the ERs with their encountering nodes and exchange the records. We measured the following metrics in the experiments.

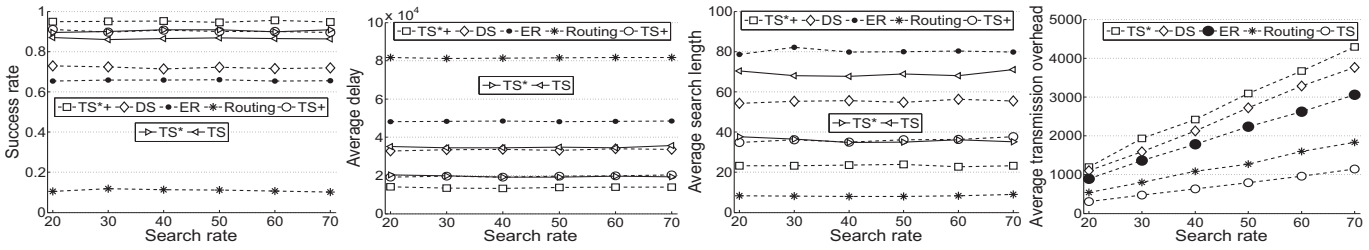
- *Success rate*: The percentage of locators that successfully find their target nodes within searching TTL.
- *Average delay*: The average time (in seconds) used by locators to search for the target nodes. Note that the time spent by unsuccessful locators, which is the searching TTL, is also considered in calculating this metric.
- *Average search length*: The average number of transits between sub-areas that locators move. Note that the transits of unsuccessful locators are also included.
- *Average transmission overhead*: The average number of all packets transmitted among nodes.
- *Average information query*: The average number of information queries received by each node, including the information update caused by encounters. Specifically, the information queries are categorized into request for ER, friend, preferred location and relation graph in TSearch, request for VR, MPT entry in DSearch and request for meeting frequency in Routing.
- *Average node memory usage*: The average number of memory units used by each node. VR, ER records location and time, while each MPT entry, friend, preferred location records location or node id and probability. Since location and node id consumes 32 bits, and time and probability consumes 64 bits [15], we consider each piece of information (i.e., VR, MPT entry, ER, friend, preferred location) has similar size, namely one memory unit.
- *Average anchor memory usage*: The average number of memory units used by each anchor or host node.

Since nodes don't share additional information, so it creates no transmission overhead. Since nodes just use existing information to generate additional information, so additional information consumes no extra memory. Since locators request additional information along with ER, friends and preferred locations, so additional information won't change the locators' query frequency. Based on above considerations, we only illustrate the improvements of success rate, search delay and search length brought by additional information.

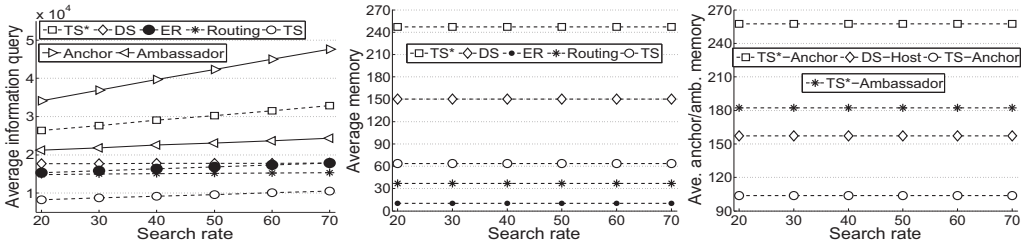
A. Experiments with Different Search Rates and Search TTLs

We conducted two experiments. In one experiment, we varied the search rate from 20 to 70 with 10 as the step size. In the other one, we varied the search TTL from 18 hours to 33 hours in DART and from 2 hours to 7 hours in DNET.

1) Success Rate: Figure 14(a) and Figure 15(a) show the success rates under different search rates in DART and DNET, respectively. Figure 16(a) and Figure 17(a) show the success rates under different search TTLs in

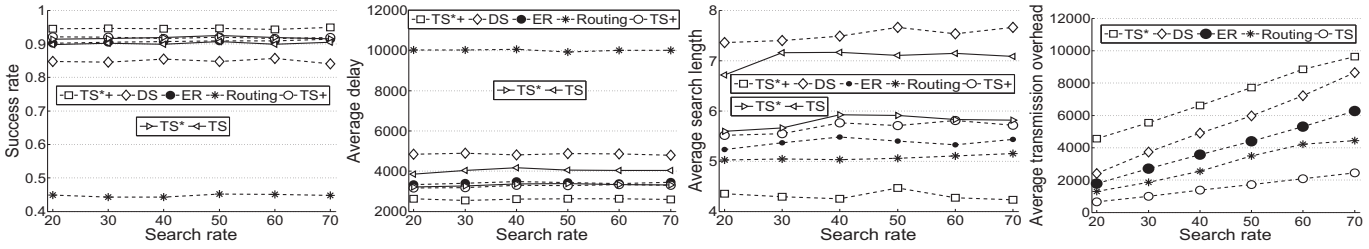


(a) Success rate. (b) Average delay. (c) Average search length. (d) Average transmission overhead.

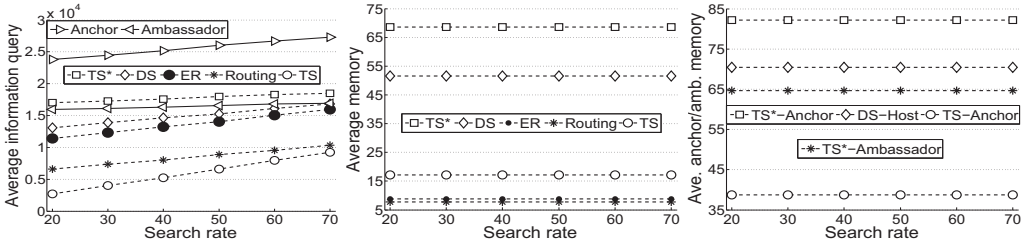


(e) Average information query. (f) Average node memory usage. (g) Average anchor memory usage.

Fig. 14: Performance with different search rates using the DART trace.



(a) Success rate. (b) Average delay. (c) Average search length. (d) Average transmission overhead.



(e) Average information query. (f) Average node memory usage. (g) Average anchor memory usage.

Fig. 15: Performance with different search rates using the DNET trace.

DART and DNET, respectively. We see the results follow: $TS*+ > TS* \approx TS+ > TS > DS > ER \gg Routing$ in DART, and follow: $TS*+ > TS* \approx TS+ \succ TS \succ ER > DS \gg Routing$ in DNET, where $\approx$ means "approximate", $\succ$ means "slightly higher" and $\gg$ means "significantly higher".

In both traces, *Routing* always produces the lowest success rate. This is because the locators do not move proactively to search the targets, but only adhere to nodes that have the highest probabilities of meeting the targets. Since the nodes have independent mobility patterns, many locators fail to find their targets within TTL. Compared with *Routing*, *DS* and *TS* have remarkably higher success rate. *TS* has higher success rate than *DS* in both traces. Because a locator moves directly to the target's latest location in *TS*, while follows the movement path of the target in *DS*. Also, in *DS*, locators must move to the targets' home-areas first, while in *TS*, the locators search from their current locations. Therefore, *TS* enables locators to search more quickly. Moreover, *TS* relies on multiple agents to search the target in its possible locations, while *DS* only considers the place with the highest visiting probability as the next destination. We also see that *TS** generates a slightly higher success rate than *TS* in both traces, which indicates that ER exchanges can slightly improve success rate. Further, *ER* always generates a higher success rate than *Routing*, which indicates that ERs are effective in guiding node searching.

The success rate shows *DS*>*ER* in DART, but shows *ER*>*DS* in DNET. DART has much more nodes and sub-areas than DNET. Then, some locators may fail to receive the ERs of their targets in time, leading to lower success rate. This verifies the effectiveness of ERs for node searching in small networks. It also implies the necessity of anchors to facilitate the global information dissemination, especially in a large network area. We see that *TS** always achieves higher success rate than *ER*, which indicates the effectiveness of the anchors, friends and preferred locations. The similar success rates of *TS** in both traces reflect that the number of sub-areas is not the constraint due to the relatively fast dissemination of ERs. Additionally, we find that except for *TS*+*, *TS**, *TS+* and *TS*, the other three algorithms exhibit obvious improvement in success rate as TTL increases. This verifies the performance of *TS*+*, *TS**,

TS+ and *TS* under small TTL.

With the help of additional information, *TS**+ and *TS*+ always have higher success rate than *TS** and *TS*, respectively. This shows the benefit of additional information for node searching. On the way to the most preferred locations of the target's friends, the target or its information is often found.

2) Average Delay: Figure 14(b) and Figure 15(b) show the average delay under different search rates in DART and DNET, respectively. Figure 16(b) and Figure 17(b) show the average delay under different search TTLs in DART and DNET, respectively. We see the delays follow: *TS**+$<$*TS**$\approx$*TS*+$<$*DS*$\approx$*TS*$<$*ER*$\ll$*Routing* in DART, and *TS**+$<$*TS*+$\prec$*TS**$\prec$*ER*$<$*TS*$<$*DS*$\ll$*Routing* in DNET, where $\prec$ means "slightly smaller" and $\ll$ means "significantly smaller". Higher success rate means more node searches are successful during TTL. Recall we used TTL as delay for failed searches. Thus, the methods with higher success rates produce lower delay while the methods with lower success rates produce higher delay. *TS** generates a lower average delay than *TS* because locators may not need to query ERs from anchors. Under cases with node failure or absence of stable nodes, *TS** can also save some delay for node searching. *TS**+ and *TS*+ have lower average delay than *TS** and *TS*, respectively. This is because additional information helps the locator save much time in finding the target and its direct information. On average, *TS** reduces the average delay of *TS* by 30% and 5900 seconds (1.64 hours) in DART, and by 20% and 651 seconds (11 minutes) in DNET. *TS**+ further reduces the average delay of *TS** by 29% and 5869 seconds (1.63 hours) in DART, and by 21% and 708 seconds (12 minutes) in DNET.

3) Average Search Length: Figure 14(c) and Figure 15(c) show the average search lengths of the algorithms under different search rates in DART and DNET, respectively. Figure 16(c) and Figure 17(c) show the average search lengths of the algorithms under different search TTLs in DART and DNET, respectively. We see that the average search lengths follow: *Routing*$<$*TS**+$<$*TS*+$<$*TS**$<$*DS*$<$*TS*$<$*ER* in DART, and follow: *TS**+$<$*Routing*$<$*ER*$<$*TS*+$<$*TS**$<$*TS*$<$*DS* in DNET.

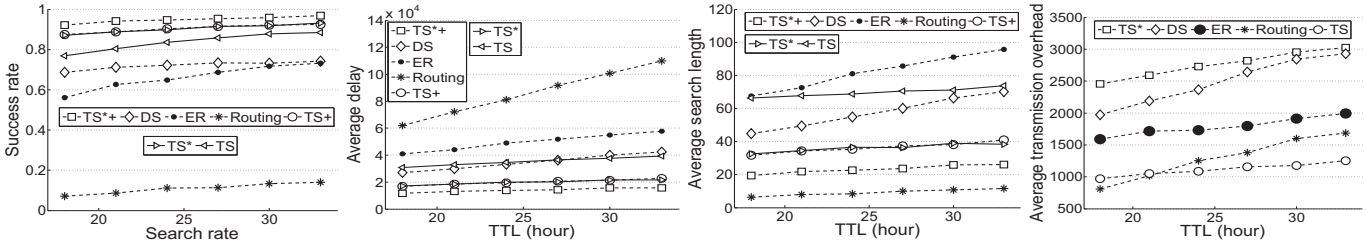
We can see that the search lengths of *TS**+, *TS*+, *TS**, *TS*, *DS* and *ER* are positively correlated with their delays. This is because the locators in these algorithms are always moving to search their targets. However, the delay of *Routing* is very high but the search length of *Routing* is the lowest or medium. This is because the locators in *Routing* cannot proactively move towards the targets but attach to nodes. Moreover, meeting frequency cannot tell locators the most suitable attaching nodes. Therefore, the locators in *Routing* don't transit much but wait on a few nodes.

4) Average Transmission Overhead: Figure 14(d) and Figure 15(d) show the average transmission overhead of the algorithms under different search rates in DART and DNET, respectively. Figure 16(d) and Figure 17(d) show the average transmission overhead of the algorithms under different search TTLs in DART and DNET, respectively. We see the result follows: *TS*$<$*Routing*$<$*ER*$<$*DS*$<$*TS**.

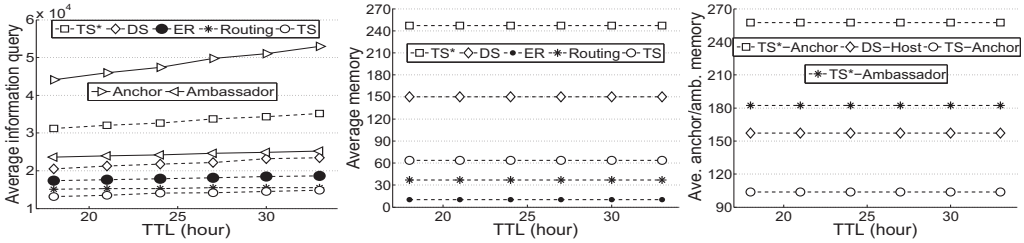
TS has the least transmission overhead because nodes only need to exchange ERs with anchors. Nodes only report their friends and preferred locations once to anchors. Each ambassador carries the information in an anchor only when it moves to another sub-area. *Routing* and *ER* produce higher transmission overheads than *TS* because they require packet exchange between nodes upon entering. In *Routing*, a node keeps its meeting probabilities with other nodes and exchanges this information with its neighbors. In *ER*, a node keeps the ERs, which may or may not involve itself. Therefore, *Routing* produces lower transmission overhead than *ER*. In *DS*, each node tells several neighbors in its current sub-area its transient VR before moving. The node also distributes its MPT from a sub-area to long-staying nodes in this sub-area. Each node's home-area information is stored in all sub-areas. As nodes move around, they transmit many transient VRs. Therefore, *DS* requires nodes to distribute MPT to host nodes of each sub-area, and the locators access the MPT from the host nodes. *TS** has the transmission overheads of both *TS* and *ER*, thus nodes exchange ERs upon encountering and also report their ERs, friends and preferred locations to the anchors once. Therefore, it produces the maximum transmission overhead among the algorithms. On average, *TS* reduces the average transmission overhead of *TS** by 74% and 2047 packets in DART, and by 78% and 5603 packets in DNET. ER exchanges enable nodes to receive ERs more quickly for faster node searching. Recall that *TS** reduces the average delay of *TS* by 20%-30%. Then, TSearch can activate or inactivate the ER exchange function based on applications. Also, nodes can choose to use ER exchanges based on their individual desires.

5) Average Information Query: Figure 14(e) and Figure 15(e) show the average number of information queries of the algorithms under different search rates in DART and DNET, respectively. Figure 16(e) and Figure 17(e) show the average number of information queries of the algorithms under different search TTLs in DART and DNET, respectively. To illustrate the overhead on anchors and ambassadors, we also draw the information query handled by anchors and ambassadors in *TS**, respectively. We see that the results follow: *TS*$<$*Routing*$<$*ER*$<$*DS*$<$*TS** in both traces. From the figures, we can find that the average number of information query increases along with the increment of search rate.

In *TS**, every encounter among nodes will create at least one information query. Besides, the anchors and ambassadors will request ERs, friends and preferred sub-areas from the nearby nodes. Since there may be several ambassadors in one sub-area, they may request repeated information from anchor nodes. Therefore, *TS** achieves the highest number of information queries. Also note that the information query overhead of the anchors in *TS** is the highest, but the information overhead of the ambassadors is lower than the average information query overhead of *TS**. Recall that only one anchor is needed for each sub-area. For long staying nodes with sufficient computing capacity and storage, the overhead is acceptable. Moreover, the information query overhead of ambassadors is comparable to the average information query overhead of *DS* and *ER*. Therefore, the overhead for anchors and ambassadors is acceptable. In *DS*, each node leaves its transient VR to neighbors before moving out of a sub-area, and reports MPT entries to host nodes in the sub-area. Compared with *TS**, the occasion of information query is much lower. But once nodes move in or out of sub-areas, information query is possible, rendering *DS* the second highest information query. In *ER*,

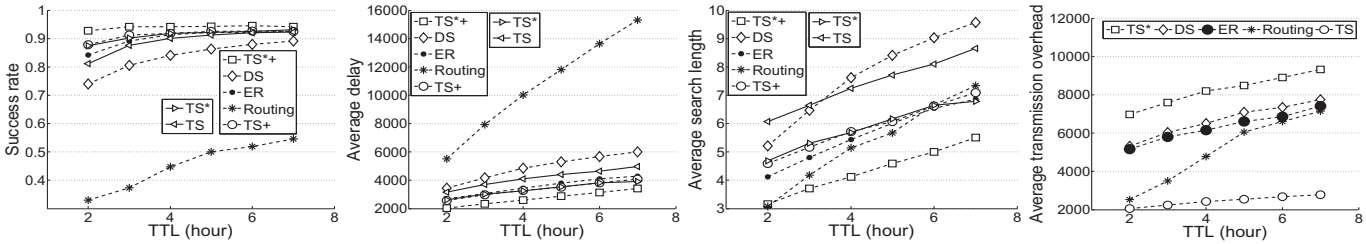


(a) Success rate. (b) Average delay. (c) Average search length. (d) Average transmission overhead.

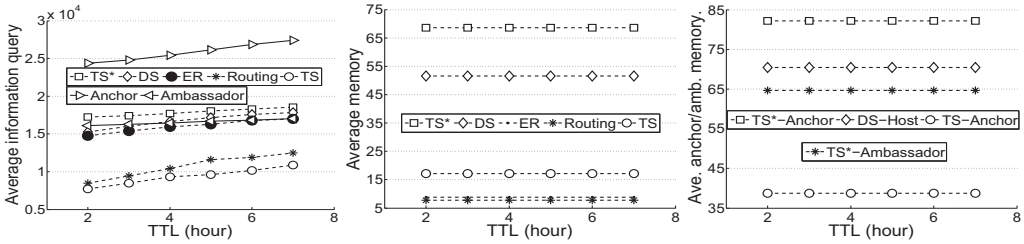


(e) Average information query. (f) Average node memory usage. (g) Average anchor memory usage.

Fig. 16: Performance with different locator TTLs using the DART trace.



(a) Success rate. (b) Average delay. (c) Average search length. (d) Average transmission overhead.



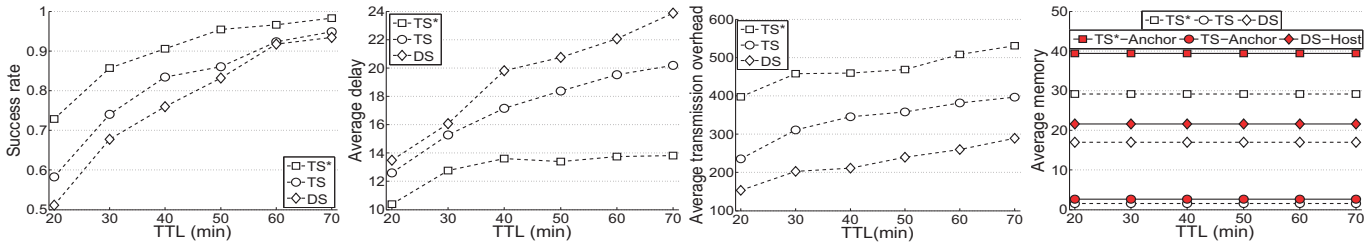
(e) Average information query. (f) Average node memory usage. (g) Average anchor memory usage.

Fig. 17: Performance with different locator TTLs using the DNET trace.

each node only needs to record, exchange and report ERs. The number of information query is determined by the encounter among nodes. Therefore, it ranks the third. In *Routing* and *TS*, each node only records its meeting frequency or ER with other nodes. Therefore, they achieve much less information query than others. Since *Routing* needs to find the nodes with the maximum meeting frequency with the target, while *TS* only checks whether the nodes have the ER of the target, *Routing* has more information query than *TS*.

6) Average Node Memory Usage: Figure 14(f) and Figure 15(f) show the average node memory usage of the algorithms under different search rates in DART and DNET, respectively. Figure 16(f) and Figure 17(f) show the average node memory usage of the algorithms under different search TTLs in DART and DNET, respectively. We see that the average memory usage follows: *ER*<*Routing*<*TS*<*DS*<*TS** in DART and *ER*≈*Routing*<*TS*<*DS*<*TS** in DNET.

In *ER*, each node stores received ERs and deletes them after TTL. In *Routing*, each node stores its meeting probabilities with other nodes. Therefore, the average memory usage of *ER* is less than *Routing* in DART. DNET has a small network and much fewer sub-areas, which enables a node to meet more nodes. Thus, the average memory usage of *ER* is close to *Routing* in DNET. In *TS* and *TS**, normal nodes maintain their ERs, friends and preferred locations, and anchors store such information from nodes. In *TS**, nodes additionally need to exchange ERs. Thus, their average memory usage is higher than *ER* and *Routing*. In *DS*, each node needs to store its home-area and its MPT. The hosts in each sub-area need to store the home-area of each node, the transient VRs and MPT entries of some nodes. In *TS**, each node records and exchanges the ERs of other nodes, and its own friends and preferred locations. Also, the anchors need to store such information. Since the MPT and VRs are collected and stored in hosts, but ERs are generated upon encountering and shared among nodes in *TS**, the memory usage of *TS** is larger than *DS*. The average memory usage of hosts and anchors follows: *TS**>*DS*>*TS* in both traces. *TS**>*DS* is because one sub-area has several hosts but one anchor, and the information is distributed to hosts in different sub-areas but each anchor needs to store information



(a) Success rate. (b) Average delay. (c) Average transmission overhead. (d) Average node memory usage.

Fig. 18: Performance with different TTLs using real environment data.

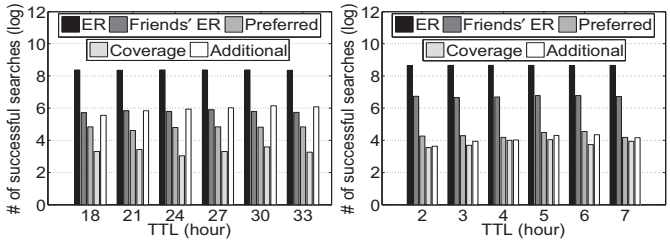
of all nodes. In *TS**, due to ER exchange between nodes, anchors can more quickly receive ERs of far-away nodes. In *TS*, an anchor only has ERs from the nodes in its sub-area. Therefore, anchors in *TS** have higher memory than in *TS*.

We find the peak number of information entries stored on each node is about 210 and 50 in DART and DNET, respectively. Each memory unit takes about 40 bytes. This means the actual average memory usage on each node is only about 8.4KB and 2KB. Based on above analysis, we can conclude that *TS** is applicable on modern mobile devices.

7) Average Anchor/Ambassador Memory Usage: Figure 14(g) and Figure 15(g) show the average anchor/ambassador memory usages under different search rates in DART and DNET, respectively. Figure 16(g) and Figure 17(g) show the average anchor/ambassador memory usages under different search TTLs in DART and DNET, respectively. We see the results follow: *TS-Anchor*<*TS*-Ambassador*<*DS-Host*≪*TS*-Anchor* in DART, while *TS-Anchor*≪*DS-Host*<*TS*-Ambassador*<*TS*-Anchor* in DNET.

In *TS**, anchors can accumulate more global mobility information from other nodes. In *DS*, the host nodes need to store the home-area of other nodes, along with the VRs and MPT entries. However, there're several host nodes per sub-area in *DS* but only one anchor in *TS**. Therefore, anchors in *TS** use more average memory than host nodes in *DS*.

In *TS*, since nodes have no exchange, each anchor only manages local mobility information, and no ambassador is needed. The anchors of *TS** need to maintain mobility information from other sub-areas, and ambassadors have to carry the mobility information of two sub-areas. Moreover, anchors generally stay in their home-area for a long time, leading to limited meeting of nodes. In contrast, the host nodes of *DS* need to maintain VR, MPT and home area of all nodes. Therefore, the anchors in *TS* utilize less memory than the host nodes in *DS*, and the anchors and ambassadors in *TS**.



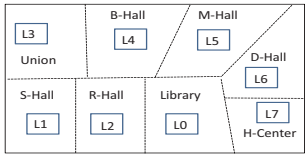
(a) DART. (b) DNET.

Fig. 19: Breakdown of success rate in different search stages (log).

8) Contribution of Different Stages in TSearch: To better illustrate the respective contribution of different search methods on success rate, we break down the total success rate into 5 stages by information: ERs, friends' ERs, preferred locations, coverage area and additional information. Figure 19 shows the ratio of contribution in log format. We see that most of the successful searches are achieved by following the target's ERs. The ERs of the target's friends have the second highest contribution. The target's preferred location offers complementary contribution. The coverage area contributes the least, which means this stage is usually not launched. Results show that additional information is effective in helping locators find the target or its information in node searching. Additional information contributes relatively more in DART than that in DNET, this is because DART has more ambassadors and anchors for each sub-area, the locator in DART is more likely to encounter the target or its information by following additional information.

B. Experiment in Real Environment

We deployed *TSearch* on our campus and collected the mobility information of 9 students from 4 departments in our university. We selected 8 buildings frequently visited by the 9 students as the sub-areas. Based on the GPS on mobile phones, each node can determine its location. Compared with the previous two traces, the distance hence the node movement latency between two sub-areas is more accurate. The distribution of sub-areas and the summary of the data are shown in Figure 20(a) and 20(b). We varied the search TTL from 20 minutes to 70 minutes and set the search rate to 40. According to our campus map, a locator usually takes about 5 minutes to move from one sub-area to a neighboring sub-area.



(a) Maps for sub-area division.

Item	Value
# Sub-areas	8
# Nodes	9
# Transits	147
Duration	4 days

(b) Statistical data.

Fig. 20: Configuration in the real environment.

The test results for different metrics for *TS**, *TS* and *DS* are shown in Figure 18. The orders of the results between the three algorithms are the same as those in the previous experiments due to the same reasons. When the search TTL increases, the success rate, average delay and average transmission overhead increase. When the TTL increases, more locators can find their targets after a longer delay, and along with higher transmission overhead. Also see when the TTL was set to 70 minutes, a successful locator takes only about 14 minutes to find the target node on average. Further, each node only needs 7 memory units on average. In conclusion, *TSearch* is effective and efficient in searching nodes.

VI. Conclusion

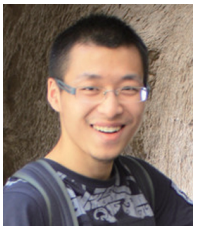
Previous node searching method in DTNs cannot achieve low searching delay by tracing a target along its movement path and also cannot guarantee high success rate by targeting the most frequently visited place (i.e., preferred locations) of the target. Our real trace data analysis confirms these drawbacks and provides foundations for the TSearch. TSearch enables a locator to always move to the target's latest appearance place known by itself, the latest appearance place of the target's most frequently meeting node, or the preferred locations of the target. Also, we design an agent-based simultaneous scheme to increase searching success rate, Even if no direct information of the target is known, the additional information is useful for deducing the target's location. Extensive trace-driven and real-world experiments show that TSearch has much higher efficiency and effectiveness in node searching compared with previous methods. In our future work, we plan to further exploit nodes' social network properties to reduce node searching delay and overhead.

Acknowledgements

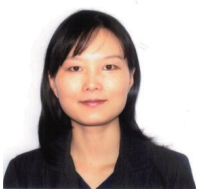
This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, Microsoft Research Faculty Fellowship 8300751.

References

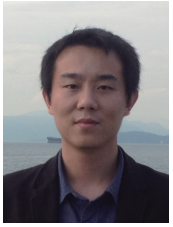
[1] Y.-T. Yu, T. Punihaole, M. Gerla, and M. Sanadidi, "Content routing in the vehicle cloud," in *Proc. of MILCOM*, 2012.
[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Proc. of ASPLOS-X*, 2002.
[3] B. Thorstensen, T. Syversen, T. Walseth, and T.-A. Bjørnvold, "Electronic shepherd - a low-cost, low-bandwidth, wireless network system." in *Proc. of MobiSys*, 2004.
[4] J. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses." in *Proc. of SenSys*, 2005.
[5] J.-H. Huang, L. Jiang, A. Kamthe, J. Ledbetter, S. Mishra, A. Cerpa, and R. Han, "Sensearch: Gps and witness assisted tracking for delay tolerant sensor networks," in *Proc. of Ad Hoc-Now*, 2009.
[6] A. Symington and N. Trigoni, "Encounter based sensor tracking," in *Proc. of MobiHoc*, 2012.
[7] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem." in *Proc. of SIGCOMM*, 2007.
[8] S. Lu, Y. Liu, Y. Liu, and M. Kumar, "Loop: A location based routing scheme for opportunistic networks." in *Proc. of MASS*, 2012.
[9] L. Xie, P. Chong, and Y. Guan, "Leader based group routing in disconnected mobile ad hoc networks with group mobility," *WIRELESS PERS COMMUN*, vol. 71, no. 3, 2013.
[10] W. Gao and G. Cao, "On exploiting transient contact patterns for data forwarding in delay tolerant networks." in *Proc. of ICNP*, 2010.
[11] X. Zhang and G. Cao, "Transient community detection and its application to data forwarding in delay tolerant networks." in *Proc. of ICNP*, 2013.
[12] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of MobiHoc*, 2007.
[13] K. Chen and H. Shen, "SMART: Lightweight distributed social map based routing in delay tolerant networks." in *Proc. of ICNP*, 2012.
[14] X. Tie, A. Venkataramani, and A. Balasubramanian, "R3: robust replication routing in wireless networks with diverse connectivity characteristics," in *Proc. of MobiCom*, 2011.
[15] K. Chen and H. Shen, "DSearching: Distributed Searching of Mobile Nodes in DTNs with Floating Mobility Information," in *Proc. of INFOCOM*, 2014.
[16] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proc. of MobiCom*, 2004.
[17] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang, "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing," in *Proc. of MobiCom*, 2007.
[18] J. Zhao, Y. Zhu, and L. M. Ni, "Correlating mobility with social encounters: Distributed localization in sparse mobile networks," in *Proc. of MASS*, 2012.
[19] K. Chen and H. Shen, "Leveraging social networks for P2P content-based file sharing in disconnected manets." *IEEE TMC*, 2013.
[20] K. Chen, H. Shen, and H. Zhang, "Leveraging social networks for P2P content-based file sharing in disconnected manets." *IEEE TMC*, 2014.
[21] F. Li and J. Wu, "MOPS: Providing content-based service in disruption-tolerant networks." in *Proc. of ICDCS*, 2009.
[22] M. Kim, D. Kotz, and S. Kim, "Extracting a mobility model from real user traces." in *Proc. INFOCOM*, 2006.
[23] L. Xie, P. Chong, and Y. Guan, "Routing strategy in disconnected mobile ad hoc networks with group mobility," *EURASIP Journal on WCN*, vol. 2013, no. 1, 2013.
[24] M. Thomas, A. Gupta, and S. Keshav, "Group based routing in disconnected ad hoc networks," *EURASIP Journal on WCN*, vol. 4297, 2006.
[25] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen, "SMART: A secure multilayer credit-based incentive scheme for delay-tolerant networks," *IEEE TVT*, vol. 58, no. 8, 2009.
[26] K. Chen and H. Shen, "Multicent: A multifunctional incentive scheme adaptive to diverse performance objectives for dtn routing," in *Proc. of SECON*, 2013.
[27] R. Lu, X. Lin, H. Zhu, X. S. Shen, and B. Preiss, "Pi: A practical incentive protocol for delay tolerant networks," *IEEE TWC*, vol. 9, no. 4, 2010.
[28] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks." *Mobile Computing and Communications Review*, vol. 7, no. 3, 2003.



Li Yan Li Yan received the BS degree in Information Engineering from Xi'an Jiaotong University, China in 2010, and the M.S. degree in Electrical Engineering from University of Florida in 2013. He currently is a Ph.D. student in the Department of Electrical and Computer Engineering at Clemson University, SC, United States. His research interests include wireless networks, with an emphasis on delay tolerant networks and sensor networks. He is a student member of IEEE.



Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on content delivery networks, mobile computing, wireless sensor networks, cloud computing, big data and cyber-physical systems. She is a Microsoft Faculty Fellow of 2010, a senior member of the IEEE, and a member of the ACM.



Kang Chen Kang Chen (S'13-M'15) received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2005, the MS in Communication and Information Systems from the Graduate University of Chinese Academy of Sciences, China in 2008, and the Ph.D. in Computer Engineering from the Clemson University in 2014. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Southern Illinois University. His research interests include emerging wireless networks and software defined networking.