DSearching: Distributed Searching of Mobile Nodes in DTNs with Floating Mobility Information

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

***Abstract*—In delay tolerant networks (DTNs), enabling a node to search and find an interested mobile node is an important function in many applications. However, the movement of nodes in DTNs makes the problem formidable. Current methods in disconnected networks mainly rely on fixed stations and infrastructure-based communication to collect node position information, which is difficult to implement in DTNs. In this paper, we present DSearching, a distributed mobile node searching scheme for DTNs that requires no infrastructure except the GPS on mobile nodes. In DSearching, the entire DTN area is split into sub-areas, and each node summarizes its mobility information as both transient sub-area visiting record and long-term movement pattern among sub-areas. Each node distributes its transient visiting record for a newly entered sub-area to nodes that are likely to stay in the sub-area that it just moves out, so that the information flows in the network for the locator to trace it along its actual movement path. Each node also stores different parts of its long-term mobility pattern to long-staying nodes in different sub-areas for the locator to trace it when visiting records are absent. Considering that nodes in DTNs usually have limited resources, DSearching constrains the communication and storage cost in the information distribution process. Extensive trace-driven experiments with real traces demonstrate the high efficiency and high effectiveness of DSearching.**

I. Introduction

In recent years, delay tolerant networks (DTNs) [1], in which nodes are sparsely distributed and no end-to-end connection can be ensured, have attracted significant research interests. In such sparsely distributed networks, enabling a node to search and find the interested node is an important function in node management and many applications. For example, in a DTN that consists of animals (i.e., ZebraNet [2]) for various purposes (e.g., environment monitoring and animal tracking), we may need to find an animal to upgrade or repair the sensor attached to it. For a DTN in battlefield, the system administrator needs to not only isolate a misbehaving device but also find the person who carries the device. In a DTN formed by mobile device users, the node search function can allow a person to find and meet another person. Figure 1 demonstrates the problem of node searching in DTNs, in which the *locator node* searches for the *target node*.

This paper addresses the node searching problem in DTNs embracing social network properties: nodes move with certain patterns and have skewed visiting preferences [3]–[5], which exist in many scenarios. For example, in the DTN consisting of wild animals, animals usually move between different places for food, water, and flock gathering with certain patterns. For DTNs in a battlefield, vehicles and soldiers usually move on specific routes. In a DTN consisting of mobile devices on campus, device holders (i.e., students) visit several buildings, e.g., library, department building, and dorm, repeatedly.

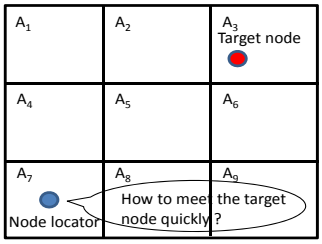


Fig. 1: Demonstration of node searching in DTNs.

There are already some works [2], [3], [6]–[8] for tracking objects (i.e., animal and people) under the context of disconnected mobile networks. In these methods, the position or mobility information of the target object is proactively collected for node tracking/searching. However, these methods require either central stations [2], [7], [8] or infrastructure-based communication (i.e., GPSR [8] and satellite communication [6]) to collect node position information. Though these techniques are possible and even common in certain scenarios, they are costly and impractical in areas where DTNs are mainly designed for, i.e., rural and mountain areas.

Therefore, decentralized node searching method is favorable in DTNs. One intuitive way is to follow the DTN routing algorithms [4], [9]–[14]. These methods deduce a node's probability of meeting other nodes and forward a message to nodes that have higher probability of meeting its destination. Then, the node locator can find the target node by following the movement of the holders of messages destined for the target node. However, this scheme may lead to a long delay because it only provides indirect mobility information of the target node (i.e., other nodes' probability of meeting the target node). Without knowing the direct mobility information of the target node, a node locator cannot move actively toward the target node for efficient node searching.

In DTNs, nodes meet opportunistically with limited communication range, leading to frequent network partitions and

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

posing great challenges on the retrieval of direct node mobility information. Though the target node can be sensed by its neighbors, its position information can hardly be forwarded to the locator quickly. Further, the limited resources on mobile nodes make the broadcasting of mobility information or the storage of all mobility information on every node not applicable in DTNs. As a result, a distributed and lightweight algorithm is desired to distribute node mobility information and make such information easily accessible for locators.

In this paper, we propose DSearching, a distributed and lightweight node searching algorithm in DTNs. The design of DSearching is based on two social network properties in DTNs. First, mobile nodes in DTNs usually exhibit certain movement patterns [5]. Indeed, one previous research reveals that the mobility of mobile devices carried by students in a campus is predictable [15]. Second, nodes often visit certain places and stay there for a relatively long time [4].

In DSearching, the entire DTN area is split into sub-areas, and each node summarizes its mobility information as both transient sub-area visiting record and long-term movement pattern among sub-areas. Both types of mobility information are distributed to selected nodes to control the overhead and meanwhile support effective node searching. The transient sub-area visiting record indicates where the node just visits. Each node distributes its visiting record for a newly entered sub-area to nodes that are more likely to stay in the sub-area it just leaves, so that the visiting records form a chain to enable locators to know the target's movement path. The long-term mobility pattern of a node in sub-area A_i reflects how it transits to next sub-areas in general. Each node distributes its long-term mobility pattern in sub-area A_i to long-staying nodes in A_i and its mobility pattern in its home-area to all other sub-areas. Then, when the visiting records are absent, such information can be used to search for the missing VRs to recover the target node's movement path.

In summary, our contributions are threefold:

(1) We propose a lightweight method to distribute and store node mobility information on mobile nodes, which enables a locator to easily access such information.
(2) We propose a node searching scheme that can efficient and timely find a target in a decentralized manner.
(3) We have conducted extensive trace-driven experiments to show the efficiency and effectiveness of DSearching.

The remainder of this paper is organized as follows. Related work is introduced in Section II. Section III presents the detailed design of DSearching. In Section IV, the performance of DSearching is evaluated through trace-driven experiments. Section V concludes this paper with future work.

II. Related Work

Tracking Objects in Disconnected Networks. Tracking objects in the network without consistent connections has been studied in several previous works [2], [3], [6]–[8]. In SOMA [3], each node utilizes its previous records on encountering nodes and places to predict the places it is going to visit. ZebraNet [2] tracks Zebras in Kenya by configuring tracking collars on them, which record their positions and send the data back to the central station through hop-by-hop broadcasting. Cenwits [7] and SenSearch [8] aim to search people in wilderness areas without a connected network. They both utilize the opportunistic encountering among nodes to forward their position information to stations or access-points in the network. The work in [6] utilizes the flock behavior to reduce costs needed to track sheep in wild areas. It basically lets the flock leader monitor and report the positions of other sheep to the server through GPRS or satellite communication. Different from these methods, DSearching does not require central stations, GPRS or satellite communication to collect position information, but enables a node locator to actively find the target node in a completely distributed manner.

Routing in DTNs. Routing in DTNs has been widely studied in recent years [4], [9]–[14]. These methods can generally be classified into two groups: probabilistic routing methods [9]–[12] and geographical routing methods [4], [13], [14]. In the former group, RAPID [9] and MaxContribution [10] predict a node's future encountering probability with the destination based on previous encountering records and forward packets to nodes with higher probability of meeting their destinations. They also specify the forwarding or storage priorities of each packet based on their delivery probabilities. BUBBLE [11] and SimBet [12] further use social factors (i.e., centrality and similarity with the destination node) to deduce a node's probability of meeting a packet's destination.

In the latter group of methods, GeoDTN [13] predicts node encountering possibility based on previous movement and forwards a packet to nodes that are more likely to meet the destination node. GeoOpps [14] deduces each possible route's minimal estimated time of delivery (METD) to reach the closest point to the destination and forwards packets to nodes that lead to smaller METD. LOOP [4] exploits mobility patterns of mobile nodes to predict their future movement to forward packets to certain areas in the network.

These algorithms only provide indirect information about the target node's position or mobility, i.e., how other nodes meet it. Therefore, they lead to low efficiency on node searching. On the contrary, DSearching uses a novel set of data structures to directly depict a node's mobility information. Therefore, DSearching enables a locator to actively move towards the target node, leading to more efficient node searching.

III. System Design

We assume a DTN with n nodes denoted by N_i ($i = 1, 2, ...n$). We regard the mobility of a node as the transits among sub-areas (different places) in the network. Then, how to decide sub-areas? Clearly, the more sub-areas, the more overhead on maintaining node status, but also the more accurate depict of node mobility. Considering the sub-area division is uniform for all nodes, we decide sub-areas based on popular places that are frequently visited by all nodes.

Specifically, we first select popular places in DTNs, which are common in DTNs with social network properties, e.g.,

libraries and dorms in the DTN on campus. Then, we partition the entire DTN area into sub-areas by below rules:

- Each sub-area contains only 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.

Further, sub-areas with size smaller than a threshold (i.e., minimal sub-area size) are merged with the smallest neighbor sub-area. This is to prevent too many small sub-areas. The popular places and minimal sub-area size are decided by the network/application administrator.

The area partition is completed off-line to generate an area map with sub-areas. Each node is configured with the area map when it initially joins in the DTN with DSearching enabled. Therefore, DSearching needs the application scenario information, i.e., popular places and general node mobility information, before applying it to a DTN. Though all sub-areas in the illustration in this paper have regular shapes, i.e., squares, they can be any shapes. Each sub-area is stored as the positions of its vertexes.

We assume that each node has a GPS. Thus, each node can know the sub-area in which it is located. The GPS is different from the infrastructures since it can easily be installed on mobile devices. Further, a node does not need to query the GPS frequently to save energy, i.e., it can query the position only when finding it may have transited to a new sub-area.

A. Representation of Node Mobility

In DSearching, each node records its mobility information for others to search for it. To enable the locator to know its actual movement path, each node leaves transient sub-area visiting records as hints along the movement path. This is like the phenomenon in which an ant leaves a trail of pheromone as it looks for food for others to follow. To enable the locator to find its regular movement pattern, each node also creates a mobility pattern table (MPT) to summarize its staying in and transits among different sub-areas.

1) Transient Visiting Record: Each node leaves a hint, i.e., visiting record (VR), when it enters a new sub-area:

$$VR :< N_i, A_{new}, A_{prev}, Time, T_s, Seq >$$

where N_i is node ID, A_{new} and A_{prev} are the newly entered sub-area and the previous sub-area, respectively, $Time$ denotes the time when the node finds that it enters A_{new}, T_s is the TTL of the VR, and Seq is the sequence number. Seq increases by 1 when a new VR is created. Considering these records are time sensitive, we often set T_s to a relative short period of time, i.e., half day or one day. The visiting record is designed to ensure that once the locator arrives at A_{prev}, it knows that the target node has moved from A_{prev} to A_{new}.

2) Long-Term Mobility Pattern: A node's MPT contains two types of information for each of its regularly visited sub-areas: staying probability and transit probabilities. This is because nodes usually present skewed preferences on staying at certain sub-areas or transiting from a sub-area to another. The staying probability of a node, say N_i, at a sub-area indicates how likely the node is in the sub-area and is calculated as

$$Ps_i(A_m) = D_m/D \quad (1)$$

where D_m represents the total time N_i has stayed in sub-area A_m and D is the period of time the node has lived.

The transit probability of a node, say N_i, at a sub-area, say A_m, describes its probability to transit to another sub-area, say A_n, in next movement, denoted $Pt_i(A_m \rightarrow A_n)$. It is calculated as below:

$$Pt_i(A_m \rightarrow A_n) = T_{mn}/T_{ma} \quad (2)$$

where T_{mn} is the number of occurrences that the node has transited from A_m to A_n and T_{ma} denotes the total number of occurrences that the node moves away from A_m.

TABLE I: Mobility pattern table on a node.

Rank	Sub-area	Staying Prob.	Next sub-areas and probabilities	Seq
1	A_5	0.50	A_8(0.8), A_2(0.2)	1
2	A_2	0.25	A_1(0.7), A_3(0.3)	1
3	A_6	0.15	A_5(1)	1
4	A_8	0.08	A_7(0.6), A_9(0.4)	1
...	...	...	...	...

After accumulating sufficient movement records, each node builds a MPT as shown in Table I. In the table, "Sub-area" records the regularly visited sub-areas of the node, which are sorted in descending order of the staying probability. In the row for a sub-area, say A_m, the "Next sub-areas and probabilities" records the probabilities that the node moves from A_m to the corresponding sub-areas. For example, $P_{ti}(A_2 \rightarrow A_1) = 0.7$. "Seq" is the sequence number of the table. It is increased by 1 when the table is updated. Nodes update their MPTs periodically.

B. Mobility Information Distribution

We assume nodes are non-malicious and are willing to carry the mobility information. The motivation of nodes to follow rules in DSearching is out of the scope of this work.

1) Distribute Visiting Record: When a node, say N_i, moves from sub-area A_m to sub-area A_n, it creates a visiting record as introduced in Section III-A1, denoted VR_{imn}. Recall that the visiting record is to enable the locator to know the movement path of the target node. Following this direction, we distribute VRs to nodes to ensure that the discovery probability that the locator can find them in the previous sub-area (e.g., A_m for VR_{imn}) is larger than a threshold Th_d.

Specifically, we use $Pd_i(A_m)$ to denote the discovery probability to find N_i's visiting record in A_m. Then, we copy VR_{imn} to nodes in A_n that are likely to transit to A_m and have a high probability to stay in A_m to satisfy that

$$Pd_i(A_m) = 1 - \prod_{r=1}^{k}(1 - Pt_r(A_n \rightarrow A_m) * Ps_r(A_m)) \geq Th_d \quad (3)$$

where k is the number of selected nodes and $Pt_r(A_n \rightarrow A_m)$ and $Ps_r(A_m)$ denote the transit probability from A_n to A_m and the staying probability in A_m of the r-th node, respectively. Therefore, we set Th_d to 0.8 in this paper.

Finally, whenever a node moves from one sub-area to another sub-area, a visiting record is created to leave hint in the previous sub-area. These hints form a linked VR chain to help the locator find the target node gradually and effectively.

2) Distribute the MPT: DSearching utilizes the storage on mobile nodes to store the MPT in a distributed manner for node searching. To save storage, a node, say N_i, needs to choose a subset of nodes to store its MPT so that

- The locators for N_i can easily retrieve the MPT of N_i;
- The overhead for distributing N_i's MPT is controlled.

We take advantage of node mobility pattern and the mobility of locators to realize the two goals. The general method is to ensure that only the necessary part of MPT remains relative stable in each sub-area for efficient retrieval.

Storage Host List: DSearching is proposed for DTNs with social network property that each node has several long staying places [4]. We then define hosts of sub-area A_m as nodes with staying probability in it larger than a threshold.

Then, to store a node's MPT in a sub-area, a certain number of hosts of the sub-area are selected to guarantee that the probability that at least one host holding the MPT stays in the sub-area is larger than a threshold (Th_t). In detail, each node maintains a *host list* for each sub-area containing the hosts in the sub-area that can store its MPT, as shown in Table II.

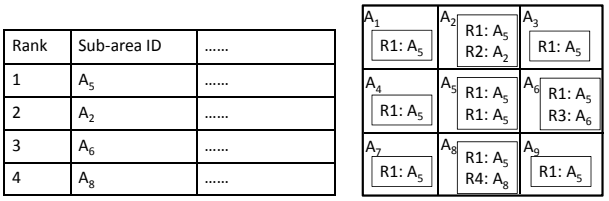
TABLE II: Host list for each sub-area.

Sub-area	Host list	Staying prob.	MPT staying prob.
Sub_1	N_1, N_2, N_4	0.90, 0.8, 0.7	0.994
Sub_2	N_9	0.99	0.99
...			
Sub_M	N_3, N_4	0.6, 0.7	0.88

The "MPT staying prob." denotes the probability that at least one copy of MPT is in the sub-area and is calculated by $1-\prod_{r=1}^{k}(1-Ps_r(A_m))$, where k is the number of hosts and $Ps_r(A_m)$ is the r-th host's staying probability in A_m.

Each node determines its host lists individually while moving in the network. Specifically, suppose N_i tries to decide whether N_j can be added to its host lists. For each sub-area in the table, N_j is temporarily added to the host list for the sub-area when it satisfies: 1) N_j is a host of the sub-area with available memory, and 2) is not in the host list of the sub-area. Then, if the MPT staying probability of the sub-area is smaller than Th_t, no further action is needed. Otherwise, the host with the least staying probability is removed until the MPT staying probability has the minimum value no less than Th_t. The host list of each sub-area can also be determined off-line based on node movement pattern (similar to the sub-area division process). We set $Th_t = 0.7$ in this paper.

MPT Distribution and Update: After creating the host lists from either on-line or off-line method, each node distributes its MPT to the nodes in the host list for each sub-area when encountering them. In this step, DSearching does not store the entire MPT on each node. Instead, only a part of the MPT is stored on a node. Since the locator searches in a sub-area by sub-area manner, the MPT in each sub-area only needs to ensure that the locator knows where to search in the next step.



(a) Mobility pattern table (b) Mobility information distribution

Fig. 2: Distribution of MPT entries (R1 denotes Row 1).

For example, based on Table I, a locator in A_5 only needs to know the row for A_5 in the target's MPT in order to know where to search in the next step.

Therefore, when N_i meets a node in its host lists, say N_j, N_i decides the content to be stored in N_j as below.

- N_i copies the first row of its MPT to N_j.
- If the home sub-area of N_j, denoted $hSub(N_j)$, exists in N_i's MPT, N_i copies the corresponding row to N_j.
- When a node receives the row of a MPT that it already has, it only keeps the latest one.

We define a node's home sub-area as the sub-area it has the highest staying probability. The MPT row for the home sub-area is distributed to all sub-areas because it is the sub-area in which the target node is most likely to stay. Figure 2 shows an example of the distribution of a MPT. We see that in addition to the first row in the MPT, the hosts in sub-areas A_5, A_2, A_6 and A_8 store the corresponding rows in N_i's MPT.

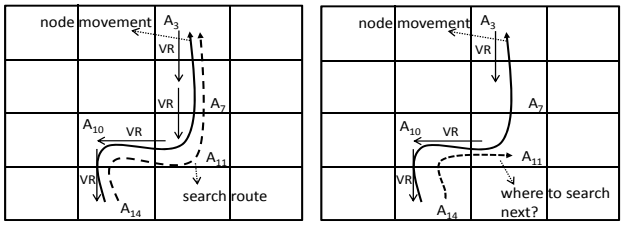
Finally, this strategy can realize the aforementioned two goals. Firstly, wherever a locator starts searching for the target node, it can easily retrieve the mobility pattern information for efficient node searching. Secondly, each node only stores part of its MPT on a selected set of nodes, which saves communication and storage cost on the MPT distribution.

C. Node Searching

In node searching, we assume that the locator can move much faster than the target node, i.e., can pass more sub-areas in a unit time in average. This is reasonable because the locator is dedicated for node searching while the target node may stay at certain places and is not continuously moving. We also assume that when the locator arrives at a sub-area, it can search around to determine whether the target node is in the sub-area with a very high accuracy.

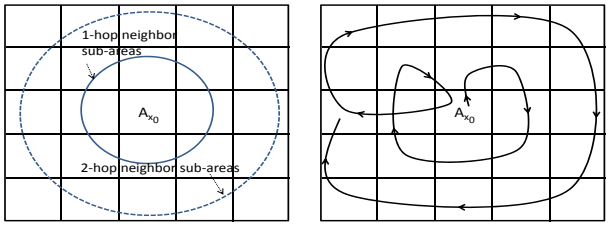
1) Overview: DSearching utilizes the visiting records and MPT tables distributed in the network to search for the target node. When a locator is initialized, it first searches the home sub-area of the target. Then, the locator tries to follow the VR chain to search for the target node along its actual movement path. When a visiting record on the VR chain cannot be found, i.e., there is a gap on the VR chain, DSearching uses the MPT to search for a valid VR that can bridge the gap. During this process, whenever a valid visiting record is obtained, the locator moves to the A_{new} indicated in it. This process repeats until the target node is found.

2) Searching Startup: When a locator is initialized, it knows nothing about the mobility of the target node and



(a) Ideal situation. (b) A VR is missing.

Fig. 3: Node searching with VRs.



(a) N-hop neighboring sub-areas. (b) A searching route.

Fig. 4: Node searching without VRs.

can only search randomly. However, as mentioned in Section III-B2, the first row (home sub-area) of each node's MPT table is copied to all sub-areas. Therefore, the locator can easily know the target's home sub-area. Then, the locator moves to the home sub-area of the target to search for it. The rationale behind such a design is that the target stays in the home sub-area longer than in any other sub-areas.

3) Node Searching with VRs: Whenever the locator discovers one or more VRs of the target node that are newer than the previous one it uses, it moves to the A_{new} in the latest VR, i.e., the one with the largest sequence number, to search for the target node. In this sub-area, if the target node cannot be found, the locator is supposed to discover the VR indicating where the target node moves to from the sub-area. When the locator finds such a VR, it again goes to search the A_{new} in the VR. Ideally, following this manner, the locator searches for the target node along its movement path indicated in a chain of VRs. As shown in Figure 3(a), the locator searches along the actual movement path of the target with the help of VRs, i.e., $A_{14} \rightarrow A_{10} \rightarrow A_{11} \rightarrow A_7 \rightarrow A_3$.

4) Node Searching without VRs: Although the design in Section III-B1 requires that a VR should exist in the previous sub-area with a high probability ($\geq Th_d$), the VRs actually are floating in the network due to node mobility. Therefore, it is common that a certain VR cannot be discovered by the locator, leading to a gap on the VR chain. As shown in Figure 3(b), the VR created in sub-area A_7 fails to reach A_{11}. Then, after searching A_{11}, the locator cannot know where to search for the next step. In this case, the MPT table and geographical limitations are jointly considered to provide a heuristic solution with a low cost.

Specifically, suppose the locator fails to find the expected VR in a sub-area, say A_{x_0}. Then, the locator moves around to find the missing VR or a VR created after it. We define the cost in this process as the expected number of searching hops, denoted $\mathcal{W}$. One searching hop refers to the movement from one sub-area to another sub-area. Then, the goal is *minimizing the $\mathcal{W}$ needed to find a VR that can bridge the gap on the VR chain.* In below, we first assume that when arriving at a sub-area, the locator can obtain the corresponding row for the sub-area in the target's MPT from nearby nodes, which can easily be realized based on the design in Section III-B2. We then present the case without MPT information later.

A Practical Method: In order to find a practical and effective method to find the missing VR, we first investigate the geographical limitation. We define H-hop neighbor sub-area set of a sub-area, say A_{x_0}, as the set of sub-areas that a node needs at least H hops to reach them from A_{x_0}. In Figure 4(a), the 1-hop and 2-hop neighbor sub-areas of A_{x_0} are connected by two circles, respectively. We can find that after moving out of A_{x_0}, the target node needs to visit at least one H-hop neighbor ($H = 1$ or 2) sub-area to pass through the area covered by these sub-areas. Therefore, it is highly possibility that a VR that can bridge the gap on the VR chain can be found in these sub-areas. We then limit the searching for VRs within 1-hop and 2-hop neighbor sub-areas. In detail, we let the locator searches all 1-hop neighbor sub-areas first and then all 2-hop neighbor areas.

Furthermore, we let the locator only searches sequentially along the circle connecting all H-hop sub-areas, i.e., the circles in Figure 4(a). This is because the movement from one H-hop sub-area to another non-neighboring H-hop sub-area needs at least 2 and at most $2 * H$ hops, while the sequential searching takes only Q_H hops to search all sub-areas, where Q_H is the number of sub-areas in the set ($Q_1 = 8$ and $Q_2 = 16$ in Figure 4(a)). Such a simplification can greatly reduce the complexity of finding the best route to find the missing VR.

With above simplification, when searching the H-hop neighbor sub-area set, we only need to determine the start sub-area and the searching direction, which has only $2 * Q_H$ cases, i.e., Q_H possible start sub-areas and each has two directions. Specifically, for 1-hop neighbor sub-area set, the probability that a VR that can bridge the gap in the VR chain can be found in A_{x_r} ($r \in [1, M]$), denoted $Pf(A_{x_r})$, can be expressed as

$$\begin{aligned} Pf(A_{x_r}) &= Pv(A_{x_r})Pd(A_{x_r}) \\ &= Pt(A_{x_0} \rightarrow A_{x_r}) * Th_d \end{aligned} \quad (4)$$

where $Pv_(A_{x_r})$ denotes the probability that the target has visited A_{x_r} after moving out of A_{x_0}, and $Pd(A_{x_r})$ denotes the probability that the VR created in the sub-area immediately after A_{x_r} can be found in A_{x_r}, which is introduced in Equation 3. Then, $Pv(A_{x_r})$ and $Pd(A_{x_r})$ are approximated by $Pt(A_{x_0} \rightarrow A_{x_r})$ and Th_d, respectively. For the former, the simplification only considers the 1-hop transit from A_{x_0} to A_{x_r}, i.e., $Pt(A_{x_0} \rightarrow A_{x_r})$, because 1) the 1-hop transit accounts for the majority of all transits and 2) this is the only information the locator can get from the target's MPT. For the latter, since DSearching requires that $Pd(A_{x_r})$ is larger than Th_d, which is set to a large number, e.g., 0.8, it is acceptable to use Th_d to represent $Pd(A_{x_r})$. Then, DSearching calculate

the $\mathcal{W}$ for all the $2 * Q_1$ cases by below.

$$\begin{aligned}\mathcal{W} &= \sum_{r=1}^{M} S(A_{x_{r-1}}, A_{x_r}) * (\prod_{s=1}^{r-1}(1 - Pf(A_{x_s})))Pf(A_{x_r}) \\ &= (\prod_{s=1}^{r-1}(1 - Pf(A_{x_s})))Pf(A_{x_r}) \quad (5)\end{aligned}$$

where $S(A_{x_{r-1}}, A_{x_r})$ denotes the number of hops needed to move from $A_{x_{r-1}}$ to A_{x_r}. It equals to 1 with our aforementioned simplification. Finally, the start sub-area and searching direction that lead to the minimal $\mathcal{W}$ is selected.

After searching all 1-hop neighbor sub-areas, if a VR that can bridge the gap on the VR chain is not found, the locator then searches the 2-hop neighbor sub-area set. This process is the same as that for the 1-hop neighbor sub-areas except the calculation of $Pf(A_{x_r})$. In this case,

$$Pv(A_{x_r}) = \sum_{r=1}^{R_r} Pt(A_{x_0} \rightarrow A_{y_r}) * Pt(A_{y_r} \rightarrow A_{x_r}) \quad (6)$$

where A_{y_r} denotes the intermediate 1-hop sub-area through which the target node can move from A_{x_0} to A_{x_r}, and R_r is the number of such sub-areas. Then, the start sub-area and searching direction that lead to the minimal $\mathcal{W}$ is selected to search all 2-hop neighbor sub-areas. Note that $Pt(A_{y_r} \rightarrow A_{x_r})$ can be obtained in the searching of 1-hop sub-areas.

Figure 4(b) demonstrates the searching of the 1-hop and 2-hop neighbor sub-areas. During this process, if a VR that can bridge the gap on VR chain can be found, the locator simply follow the VR to continue the search. If not, the locator moves randomly out of the areas covered by the 1-hop and 2-hop neighbor sub-areas to search for VRs.

Without MPT Information: Due to node mobility, it is possible that the MPT information in a 1-hop or 2-hop neighbor sub-area cannot be obtained, which means that the locator cannot know corresponding $Pt(A_{x_0} \rightarrow A_{x_r})$. In this case, we simply take $Pt(A_{x_0} \rightarrow A_{x_r})$ as the average value, i.e., all transits have the same possibility.

D. Summary of the Behaviors of Nodes and Locators

We further summarize the behaviors of nodes and locators in DSearching. For mobile nodes, they first collect enough movement records to create the mobility pattern table as introduced in Section III-A2. Meanwhile, when a node enters a new sub-area, it creates a visiting record as introduced in Section III-A1. Both visiting records and mobility pattern tables are distributed to nodes in the network following the methods in Section III-B1 and III-B2, respectively.

The locator first moves to the home sub-area of the target to search for the target, as introduced in Section III-C2. Then, from the home sub-area, the locator follows the VR chain to search for the target along its actual movement path, as introduced in Section III-C3. In case an expected VR cannot be found, i.e., there is a gap on the VR chain, the locator follows the practical method in Section III-C4 to find such a VR. Once an expected VR is found, the locator again moves along the VR chain to search for the target node.

IV. Performance Evaluation

In this section, we evaluate the performance of DSearching with two real DTN traces and the trace obtained by 9 students carrying a mobile phone on campus.

A. Empirical Datasets

The first trace, denoted Dartmouth trace (DART) [16], records the association of students' digital devices with APs on the Dartmouth campus. We regarded each building as a sub-area and merged neighboring records for to the same mobile device and the same sub-area. We also removed short connections ($< 200s$) and nodes with few records (< 500). Finally, we obtained 320 nodes and 159 sub-areas.

The second trace, namely DieselNet AP trace (DNET) [17], collects the AP association records of 34 buses in the downtown area of a college town. Since there are many APs that do not belong to the experiment in the outdoor environment, APs with few appearances (< 50) were removed from the trace. We mapped APs that are within certain distance ($< 1.5km$) into one sub-area. The trace is pre-processed similarly as the DART trace. Finally, we obtained 34 nodes and 18 sub-areas.

The characteristics of the two traces are shown in Table III.

TABLE III: Characteristics of mobility traces.

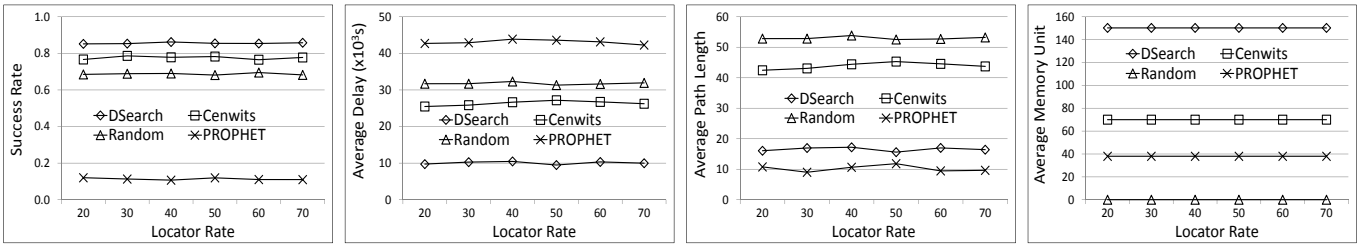
	DART	DNET
# Nodes	320	34
# Sub-areas	159	18
Duration	119 days	20 days
# Transits	477803	25193

B. Experiment Setup

We set the initial period to 30 days for the DART trace and 5 days for the DNET trace, during which nodes collect mobility information to build the MPT. Then, locators were generated with random start sub-area and target node at the rate of R_p per day, which was set to 40 by default. Considering students move less frequently than the bus, the default locator TTL was set to 24 hours in the DART trace and 4 hours in the DNET trace. Since both traces do not provide the map information, we assume that the locator needs 10 minutes to move from one sub-area to another sub-area on average.

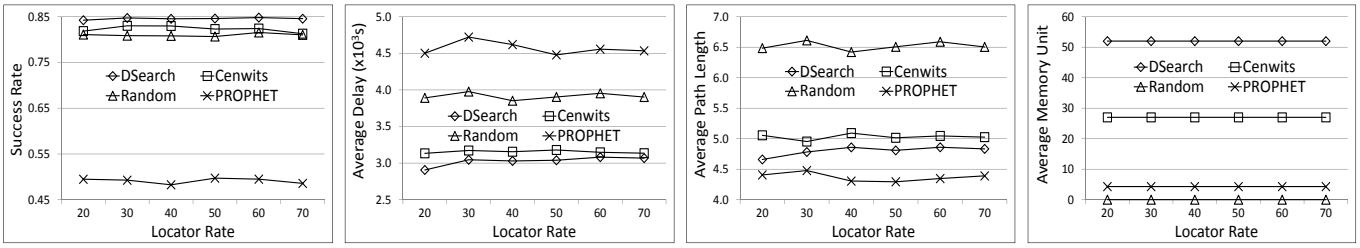
We compared DSearching with three representative methods: an encountering based method (Cenwits) [7], a routing based method (PROPHET) [18], and a random searching method (Random). In Cenwits, nodes record their meeting locations and times with other nodes and exchange such information with others. The locator collects such information from encountered nodes and moves to the most recent place where the target node appears to search for it. In PROPHET, the locator follows the node that has the highest possibility to meet the target node to search for it. In Random, the locator moves randomly to search for the target node.

We measured four metrics: *Success rate*, *Average delay*, *Average path length*, and *Average node memory usage*. The former three refer to the percentage of locators that successfully find their target nodes and the average delay and average



(a) Success rate. (b) Average delay. (c) Average path length. (d) Average node memory usage.

Fig. 5: Performance with different locator rates using the DART trace.



(a) Success rate. (b) Average delay. (c) Average path length. (d) Average node memory usage.

Fig. 6: Performance with different locator rates using the DNET trace.

path length of these locators, respectively. The last one denotes the average number of memory units used by each node. For DSearching, we take one row of the MPT and four visiting records as one memory unit. For Cenwits, 4 meeting records with others are regarded as one memory unit. For PROPHET, 8 meeting probabilities are regarded as one memory unit. We set the confidence interval to 95% in the paper.

C. Experiments with Different Locator Rates

In this test, we varied the locator rate R_p from 20 to 70.

1) Success Rate: Figure 5(a) and Figure 6(a) present the success rates of the four methods in the tests with the DART trace and the DNET trace, respectively. We see that the success rates follow: DSearching>Cenwits>Random>PROPHET.

PROPHET has the lowest success rate because locators in it only follow the mobile node that has the highest probability to meet the target node, which has its own mobility pattern. Therefore, a lot of locators expire due to TTL, leading to the lowest success rate. For Random, locators move actively but search blindly, resulting in a low success rate. Cenwits explores the witness of the target node's appearances in different sub-areas to actively search for it. Therefore, it has higher success rate than Random and PROPHET. However, Cenwits has lower success rate than DSearching. This is because Cenwits only simply utilizes the recent appearances of the target node but neglects the mobility pattern of the target node. On the contrary, DSearching combines the two information to enable more efficient and accurate node searching.

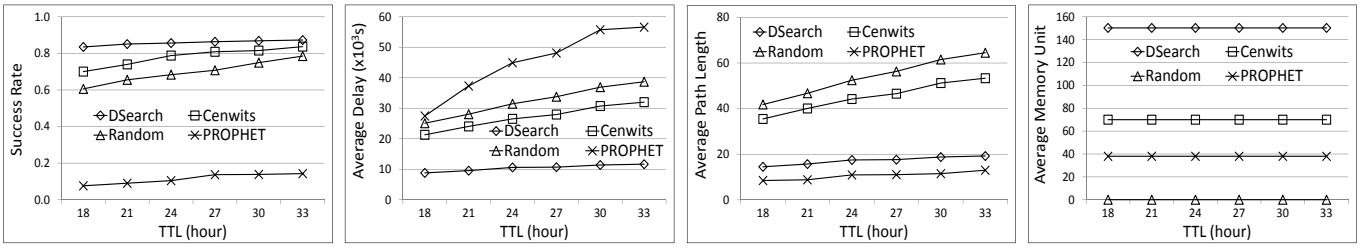
We also find that except DSearching, other three methods have obvious higher success rate in the test with the DNET trace than in the test with the DART trace. This is because the DART trace represents a network with a lot of sub-areas (i.e., 159) while the DNET trace is a small scenario with 18 sub-areas. Then, in the test with the DNET trace, though locators in Random, PROPHET, and Cenwits have limited information about the mobility of the target nodes, they can meet the target nodes easily. DSearching has similar success rate in the tests with both traces because the locator always moves towards the most possible sub-area where the target node would be. Such a result validates the effectiveness of the mobility information distribution in DSearching in networks with different sizes.

2) Average Delay: Figure 5(b) and Figure 6(b) present the average delays of the four methods in the tests with the DART trace and the DNET trace, respectively. We see that the average delays follow: DSearching<Cenwits<Random<PROPHET.

PROPHET has the highest average delay because the node followed by the locator may stay in certain sub-areas for a long time, resulting in an extremely long delay. Random also has a large average delay since nodes search randomly. Cenwits actively searches where the target node has shown recently. Since nodes usually would stay in a sub-area for a while, Cenwits has a small average delay. For DSearching, it further reduces the delay by utilizing both visiting records and MPT to guide node searching, which can help predict where the target node would be more accurately, leading to the least delay.

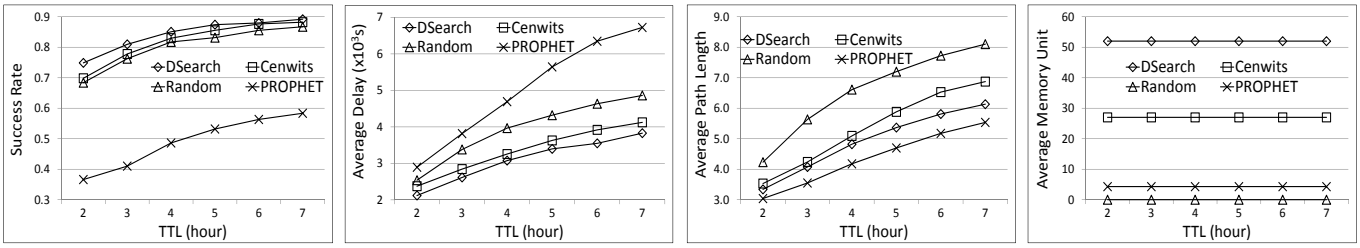
3) Average Path Length: Figure 5(c) and Figure 6(c) show the average path lengths of the four methods in the tests with the DART trace and the DNET trace, respectively. We see that the average path lengths of the four methods follow: PROPHET<DSearching<Cenwits<Random.

PROPHET has the lowest average path length. This is because locators in it often stay in a sub-area for a long time. Therefore, successful locators in PROPHET may only search a few sub-areas, leading to the shortest search path. DSearching has the second least average path length because the locator movement in it has the highest possibility of encountering the target node by utilizing both transient visiting records and long term mobility pattern of the target node. For Cenwits, it only utilizes the transient appearance records of the target node to guide the node searching, resulting in less efficient locator movement and relative large average searching path length. Random has the highest path lengths because the locator



(a) Success rate. (b) Average delay. (c) Average path length. (d) Average node memory usage.

Fig. 7: Performance with different locator TTLs using the DART trace.



(a) Success rate. (b) Average delay. (c) Average path length. (d) Average node memory usage.

Fig. 8: Performance with different locator TTLs using the DNET trace.

searches randomly in the network.

4) Average Node Memory Usage: Figure 5(d) and Figure 6(d) show the average node memory usages of the four methods in the tests with the DART trace and the DNET trace, respectively. We see that the average memory units of the four methods follow: Random<PROPHET<Cenwits<DSearching.

Random has 0 average memory unit since nodes do not need to store any information for node searching. In PROPHET, each node needs to store its meeting probabilities with all other nodes, leading to a small amount of memory units. Cenwits has higher average memory units than PROPHET since it needs to store a large amount of node appearance records on each node. For DSearching, each node stores both visiting records and MPT of other nodes, resulting in the highest memory usage.

Though DSearching has the most memory usage among the four methods, the absolute amount of memory usage is acceptable. We find that the average memory unit on each node is about 150 and 50 in the tests with the two traces. Recall that each unit is about 200 bytes (i.e. one row of the MPT and four visiting records). This means the average memory usage on each node is only about 30 KB and 10 KB in the two tests, which can easily be satisfied in modern devices. Therefore, we conclude that the designed mobility information distribution algorithm is memory efficient in DTNs.

D. Experiments with Different Locator TTLs

We varied the TTL of each locator to see how different methods scale to locator TTL. Considering the DART trace is much longer than the DNET trace (119 days vs. 20 days) and has relative slow node movement (students vs. bus), we varied the TTL from 18 hours to 24 hours and from 2 hours to 7 hours for the DART trace and the DNET trace, respectively.

1) Success Rate: Figure 7(a) and Figure 8(a) present the success rates of the four methods in the tests with the DART trace and the DNET trace, respectively. We see that the success rates of the four methods follow: DSearching> Cenwits>Random>PROPHET. This is the same as that in Figure 5(a) and Figure 6(a) for the same reasons.

We further find that when the TTL increases, Cenwits and PROPHET have closer and closer success rate with DSearching. This is because when TTL is large, locators in Cenwits and PROPHET can also eventually find most target nodes, leading to a high success rate. However, this comes at the cost of high average delay, as shown in next section.

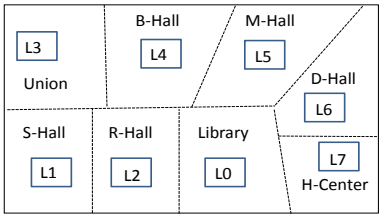
2) Average Delay: Figure 7(b) and Figure 8(b) present the average delays of the four methods in the tests with the DART trace and the DNET trace, respectively. We find that the average delays of the four methods follow the same relationship as in Figure 5(b) and Figure 6(b): DSearching <Cenwits<Random<PROPHET. The reasons are the same with those in the tests with different locator rates.

We also see that when the TTL increases, the average delay increases. This is because when the TTL increases, locators that may fail to find their target nodes when TTL is small can find their target nodes. Then, the average delay of locators increases due to more successful locators with a large delay.

3) Average Path Length: Figure 7(c) and Figure 8(c) present the average path lengths of the four methods in the tests with the DART trace and the DNET trace, respectively. We find that the average path lengths follow: Random< DSearching<Cenwits<PROPHET, which is the same as in Figure 5(b) and Figure 6(b). The reasons are also the same.

We also see that when TTL increases, the average path length increases. This is caused by the same reason as in Section IV-D1 and IV-D2: when TTL increases, more locators can find their targets after searching many sub-areas, leading to increased average search path length.

4) Average Node Memory Usage: Figure 7(d) and Figure 8(d) present the average memory units a node uses in the tests with the DART trace and the DNET trace, respectively. We see that the results are the same with Figure 5(d) and Figure 6(d). This is caused by the fact that the memory usage



(a) Maps for sub-area division.

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

(b) Statistical data.

Fig. 9: Configuration in the real environment.

is independent with the locator TTL.

Combining all above results, we conclude that DSearching presents superior performance compared to other methods with different locator rates and TTLs. Such a result justifies our design goal: efficient node searching with acceptable cost.

E. Test with Real Environment Data

We further applied the DSearching to the mobility data of 9 students on our campus. The 9 students are from 4 departments in our university. We selected 8 sub-areas, each of which is represented by a frequently visited building. Such mobility data is obtained based on the GPS on mobile phones, which is more accurate than the AP association records in the two real traces. The test environment and the mobility information are summarized in Figure 9(a) and 9(b).

Since DSearching shows stable performance with different locator rates in previous tests, we only varied the locator TTL from 20 minutes to 70 minutes and set the R_p. Since the distance between the test buildings are not far, we assume that a locator averagely takes 5 minutes to move from one sub-area to a neighboring sub-area.

TABLE IV: Results with real environment data

TTL (min)	20	30	40	50	60	70
Success Rate	0.62	0.75	0.83	0.88	0.89	0.93
Ave. Delay (min)	7.6	10.2	10.8	12.2	13.8	17.6
Ave. Path Length	1.4	1.9	2.0	2.3	2.6	3.1
Ave. Node Memory Usage	5	5	5	5	5	5

The test results are shown in Table IV. We find that when the locator TTL increases, success rate, average delay and average path length increase. This is the same as our observation in previous experiments with the two real traces. The reasons are the same that when the TTL increases, more locators can find the target nodes after a large delay and a long searching path.

We also see that when the TTL was set to 70 minutes, a successful locator takes only about 17 minutes (or 3 transits between sub-areas) to find the target node on average. Further, each node only needs 5 units of memory on average to support the node searching, which is very low and can easily be satisfied. In conclusion, DSearching is effective and efficient in searching mobile nodes in realistic DTNs.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants IIS-1354123, CNS-1254006, CNS-1249603, CNS-1049947, CNS-0917056 and CNS-1025652, Microsoft Research Faculty Fellowship 8300751.

V. CONCLUSION

In this paper, we propose DSearching, a distributed mobile node searching scheme in DTNs where nodes present certain mobility patterns. In DSearching, the whole network is split into sub-areas. A node's mobility information is summarized as both transient sub-area visiting records and long-term transition patterns among sub-areas. Each node distributes its visiting record for a sub-area to nodes that are likely to stay in the previous sub-area. The long term mobility information of a node in a sub-area is distributed to a limited number of long-staying nodes in the sub-area. The combination of sub-area visiting records and transition pattern enables the locator to search along the path that the target node traverses, leading to efficient node searching. Extensive trace driven experiments with both real traces and an on-campus DTN trace validate the high effectiveness and high efficiency of DSearching. In the future, we plan to investigate how to fully utilize node mobility information to further improve node searching efficiency.

REFERENCES

[1] S. Jain, K. R. Fall, and R. K. Patra, "Routing in a delay tolerant network," in *Proc. of SIGCOMM*, 2004.
[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] J. Zhao, Y. Zhu, and L. M. Ni, "Correlating mobility with social encounters: Distributed localization in sparse mobile networks." in *Proc. of MASS*, 2012.
[4] S. Lu, Y. Liu, Y. Liu, and M. Kumar, "Loop: A location based routing scheme for opportunistic networks." in *Proc. of MASS*, 2012.
[5] K. Chen and H. Shen, "Leveraging social networks for p2p content-based file sharing in disconnected manets." *IEEE TMC*, 2013.
[6] 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.
[7] J. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses." in *Proc. of SenSys*, 2005.
[8] 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.
[9] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem." in *Proc. of SIGCOMM*, 2007.
[10] K. Lee, Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong, "Max-Contribution: On optimal resource allocation in delay tolerant networks." in *Proc. of INFOCOM*, 2010.
[11] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proc. of MobiHoc*, 2008.
[12] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of MobiHoc*, 2007.
[13] J. Link, D. Schmitz, and K. Wehrle, "GeoDTN: Geographic routing in disruption tolerant networks," in *Proc. of GLOBECOM*, 2011.
[14] I. Leontiadis and C. Mascolo, "GeOpps: Geographical opportunistic routing for vehicular networks," in *Proc. of WOWMOM*, 2007.
[15] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain, "Predictability of wlan mobility and its effects on bandwidth provisioning." in *Proc. of INFOCOM*, 2006.
[16] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proc. of MOBICOM*, 2004.
[17] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Enhancing interactive web applications in hybrid networks," in *Proc. of MOBICOM*, 2008.
[18] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks." *Mobile Computing and Communications Review*, vol. 7, no. 3, 2003.