An Efficient Indoor Navigation Scheme Using RFID-based Delay Tolerant Network

Hao Ji, Lei Xie, Yafeng Yin, Sanglu Lu
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Email: jiqiusheng@dislab.nju.edu.cn, lxie@nju.edu.cn, yyf@dislab.nju.edu.cn, sanglu@nju.edu.cn

***Abstract*—As a supporting technology for most pervasive applications, indoor localization and navigation has attracted extensive attention in recent years. Conventional solutions mainly leverage techniques like WiFi, cellular network etc. to effectively locate the user for indoor localization and navigation. In this paper, we investigate into the problem of indoor navigation by using the RFID-based delay tolerant network. Being different from the previous work, we aim to efficiently locate and navigate to a specified mobile user who is continuously moving within the indoor environment. We respectively propose a framework to schedule the tasks and manage the resources in the network and a navigation algorithm to locate and navigate to the moving target. Experiment results show that our solution can efficiently reduce the average searching time for indoor navigation.**

I. Introduction

As the rapid proliferation of pervasive applications in indoor environment, a lot of location-based services and context-aware services are put forward, in which location is viewed as one of the most significant factors. For most applications, it is required to provide an accurate location for the specified objects. However, the current mature technology like global position system (GPS) can only be used in the outdoor environment for localization, several issues like the multi-path effect and severe path loss make the indoor localization a lot more complicated than the outdoor situation. Therefore, a lot of research works have focused on localization and navigation schemes for indoor environment [1-7]. Most of the solutions are rather complicated and fairly expensive.

Recent technological advances have enabled the development of low-cost, low-power devices [8-9]. RFID, as a novel technology for automatic identification, provides us with a new opportunity for indoor localization and navigation. For example, the low-cost RFID tags can be widely deployed inside the indoor environment and act as landmarks for localization. Since current smart phones can be equipped with near field communication (NFC) or bluetooth modules, which can effectively communicate with the active/passive tags, the mobile users can actively interrogate the surrounding tags with tiny devices like smart phones and leave messages or traces to the tags. In this way, the RFID-based infrastructure forms a delay tolerant network. As the scanning range of RFID system is usually no more than 5m, the system can effectively locate the users by limiting the positioning error to at most 5m.

In conventional indoor applications, the users are continuously moving within the indoor environment. Then, one important problem is how to locate and navigate to a specified mobile user. For example, when a baby or a dog is lost in a shopping mall, how to quickly locate and navigate to the mobile target? Obviously, the mobile target can only passively leave some traces in the environment through the equipped NFC or bluetooth modules. It cannot actively propagate its current position directly to the searchers. Besides, time-efficiency is very critical to the searchers, since the less time to use, the more opportunities to find the target.

Corresponding Author: Dr. Lei Xie, lxie@nju.edu.cn

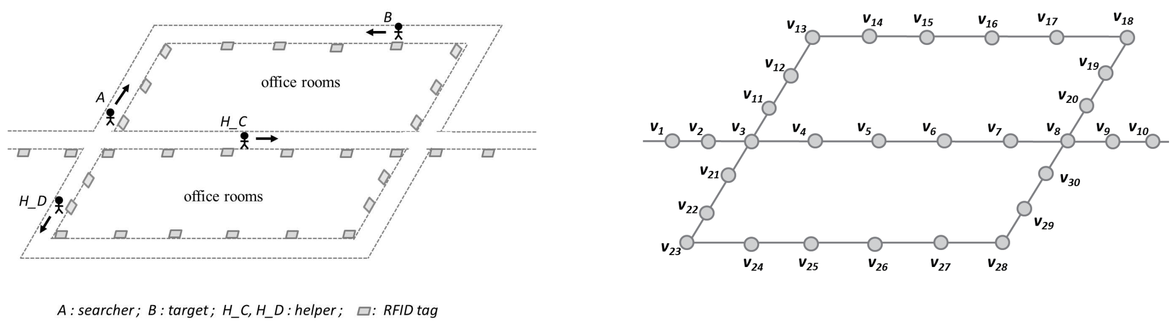
Therefore, it is essential to devise a time-efficient navigation scheme by using the RFID-based delay tolerant network. In this paper, we first propose a framework to schedule the tasks and manage the resources in this network. Furthermore, we propose a navigation algorithm to locate and navigate to the moving target. The main contributions of this paper are summarized as follows:

- We propose a framework leveraging RFID-based delay tolerant network for localization and navigation. By sufficiently leveraging the “store-and-forward” properties of the delay tolerant network, our solution provides an effective mechanism for navigation using “crowdsourcing” capabilities. By effectively scheduling the tasks and managing the limited resources in the tags, the system can provide navigation services for a large number of users.
- We propose a time-efficient scheme to locate and navigate to a mobile target who is continuously moving. According to the latest obtained spots of appearance, our solution navigates the searcher to the most possible region of the target, which achieves a good performance in terms of the time-efficiency.

II. Related Work

Many research works use RFID technology for indoor localization [10-13]. LANDMARC [10] is a tag localization prototype in indoor environment. By utilizing extra fixed location reference tags to help location calibration, it can increase location accuracy without deploying large numbers of RFID readers. Zhu et al. propose an fault-tolerant RFID reader localization approach to solve the problem of frequent occurred RFID faults [13]. Moreover, they also propose the index to measure the quality of a localization result.

Escort [14] is an office environment localization and navigation system which uses client/server architecture. The client running on the user-carried mobile phones periodically measures the value of accelerometer and compass of the user’s



(a) A simplified single floor plan of the indoor environment (b) Graph of the RFID-based dealy tolerant network network

Fig. 1. An overview of the indoor navigation scheme using RFID-based delay tolerant network

walking trail. The user's trail is periodically reported to escort server. To correct user's position diverging from his actual location, audio beacons are placed in the building as a reference frame. Encounter between two mobile phones, and encounter between the mobile phone and the beacon will both be reported to the escort server. Escort server utilizes user's walking trail and encounters to compute the current position of each user and routing directions. Being different from the previous works, we focus on the problem of navigating to a moving target using RFID-based delay tolerant network.

III. System Overview

Figure 1a shows the envisioned application scenario for our RFID-based delay tolerant network navigation scheme. The mobile users can actively interrogate the surrounding tags with tiny devices like smart phones and leave messages to the tags. Those tags act as landmarks for localization, so the placement of them are very important. In order to save labor costs of tag deployment without reducing the quality of navigation, we can deploy RFID tags in key locations, such as door of each office room, meeting room and washing room. In addition, some public locations also need to be deployed with tags, such as corners of corridor, elevators and entrances of building.

Each interrogator carried by the user in our scheme has two kinds of operations: *post* and *query*. Interrogator uses post operation to write messages to tags, and uses query operation to read messages from tags. For example, when a user reaches an office room, the user can post an event message to the surrounding tags through the interrogator. This event message states that the user has been here before. At the same time, the user can also query other users' event messages from the surrounding tags, so as to detect other users' traces.

There exist three types of roles in our scheme: *searcher*, *target* and *helper*. Each user is a helper for others. Specially, a user can also be a searcher if he or she is looking for another user. Coincidentally, if he or she is also a target of others, the user will have the third role as a target.

Suppose a searcher ***S*** is looking for a target ***T***. At first, the searcher doesn't know the position of target. Searcher ***S*** queries the surrounding tags to detect the target ***T***'s event messages. Besides, searcher ***S*** also posts request message to the surrounding tags which states that searcher ***S*** is looking for the target ***T***. As time goes on, when a helper ***H*** detects the request message, the helper will know ***S*** is looking for ***T***. The helper ***H*** has two ways to help the searcher. One way is to post this request message to surrounding tags which he will pass by. Thus more users may detect the request and give help to the searcher; The other way is to post event messages of the target ***T*** detected by ***H*** to more tags. The searcher ***S*** continuously detects event messages about the target ***T***, and adjusts his movement direction until he meets the target at some place.

Our goal is to design a time efficient approach to navigate a searcher to a moving target in indoor environment. There are two challenges in the problem. One is that the target is in the movement. In the RFID-based delay tolerant network, there is no central server which can record each user's position in real time. The target's position information is stored in tags which are distributed in the environment. This is the main difference between our navigation scheme and the escort [14]. The other challenge is that the storage capacity of RFID tags is limited. The operation of posting message needs to be managed reasonably.

IV. Problem Analysis

We analyse the problem of locating and navigating to a specified mobile user within the indoor environment on a *undirected graph* $G = (V, E)$. The undirected graph is an abstraction of the RFID tag location reference frame, for example, Figure 1b. V is the set of vertices. Each vertex $v_i \in V$ represents a tag in the frame. Each vertex v_i carries a positive value s_i which is the memory size of the tag. E is the set of edges. If two vertices $v_i \in V$ and $v_j \in V$ are adjacent in physical space, there is an edge (v_i, v_j) between vertex v_i and vertex v_j. Each edge (v_i, v_j) carries a positive value $w[v_i][v_j]$ which is the distance between the two vertices connected by the edge.

On the graph G, we want to find the optimal vertex sequence

$$\langle v_1', v_2', ..., v_n' \rangle \,, \quad (v_i', v_{i+1}') \in E, 1 \le i \le n-1 \qquad (1)$$

where v_1' is the current location of the searcher ***S***, and v_n' is the finally location where the searcher found the target ***T***, so as to minimize the total distance of the vertex sequence. Let

L be the total distance which is defined below:

$$L = w[v'_1][v'_2] + w[v'_2][v'_3] + ... + w[v'_{n-1}][v'_n] \quad (2)$$

Formally, our goal is to

$$min \quad L \Leftrightarrow min \quad \sum_{i=1}^{n-1} w[v'_i][v'_{i+1}] \quad (3)$$

subject to the memory size $\leq s_i$ for each vertex v'_i. In this paper, we use the average searching time to measure the performance of our solution.

V. Distributed Solution

A. The Framework for Mobile Navigation

We describe the framework of mobile navigation on the undirected graph G as shown in **Algorithm 1**. Each user moves from one vertex to another vertex on the graph. Our task is to navigate a searcher ***S*** to a moving target ***T***.

There exist two kinds of messages in the framework: *event* message and *request* message. Each of them is a three tuple. We use $E\langle H_i, v_j, t\rangle$ to represent an event message, and $R\langle S, T, t\rangle$ to represent a request message, respectively. The meaning of these two messages is defined as follows:

1. $E\langle H_i, v_j, t\rangle$: H_i represents a user; v_j represents a vertex on the graph G; t represents the time when the event took place. It states that user H_i passed by vertex v_j at time t.

2. $R\langle S, T, t\rangle$: ***S*** represents a searcher; ***T*** represents a target; t represents the time when the request was generated. It states that ***S*** wants to look for ***T*** at time t.

Combining two operations: post, query with these two kinds of messages, users get four ways to communicate with the surrounding RFID tags. They are *post event*, *post request*, *query event* and *query request*. Users use post operations to write event or request messages to the surrounding tags, and query operations to read messages from the surrounding tags.

Each user has two modes: **normal mode** and **searcher mode**. At first, all the users and the potential targets are working in the normal mode. When the user is looking for a target, he will jump to the searcher mode. The searcher mode is an escalation state of the normal mode.

In the normal mode, when a user is passing by a vertex, he will do three things. First, he queries all event messages stored on the vertex and adds them to his **trace table**, so as to detect other users' traces. Second, he posts an event message to the vertex, so as to leave traces of himself for potential searchers. Third, he queries all request messages stored on the vertex and adds them to his **help queue**, so as to detect searchers' request messages.

For each request message $R\langle S, T, t\rangle$ in user H_i's help queue, user H_i will make a decision whether to help the searcher ***S*** to look for the target ***T***. If the current time t_{now} minus the timestamp of the request message t exceeding a given threshold θ_t ($\theta_t \geq 0$), H_i discards the request and does nothing. Otherwise H_i decides to help the searcher. We define τ for each pair of searcher ***S*** and target ***T***. At first, τ is initialized to an given positive threshold θ_R. As time goes on, H_i will carry and forward this request message $R\langle S, T, t\rangle$ and event messages of the target to vertices he passes by. We call this process is the forwarding process. When τ is reduced to zero, user H_i stops this forwarding process, and discards the request message.

In the searcher mode, when a searcher is passing by a vertex, he will post a request messages $R\langle S, T, t\rangle$ to the vertex. Once other users pass by the same vertex, they will detect this request message, and know that the searcher ***S*** is looking for the target ***T***. Using this way, searcher ***S*** gets helps from other users. The searcher uses **Algorithm 2** to select the next neighbor vertex to move. Because ***S*** may be a helper for others, he also needs to do step 2, 3 in the normal mode.

Algorithm 1 The Framework for Mobile Navigation

```
Normal Mode
1. If user $H_i$ is looking for target $T$ then
       Jump to Searcher Mode.
2. When user $H_i$ is passing by vertex $v_j$ at time $t$,
     (a). user $H_i$ queries all event messages stored on $v_j$,
          and adds these event messages $\{E\langle *, v_j, *\rangle\}$ to
          $H_i$'s trace table;
     (b). user $H_i$ posts an event message $E\langle H_i, v_j, t\rangle$ to $v_j$;
     (c). user $H_i$ queries all request messages stored on $v_j$,
          and adds these request messages $\{R\langle S, T, *\rangle\}$ to
          $H_i$'s help queue.
3. For each $R\langle S, T, t\rangle$ in $H_i$'s help queue
     dequeue $R\langle S, T, t\rangle$;
     if $t_{now} - t < \theta_t$ then
         $\tau = \theta_R$;
         repeat :
             when $H_i$ is passing by vertex $v_k$,
                 $\tau = \tau - 1$;
           (a). $H_i$ posts a request message $R\langle S, T, t\rangle$ to $v_k$;
           (b). $H_i$ selects event messages $\{E\langle T, *, *\rangle\}$ of
                the target $T$ from $H_i's$ trace table; then
                $H_i$ posts these event messages to $v_k$.
         until $\tau = 0$ .
Searcher Mode
1. Step 2, 3 in the Normal Mode.
2. When passing by vertex $v_j$ at time $t$, searcher $S$ posts
    a request message $R\langle S, T, t\rangle$ to $v_j$.
3. Call Algorithm 2 to select the next neighbor vertex.
4. When searcher $S$ finds target $T$,
       Jump to Normal Mode.
```

B. Trace Table and Help Queue

Under the above navigation framework, each user maintains a trace table to record other users' event messages. Each entry in the trace table is a key-value pair. The key of the entry is *user_id* which is the identification of a user. The value of the entry is an event message list which records the top θ_k most recent traces of the *user_id*, where θ_k is a positive threshold predefined.

In addition, each user also maintains a help queue to record request messages of searchers under the framework. Each entry in the help queue is a request message which gives the identifications of searcher and target, also the timestamp of this message. This request message was firstly posted by the searcher, then other users carry and forward this message without changing the timestamp of it. We use a positive time threshold θ_t to decide whether to discard outdated requests.

C. Management of RFID tags

Because the memory size of RFID tag is limited, we should make full use of these storage resources. Under the navigation framework, post operation will write messages to tags. Therefore, a reasonable writing and replacing strategy should be used. Now, considering the following situations.

Situation 1. Normal Mode 2(b) Each tag maintains at most λ event messages of each user H_i. If the number of event messages is less than λ, then $E\langle H_i, v_j, t\rangle$ will be written to v_j. Otherwise, the oldest event message of H_i stored on v_j will be replaced with $E\langle H_i, v_j, t\rangle$.

Situation 2. Normal Mode 3(a) Each tag maintains at most one request message of searcher ***S*** and target ***T***. If there already exists a request message $R\langle S, T, t'\rangle$ on v_k and $t' < t$, then the timestamp t' will be updated to t. No extra memory space will be used. Besides, we take the same strategy in Searcher Mode step 2.

Situation 3. Normal Mode 3(b) Because there are many *helpers* in the system, if all of them post messages to a vertex as soon as they pass by it, the memory space of the tag will be reduced rapidly. Therefore, we use a posting probability $p = s'_i/s_i, (0 \leq p \leq 1)$ to adjust users' post operations, where s_i is the total memory size of v_k, and s'_i is the memory size which has not been used of vertex v_k. When a helper passes by a vertex, he will post messages to the vertex with a posting probability p.

D. Navigation Algorithm

Our goal is to find the optimal vertex sequence, so as to navigate searcher ***S*** to a moving target ***T*** with minimum time cost. At the beginning of the above navigation framework, if there is no priori knowledge of the target's movement, searcher ***S*** can only randomly select a place to move without any guidance. After a period of time, the searcher collects some event messages of the target which states that the target appears at some places. The more traces searcher collects, the better searcher will know about the target's movement. If history data of the target's movement can be obtained, the searcher can even predict the current location of the target according to those traces he gets now. At least, searcher ***S*** has some instructive guidance at this time.

When searcher ***S*** has collected some event messages of the target ***T***, the searcher could use these traces as the guidance to select the next neighbor vertex to move. A simple way to do this is selecting the neighbor vertex which is nearest to the latest appearance location of the target. For example, suppose the weight of all edges are equal in Figure 1(b). The current location of the searcher is v_3, and the latest appearance location of the target is v_{17}. Then vertex v_{11} will be selected, because it is the nearest neighbor of v_3 to the vertex v_{17}.

However, three reasons make us do not use the above simple method to select the next vertex to move. The first reason is that the trace with latest timestamp may be far away. There should be a tradeoff between the timestamp of the trace and distance. The second reason is that the target is a moving object whose motion is a continuous process. The third reason is that the movement behavior of the target has locality in the real world. We use **Algorithm 2** to select the next neighbor to move and guide the searcher to be close to a target *region* rather than a single point.

Algorithm 2 Navigation Algorithm

```
Input: $t_{now}$; $v_s$; $d$;    $V = \langle \nu_1, \nu_2, ..., \nu_m \rangle$;
       $U = \langle u_1, u_2, ..., u_n \rangle$; $\Gamma = \langle t_1, t_2, ..., t_n \rangle$;
Output: next neighbor $\nu_{next}$ to move
1: Calculate all vertices pairs' shortest path length on graph $G$ using algorithm such as Floyd-Warshall or Dijistra; and store these values in matrix $d$.
2: for each $u_i \in U$ do
3:     $\alpha_i = \frac{1}{t_{now} - t_i}$;
4: end for
5: $\alpha = \sum_{i=1}^{n} \alpha_i$;
6: for each neighbor $\nu_j \in V$ of $v_s$ do
7:     $f(\nu_j) = d[v_s][\nu_j] + \sum_{i=1}^{n} \frac{\alpha_i}{\alpha} \times d[\nu_j][u_i]$;
8: end for
9: Select the smallest value $f(\nu_{min})$ from $f(\nu_1)$,...,$f(\nu_m)$; $\nu_{next} = \nu_{min}$;
```

In **Algorithm 2**, t_{now} is the current time; v_s is the current location of the searcher ***S***; d is a shortest path matrix which can be precalculated; each element $d[v][v'] \in d$ is the shortest path length between vertex v and v'; all vertices in collection $V = \langle \nu_1, \nu_2, ..., \nu_m \rangle$ are the neighbors of vertex v_s. Besides, we use the trace table of searcher ***S*** to predict the current location of the target. Let n ($n \leq \theta_k$) is the event messages number of the target ***T***. Arrange these event messages in descending order according to the timestamp. Represent the arrangement with the list $E\langle T, u_1, t_1\rangle$, $E\langle T, u_2, t_2\rangle$, ... , $E\langle T, u_n, t_n\rangle$. Correspondingly, $\Gamma = \langle t_1, t_2, ..., t_n \rangle$ is the list of timestamps, and $U = \langle u_1, u_2, ..., u_n \rangle$ is the list of vertices. Vertices in list U compose the target region.

For each neighbor ν_j of v_s, we define a cost function $f(\nu_j)$ to predict whether the searcher is close to the target or far from the target. We assign a normalized weight α_i/α for each vertex u_i in the target region according to the timestamp. The weight of vertex with latest traces will have higher value than that of other vertices. Cost function $f(\nu_j)$ is defined below:

$$f(\nu_j) = w[v_s][\nu_j] + \sum_{i=1}^{n} \frac{\alpha_i}{\alpha} \times d[\nu_j][u_i] \tag{4}$$

The neighbor vertex which has the lowest cost will be selected as the next intermediate vertex. For example, in Figure 2, we assume that all the edges between two vertices on the graph equals to 1; $n = 3$; $V = \langle v_2, v_4, v_{11}, v_{21} \rangle$, $U = \langle v_{17}, v_{18}, v_{19} \rangle$,

$\Gamma = \langle 4, 3, 1 \rangle$, the current time $t_{now} = 5$. Then vertex v_{11} will be selected as the next intermediate vertex. Because the cost of v_{11} is 7.57 which is less than the cost of v_2, v_4 or v_{21}.

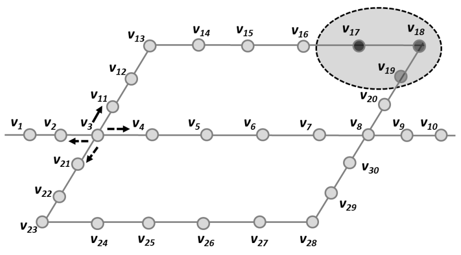


Fig. 2. Select the next position (vertex) and move to a target region

VI. Performance Evaluation

We implement a simulator in Java to simulate the navigation process, thus to evaluate the performance of our solution. We conduct the simulation based on a 30×30 grid-graph. Each vertex on the graph represents a tag in the RFID-based delay tolerant network. In addition to the boundary vertices, each vertex has four neighbor vertices. To simulate the movement behaviors of users on the graph, we use two classical human mobility models [15, 16].

We compare the performance of our solution with two other solutions: blind navigation solution and centralized navigation solution by the average searching time. In blind navigation solution, searcher randomly selects a vertex on the graph and moves to the vertex with shortest path. If the searcher doesn't find the target on the path, then the searcher will repeat the above process until he finds the target. In centralized solution, we suppose that the searcher can obtain the target's current position in real time. The searcher always moves to the neighbor vertex which is nearest to the target. In this paper, we regard the centralized solution as the approximate optimal solution and the blind navigation solution as the worst solution.

A. Experiments Settings

At the beginning of the simulation, we randomly select a vertex on the 30×30 grid-graph for each user as the user's initial position. All the users are limited to move on the edges of the grid-graph. The length of each edge is 4 m. In order to avoid situation that searcher never catches up with the target, we assign the movement speed of the searcher is 2 m/s and other users' are 1 m/s. If searcher and the target happen to be on the same edge, we think the searcher has found the target.

To simulate the movement behaviors of users, we use a simplified Random Walk Mobility Model (RW) and a Random Way-Point Mobility Model(RWP). Under the random walk mobility model, whenever a user reaches a vertex, the user will randomly select a neighbor vertex to move. Under the random way-point mobility model, each user randomly selects a vertex within a local region on the graph and moves to the vertex with shortest path. When arriving at the vertex, the user will stay on the vertex for a period of time, then repeat the above process. We use a circular region to describe such a local region. The center of the circle is the user's initial position and the radius of the circle obeys a normal distribution $N(\mu, 6^2)$, where μ is the expectation of radius. The stay time of each user obeys a truncated power-law distribution $p(t) = C \times t^{-2.5}(s)$, $(5 \leq t \leq 60)$. Parameters used in the experiments are showed in Table 1. For all figures presented, we run the simulation 1,000 times to get average values.

Table 1. Parameters used in the simulation

Parameter	Description	Value
m	number of users	[50, ..., 500]
θ_R (s)	time to live of request message	[10, ..., 200]
θ_k	number of event message	[1, ..., 10]
μ (m)	expectation of region radius	[6, ..., 60]

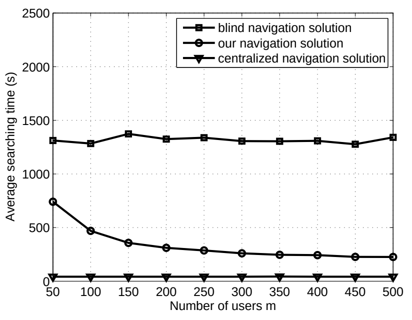
B. Experiments Result

1) Average Searching Time: In Figure 3a, 3b, we plot the average searching time of three solutions under two mobility models with different numbers of users. We guarantee that the inputs of users' movement behaviors for each solution are the same. As we can see from these 2 subfigures, with the number of users increasing, the average searching time of our solution is reduced gradually. When increasing the number of users from 50 to 500, our solution's average searching time is reduced from 742/487 seconds to 226/234 seconds. While the average searching time of blind navigation or centralized navigation is essentially unchanged. This is consistent with our intuition that more users means more helpers for the searcher.

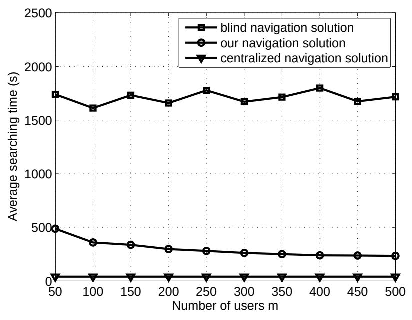
In Figure 3c, 3d, we plot the average searching time of our solution under two mobility models with different values of θ_R and θ_k. When the value of θ_R is small, the average searching time is large. Because only a few number of users could receive request messages and the searcher gets little help from others. When increasing the value of θ_R from 10 seconds to 200 seconds, the average searching time is reduced more than 30%. Different from θ_R, the impact of θ_k is little. When the value of θ_k is 1, 5 under RW or 1, 3, 9 under RWP, the average searching time is slightly smaller than others.

In Figure 3e, we plot the average searching time of our solution under RWP with different localities of users' movement behaviors. When the value of μ is less than 12, users move in a small region. The request messages of searcher spread slowly. Thus it takes a long time for the searcher to detect the target's traces, and the average searching time is large. When the locality of users' movements is week, say $\mu \geq 30$, the movement region of the target is large. So, it is also difficult for the searcher to find the target. We find that when μ is 20 the average searching time is the smallest.

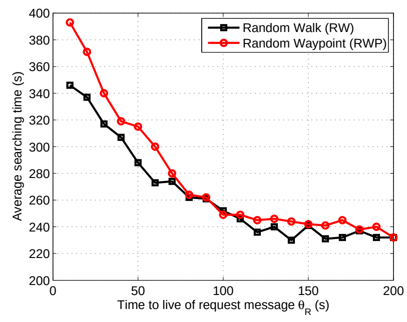
2) Number of Messages: In Figure 3f, we plot the average number of messages stored in the delay tolerant network as time grows. The number of request messages under RWP is sightly smaller than RW. When the time is about 200 (s), the number of request messages reaches its maximum. The number decreases as the request message's time to live is reduced to zero. The number of event messages grows fast between 150 (s) to 350 (s). After 400 (s), the number of event messages is essentially unchanged. Because no more request messages are generated after that.



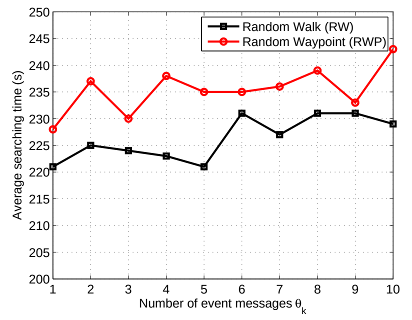
(a) RW, $\theta_R = 200(s)$, $\theta_k = 10$



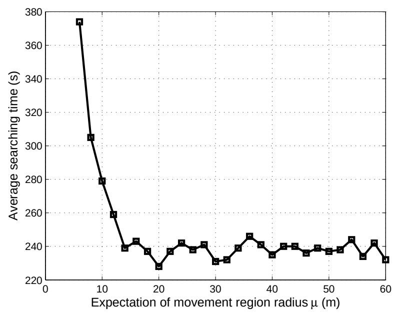
(b) RWP, $\theta_R = 200(s)$, $\theta_k = 10$, $\mu = 18(m)$



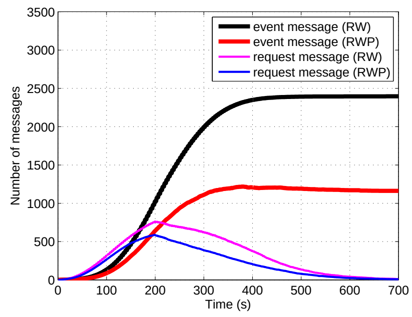
(c) $m = 450$, $\theta_k = 10$, $\mu = 18(m)$



(d) $m = 450$, $\theta_R = 200(s)$, $\mu = 18(m)$



(e) RWP, $m = 450$, $\theta_R = 200(s)$, $\theta_k = 10$



(f) $m = 450$, $\theta_R = 200(s)$, $\theta_k = 10$, $\mu = 18(m)$

Fig. 3. Comparison of the average searching time under Random Walk Mobility Model (RW) and Random Way-Point Mobility Model (RWP) with different parameters (number of users m; time to live of request message θ_R; number of event messages θ_k; expectation of movement region radius μ).

VII. Conclusion

This paper proposes a framework using RFID-based delay tolerant network for indoor navigation. By sufficiently leveraging the store-forward properties of delay tolerant network, our solution provides an effective mechanism for indoor navigation. Simulation results show that our solution can efficiently reduce the average searching time of navigation. We discuss the influences of different mobility parameters in the end.

VIII. Acknowledgement

This work is partially supported by the National Basic Research Program of China (973) under Grant No. 2009CB320705; the National Natural Science Foundation of China under Grant No. 61100196, 61073028, 61021062, 91218302; the JiangSu Natural Science Foundation under Grant No. BK2011559.

References

[1] N. B. Priyantha, A. K. Miu, H. Balakrishnan, and S. Teller, "The cricket compass for context-aware mobile applications," in *Proc. MobiCom*, Rome, Italy, July 2001, pp. 1-14.

[2] M. Minami, Y. Fukuju, K. Hirasawa, S. Yokoyama, M. Mizumachi, H. Morikawa, and T. Aoyama, "DOLPHIN: a practical approach for implementing a fully distributed indoor ultrasonic positioning system," in *UbiComp 2004: Ubiquitous Computing*, Tokyo, Japan, Sept. 2004, pp. 347-365.

[3] G. Fischer, B. Dietrich, and F. Winkler, "Bluetooth indoor localization system," in *Proc. WPNC*, Hanover, Germany, March 2004, pp. 147-156.

[4] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: mobile phone localization via ambience fingerprinting," in *Proc. MobiCom*, Beijing, China, Sept. 2009, pp. 261-272.

[5] J. Biswas, and M. Veloso, "Wifi localization and navigation for autonomous indoor mobile robots," In *Proc. ICRA*, Anchorage, Alaska, USA, May 2010, pp. 4379-4384.

[6] X. Jiang, C.-J. M. Liang, F. Zhao, K. Chen, J. Hsu, B. Zhang, and J. Liu, "Demo: Creating interactive virtual zones in physical space with magnetic-induction," in *Proc. SenSys*, Seattle, WA, USA, Nov. 2011, pp. 431-432.

[7] X. Jiang, C.-J. M. Liang, K. Chen, B. Zhang, J. Hsu, J. Liu, B. Cao, and F. Zhao, "Design and evaluation of a wireless magnetic-based proximity detection platform for indoor applications," in *Proc. IPSN*, Beijing, China, April 2012, pp. 221-232.

[8] L. Xie, B. Sheng, C. Tan, H. Han, Q. Li and D. X. Chen, "Efficient tag identification in mobile RFID systems," in *Proc. INFOCOM*, San Diego, CA, USA, March 2010, pp. 1-9.

[9] L. Xie, Q. Li, X. Chen, S. L. Lu and D. X. Chen, "Continuous Scanning with Mobile Reader in RFID Systems: an Experimental Study," in *MobiHoc*, Bangalore, India, Aug. 2013.

[10] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: indoor location sensing using active RFID," *Wireless Networks*, vol. 10, no. 6, pp. 701-710, Nov. 2004.

[11] H. J. Lee and M. C. Lee, "Localization of mobile robot based on radio frequency identification devices," in *International Joint Conference SICE-ICASE*, Busan, Korea, Oct. 2006, pp. 5934-5939.

[12] S. S. Saad and Z. S. Nakad, "A standalone RFID indoor positioning system using passive tags," *IEEE Trans. Industrial Electronics*, vol. 58, no. 5, pp. 1961-1970, 2011.

[13] W. Zhu, J. Cao, Y. Xu, L. Yang, and J. Kong, "Fault-tolerant RFID reader localization based on passive RFID tags," in *Proc. INFOCOM*, Orlando, Florida USA, March 2012, pp. 2183-2191.

[14] I. Constandache, X. Bao, M. Azizyan and R. R. Choudhury, "Did you see Bob?: human localization using mobile phones," in *Proc. MobiCom*, Chicago, Illinois, USA, Sept. 2010, pp. 149-160.

[15] T. Camp, J. Boleng and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483-502, 2002.

[16] K. Lee, S. Hong, S. J. Kim, I. Rhee and S. Chong, "Slaw: A new mobility model for human walks," in *Proc. INFOCOM*, Rio de Janeiro, Brazil, April 2009, pp. 855-863.