Journal of Network and Computer Applications 52 (2015) 79–89

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

CrowdSensing: A crowd-sourcing based indoor navigation using RFID-based delay tolerant network

Hao Ji, Lei Xie*, Chuyu Wang, Yafeng Yin, Sanglu Lu

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

ARTICLE INFO

Article history:
Received 20 April 2014
Received in revised form
17 November 2014
Accepted 16 February 2015
Available online 12 March 2015

Keywords:
Indoor localization
Indoor navigation
RFID
Delay-tolerant network
Crowd-sourcing

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 and cellular network to effectively locate the user for indoor localization and navigation. In this paper, we investigate the problem of indoor navigation by using the RFID-based delay tolerant network. 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. As the low-cost RFID tags are widely deployed inside the indoor environment and acting as landmarks, the mobile users can actively interrogate the surrounding tags with devices like smart phones and leave messages or traces to the tags. These messages or traces can be carried and forwarded to more tags by other mobile users. In this way, the RFID-based infrastructure forms a delay tolerant network. By using the crowd-sourcing technology in RFID-based delay tolerant network, we respectively propose a framework, namely CrowdSensing, to schedule the tasks and manage the resources in the network. We further propose a navigation algorithm to locate and navigate to the moving target. We verify the performance of proposed framework and navigation algorithm on mobility model built on real-world human traceset. Experiment results show that our solution can efficiently reduce the average searching time for indoor navigation.

1. 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 (Priyantha et al., 2001; Minami et al., 2004; Fischer et al., 2004; Azizyan et al., 2009; Biswas and Veloso, 2010; Jiang et al., 2011, 2012). Most of the solutions are rather complicated and fairly expensive.

Recent technological advances have enabled the development of low-cost and low-powered devices (Xie et al., 2010, 2013). 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 5 m, the system can effectively locate the users by limiting the positioning error to at most 5 m.

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.

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.

* Corresponding author.
E-mail address: lxie@nju.edu.cn (L. Xie).

http://dx.doi.org/10.1016/j.jnca.2015.02.010
1084-8045/

A preliminary version of this work appeared in Ji et al. (2013), the main differences of this paper are three folds. First, we conduct an in-depth study on the storage management of RFID tags, we propose four strategies of writing and replacing tag storage memory and their corresponding usage scenarios (Section 4.3). Second, we conduct detailed analysis on the relationship between tag density and network parameters, such as the number of users, the movement speed of each user and the range of each users' activity area. It provides a fundamental guidance for the deployment of RFID-based delay tolerant network (Section 4.5). Third, we conduct the experiments with real-world human traces under shopping mall mobility model, illustrate some novel observations from the experiment results, and further verify the rationality of our navigation solution in practical situations (Section 5.1.1).

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.
- We conduct two kinds of experiments: large-scale experiment through simulation and fairly small-scale experiment using realistic human traces. In the large-scale simulation experiments, we investigate the relationships among number of users, tag density and performance of navigation. In the small-scale experiments, we strive to accurately reconstruct the movement scenes of a shopping mall founded on real-world human traces to verify our navigation framework and scheme.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 provides an overview of the system. Section 4 introduces the distributed solution for indoor navigation. Section 5 shows the performance evaluation. Section 6 concludes this paper.

2. Related work

Many research works use RFID technology for indoor localization. LANDMARC (Ni et al., 2004) 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. Lee and Lee (2006) construct an absolute positioning system based on RFID and investigate how localization technique can be enhanced by RFID through experiment to measure the location of a mobile robot. Saad and Nakad (2011) present a standalone indoor positioning system using RFID technology. The concept is based on an object carrying an RFID reader module, which reads low-cost passive tags installed next to the object path. The positioning system uses Kalman filter to iteratively estimate the location of the reader. Zhu et al. (2012) propose a fault-tolerant RFID reader localization approach to solve the problem of frequently occurred RFID faults. Moreover, they also propose the index to measure the quality of a localization result. Since the localization accuracy of LANDMARC approach largely depends on the density of reference tags and the high cost of RFID readers, Zou et al. (2013) propose two localization algorithms, namely weighted path loss (WPL) and extreme learning machine (ELM), to overcome these drawbacks. Ng et al. (2011) apply Radial Basis Function Neural Network (RBFNN) to estimate location of objects based on RFID signal strengths. Tong and Wang (2014) propose a novel RFID indoor positioning system based on Doppler Effect of moving RFID antenna. The Doppler frequency of RFID signal is recorded to compute the relative velocity between the antenna and target tags. By comparing the antenna movement with the relative velocity data, the position of the target is estimated using triangulation. Escort (Constandache et al., 2010) 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 walking trail, and reports it to escort server. Encounter between two mobile phones, and encounter between a mobile phone and an audio beacon placed in the building will both be reported to escort server. Escort server utilizes users' walking trail and encounters to compute the current position of each user and routing directions.

In the theoretical research of delay tolerant network, a lot of work have been done to reveal the relationship between latency and network parameters, such as node density, connectivity range, node and movement speed. Dousse et al. (2004) prove that under certain assumptions the message sent by a sensing node reaches the sink node with a fixed asymptotic speed that does not depend on the random location of the nodes, but only on the network parameters. Kong and Yeh (2008) use percolation theory to analyze the latency for information dissemination in large-scale mobile wireless networks. They show that under a constrained mobility model, the scaling behavior of the latency falls into two regimes. When the network is not percolated, the latency scales linearly with the initial Euclidean distance between the sender and the receiver; when the network is percolated, the latency scales sub-linearly with the distance. Zhao et al. (2011) present fundamental relationship between node density and transmission delay in large-scale wireless ad hoc networks with unreliable links from percolation perspective. Yang et al. focus on the problem of rostering in intermittently connected passive RFID networks. They propose a rostering algorithm that employs a dynamic space-efficient coding scheme to construct hypothetic packet candidates (Yang et al., 2013). Bogo and Peserico (2013) study delay and throughput achievable in delay tolerant networks with ballistic mobility. They show that, under some very mild and natural hypotheses, as the number of nodes grows, (a) per-node throughput does not become vanishingly small and (b) communication delay does not become infinitely large. Kim et al. (2014) propose an efficient DTN routing scheme by using a node's social relation where each node chooses a proper relay node based on its contact history.

Greatly different from previous works, in this paper we focus on the problem of navigating to a moving target in indoor environment. The localization and navigation service are provided based on a RFID-based delay tolerant network. There is no central server in the system which can record all users' positions in real-time. The size of each RFID tag's memory space is small. By sufficiently leveraging the “store-and-forward” properties of the delay tolerant network and the ‘crowdsourcing” capabilities brought by encounters between users and tags, we propose a time-efficient scheme to locate and navigate to a mobile target.

3. System overview

Most of the previous indoor positioning and navigation systems are centralized architecture which have the advantages of timeliness

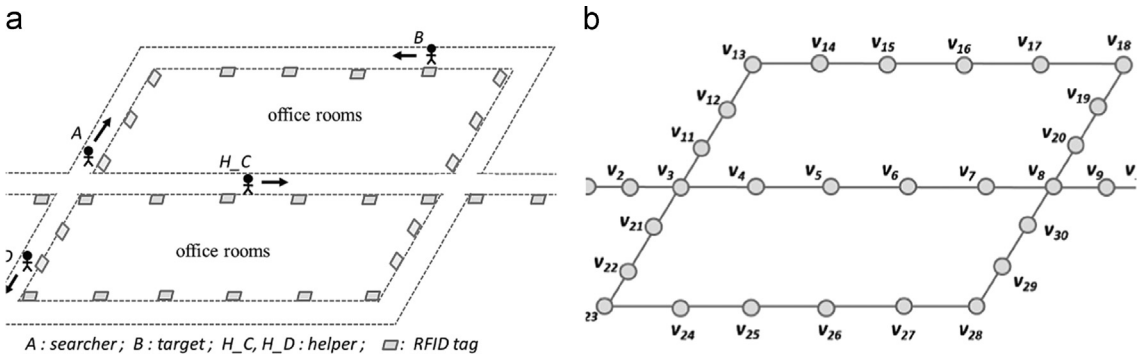

Fig. 1. An overview of the indoor navigation scheme using RFID-based delay tolerant network. (a) A simplified single floor plan of the indoor environment. (b) Graph of the RFID-based delay tolerant network.

and integrity of position. Client node detects object's location information and periodically reports it to a center server node. They can monitor the whole objects' locations and movement behaviors in real-time. In fact, the cost of constructing such a centralized navigation architecture system is always high. Communication between client node and server node is highly energy-consuming. In order to overcome the above drawbacks, we propose a novel distributed indoor navigation architecture which uses RFID-based delay tolerant network. Objects' location information are stored on surrounding tags and no centralized server exists in the system.

Figure 1(a) 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 $\boldsymbol{S}$ is looking for a target $\boldsymbol{T}$. At first, the searcher does not know the position of target. Searcher $\boldsymbol{S}$ queries the surrounding tags to detect the target $\boldsymbol{T}$'s event messages. Besides, searcher $\boldsymbol{S}$ also posts request message to the surrounding tags which states that searcher $\boldsymbol{S}$ is looking for the target $\boldsymbol{T}$. As time goes on, when a helper $\boldsymbol{H}$ detects the request message, the helper will know that $\boldsymbol{S}$ is looking for $\boldsymbol{T}$. The helper $\boldsymbol{H}$ 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 $\boldsymbol{T}$ detected by $\boldsymbol{H}$ to more tags. The searcher $\boldsymbol{S}$ continuously detects event messages about the target $\boldsymbol{T}$, 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 (Constandache et al., 2010). The other challenge is that the storage capacity of RFID tags is limited. The operation of posting message needs to be managed reasonably.

4. Distributed solution

4.1. The framework for mobile navigation

We define a *undirected graph* $G=(V,E)$ which is an abstraction of the RFID tag location reference frame, for example, Fig. 1(b). 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 physical distance between the two vertices connected by the edge.

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

Algorithm 1. The framework for mobile navigation.

Normal Mode
1. **If** user H_i is looking for target $\boldsymbol{T}$ **then**
   Jump to Searcher Mode.
2. When user H_i is passing by vertex v_j at time t,
   (a) H_i queries all event messages stored on v_j and adds these event messages $\{E\langle *, v_j, * \rangle\}$ to H_i's **event table**;
   (b) H_i posts an event message $E\langle H_i, v_j, t \rangle$ to v_j;
   (c) H_i queries all request messages stored on v_j and adds these request messages $\{R\langle S, T, * \rangle\}$ to H_i's **request queue**.
3. **For each** $R\langle S, T, t \rangle$ in H_i's **request 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 $\boldsymbol{T}$ from H_i's **event table**; then
   H_i posts these event messages to v_k.
   until $\tau = 0$.

Searcher Mode
1. **Execute Steps 2 and 3 in the Normal Mode**.
2. When passing by vertex v_j at time t, searcher $\boldsymbol{S}$ 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 $\boldsymbol{S}$ finds target $\boldsymbol{T}$,
   Jump to 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 **event 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 **request queue**, so as to detect searchers' request messages. For each request message $R\langle S, T, t \rangle$ in user H_i's **request queue**, user H_i will make a decision whether to help the searcher $\boldsymbol{S}$ to look for the target $\boldsymbol{T}$. 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 $\boldsymbol{S}$ and target $\boldsymbol{T}$. At first, τ is initialized to a 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 as 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 $\boldsymbol{S}$ is looking for the target $\boldsymbol{T}$. If **event tables** of these "helpers" contain the $\boldsymbol{T}$'s trace, they will forward $\boldsymbol{T}$'s traces to surrounding RFID tags. Using this way, searcher $\boldsymbol{S}$ gets help from other users. By sufficiently leveraging the "store-and-forward" properties, "crowdsourcing" capabilities can be obtained for navigation. Once the searcher $\boldsymbol{S}$ has collected some event message of $\boldsymbol{T}$, $\boldsymbol{S}$ would use Algorithm 2 to select the next neighbor vertex to move. Because $\boldsymbol{S}$ may be a helper for other users at the same time, he also needs to do steps 2 and 3 in the normal mode.

4.2. Event table and request queue

Under the above navigation framework, each user maintains an **event table** to record other users' event messages. Each entry in the **event 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 **request queue** to record request messages of searchers under the framework. Each entry in the **request 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.

4.3. Management of tag storage

Ideally, users' trace event messages should be stored on tags as many as possible. However, the memory size of RFID tag is limited, we should make full use of these storage resources. Under the navigation framework, the *post* operation will write messages to tags. Therefore, a reasonable writing and replacing strategy for post operation should be used. We divide the memory space of tag into two segments. Each segment consists of many storage units and the size of each unit is equal to the size of each event message or request message. Event messages and request messages are stored from either end of the tag memory space. There are a lot of page replacement algorithms in traditional operating system memory management, such as first in first out (FIFO), least recently used (LRU), and optimal page replacement (OPT). In this paper, we pay attention to the consumption rates of tag memory under a different writing and replacing strategy. According to the number of request message and event message stored on a tag, the writing and replacing strategy can be divided into the following four categories:

1. *One Request One Event* (*OROE*): For any request message $R\langle S, T, t \rangle$ on a tag, there is no other request message $R\langle S, T, t' \rangle$ stored on the same tag which satisfies $t \neq t'$. In other words, for each pair of searcher $\boldsymbol{S}$ and target T, only one request message can be stored on a tag. Similarly, for any event message $E\langle H_i, v_j, t \rangle$ stored on a tag, there is no other event message $E\langle H_i, v_j, t' \rangle$ stored on the same tag which satisfies $t \neq t'$. In other words, for each pair of user H_i and tag v_j, only one event message of H_i can be stored on the tag v_j.
2. *One Request Many Event* (*ORME*): ORME is similar to OROE. For each pair of searcher $\boldsymbol{S}$ and target $\boldsymbol{T}$, only one request message can be stored on a tag. However, ORME permits more than one event message $E\langle H_i, v_j, * \rangle$ of H_i to be stored on tag v_j.
3. *Many Request One Event* (*MROE*): Different from OROE, MROE permits more than one request message of searcher $\boldsymbol{S}$ and target $\boldsymbol{T}$ to be stored on a tag. For each pair of user H_i and tag v_j, only one event message of H_i can be stored on the tag v_j.
4. *Many Request Many Event* (*MRME*): MRME is the combination of ORME and MROE. It not only permits more than one request message of searcher $\boldsymbol{S}$ and target $\boldsymbol{T}$ to be stored on a tag, but also permits more than one event message $E\langle H_i, v_j, * \rangle$ of H_i to be stored on tag v_j.

The choice of writing and replacing strategy is related to many factors, such as the tag density in the environment, the number of

users, and the movement speed of users. If the tag density is small which means storage resource is very precious in the system, we should choose *One Request One Event* (OROE) strategy to ensure that users can get a fair chance to store at least one event message and one request message on tags. If the tag density is large, we should choose *Many Request Many Event* (MRME) strategy to store users' event messages and request messages as many as possible. If the movement speed of users is high and the searcher wants to spread out the request message to as many helpers as possible in a short time, maybe he should choose *Many Request One Event* (MROE) strategy to show the request is urgent. In this paper, we mainly use *One Request Many Event* (ORME) writing and replacing strategy in the experiment to verify our navigation framework.

Figure 2(a)–(e) shows the message interaction diagram when users pass by tags. In Fig. 2(a), when searcher A passes by tag v_1 at time t_1, searcher A posts an event message $E\langle A, v_1, t_1\rangle$ to tag v_1. Because searcher A wants to find target B, he also posts a request message $R\langle A, B, t_1\rangle$ to tag v_1. The event messages and request messages are stored from both ends to middle. In Fig. 2(b), target B posts an event message $E\langle B, v_2, t2\rangle$ to tag v_2 when he passes by tag v_2 at time $t2$. In Fig. 2(c), when helper H passes by tag v_2 at time t_3, helper H posts an event message $E\langle H, v_2, t_3\rangle$ to tag v_2. Besides,

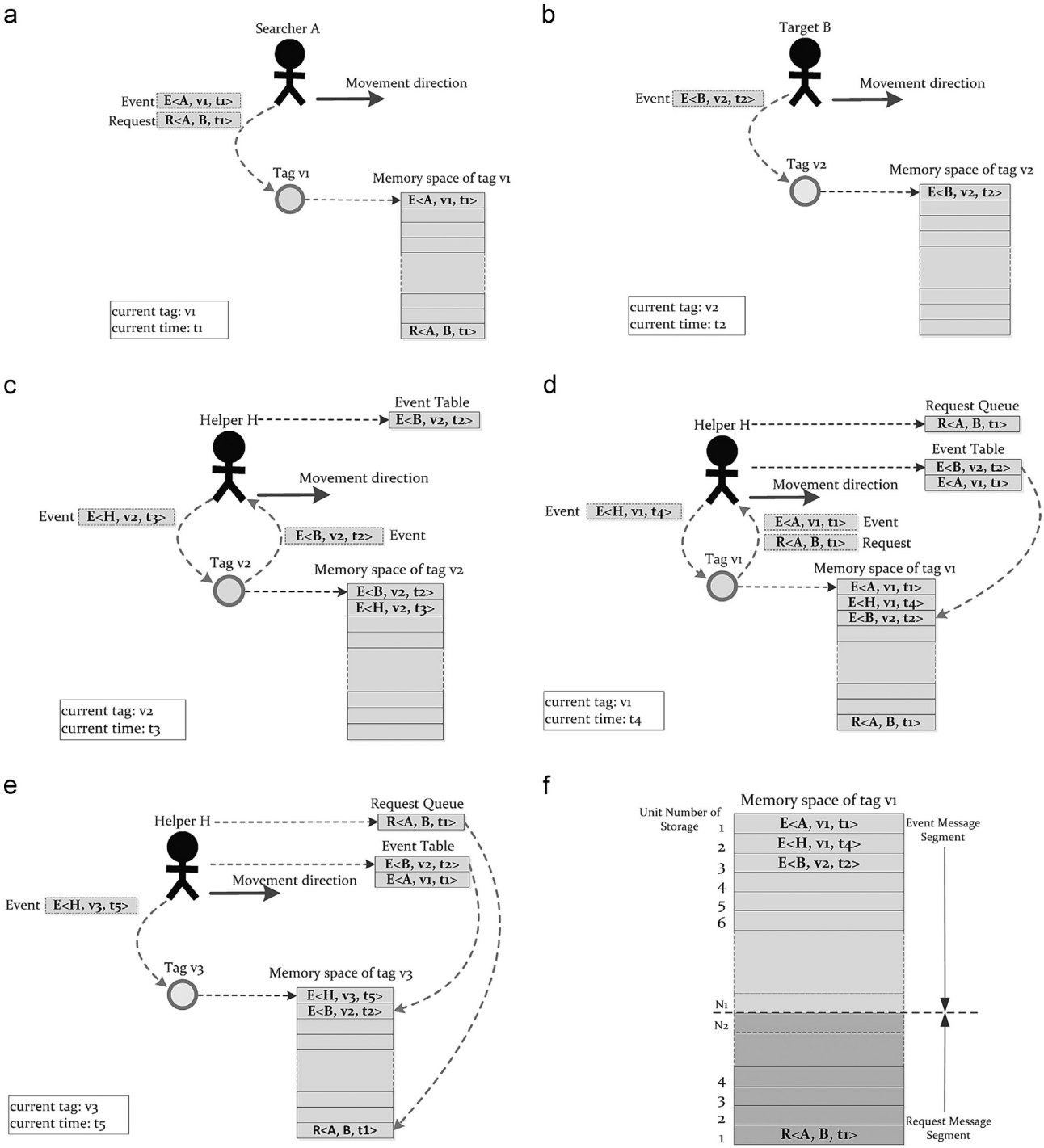

Fig. 2. Management of tag storage space using One Request Many Event (ORME) strategy. (a)–(e) Message interaction diagrams when searcher A, target B and helper H pass by a tag; (f) memory space of a tag ($N1/N2 = \lambda$). (a) when searcher A passes by tag v1 at time t1, (b) when target B passes by tag v2 at time t2, (c)when helper H passes by tag v2 at time t3, (d) when searcher H passes by tag v1 at time t4, (e) when searcher H passes by a new tag v3 at time t5 and (f) storage layout of two message segments.

helper H will also get the event message $E\langle B, v_2, t_2\rangle$ and add it to his **event table**. In Fig. 2(d), when helper H passes by tag v_1 at time t_4, helper H posts an event message $E\langle H, v_1, t_4\rangle$ to tag v_1. Besides, helper H will also get the event message $E\langle A, v_1, t_1\rangle$ and the request message $R\langle A, B, t_1\rangle$ of A, and add them to helper H's **event table** and **request queue**. Then helper H posts the event message $E\langle B, v_2, t_2\rangle$ to tag v_1. In Fig. 2(e), helper H posts the event message $E\langle B, v_2, t_2\rangle$ and request message $R\langle A, B, t_1\rangle$ to tag v_3, thus to attract more helpers to find B. In Fig. 2(f), the memory space of tag is divided into two segments: **event message segment** and **request message segment**. We will discuss more on the management of the memory space in the following:

Situation 1. *Normal Mode* 2*(b)*:

user H_i posts an event message $E\langle H_i, v_j, t\rangle$ to v_j;

Using the *One Request Many Event* (ORME) writing and replacing strategy, each tag maintains at most λ event messages of each user H_i. λ is a predefined parameter of ORME which is used to adjust the ratio between request messages and event messages. When the value λ equals to 1, the ORME strategy is reduced to OROE strategy. 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. The oldest event message of H_i stored on v_j will be replaced with $E\langle H_i, v_j, t\rangle$. Thus the size of **event message segment** is λ times as many as the size of **request message segment**.

Situation 2. *Normal Mode* 3*(a)*:

H_i posts a request message $R\langle S, T, t\rangle$ to v_k;

Using the *One Request Many Event* (ORME) writing and replacing strategy, 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. We take the same writing and replacing method in Searcher Mode step 2.

Situation 3. *Normal Mode* 3*(b)*:

*H_i selects event messages $\{E\langle T, *, *\rangle\}$ of the target **T** from H_i's **event table**; then H_i posts these event messages to v_k.*

In general, 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, when the available memory space size is reduced to a certain range, say 10%, we use a posting probability $p = \delta_i'/\delta_i, (0 \le p \le 1)$ to further reduce the number of posting events messages. δ_i is the total memory size of v_k, and δ_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.

4.4. 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 the traces searcher collects, the better the 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 is equal in Fig. 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; $V = \langle v_1, v_2, \ldots, v_m\rangle$; $U = \langle u_1, u_2, \ldots, u_n\rangle$; $\Gamma = \langle t_1, t_2, \ldots, t_n\rangle$;
Output: next neighbor v_{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 $v_j \in V$ of v_s **do**
7: $f(v_j) = d[v_s][v_j] + \sum_{i=1}^{n} \frac{\alpha_i}{\alpha} \times d[v_j][u_i]$;
8: **end for**
9: Select the smallest value $f(v_{min})$ from $f(v_1), \ldots, f(v_m)$; $v_{next} = v_{min}$;

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 vertices v and v'; all vertices in collection $V = \langle v_1, v_2, \ldots, v_m\rangle$ are the neighbors of vertex v_s. Besides, we use the **event table** of searcher ***S*** to predict the current location of the target. Let n $(n \le \theta_k)$ 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, $\Gamma = \langle t_1, t_2, \ldots, t_n\rangle$ is the list of timestamps, and $U = \langle u_1, u_2, \ldots, u_n\rangle$ is the list of vertices. Vertices in list U compose the target region.

For each neighbor v_j of v_s, we define a cost function $f(v_j)$ 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 $f(v_j)$ is defined below:

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

The neighbor vertex which has the lowest cost will be selected as the next intermediate vertex. For example, in Fig. 3, we assume that all the edges between two vertices on the graph are equal 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}.

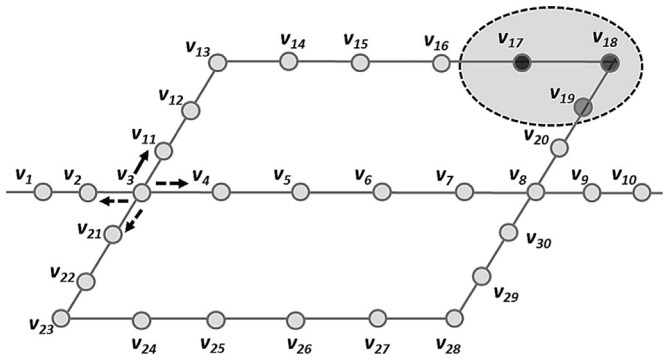

Fig. 3. Select the next position (vertex) and move to a target region.

4.5. Analysis

In the field of large-scale wireless ad hoc networks research, previous works (Dousse et al., 2004; Kong and Yeh, 2008; Zhao et al., 2011) have already showed the relationship between transmission delay and distance from source to destination. However, our problem is different from the above problems studied in previous works. The focus of our problem is locating and navigating to a mobile user. We analyze our distributed solution on the *undirected graph* $G=(V,E)$ which is an abstraction of the RFID-based delay tolerant network. On the graph, we want to find the optimal vertex sequence $\langle v'_1, v'_2, \ldots, v'_n\rangle, (\ (v'_i, v'_{i+1}) \in E, 1 \le i \le n-1)$, where v'_1 is the current location of the searcher $\boldsymbol{S}$, and v'_n is the finally location where the searcher found the target $\boldsymbol{T}$, so as to minimize the total distance of the vertex sequence. Let L be the total distance and our goal is to

$$\min L \Leftrightarrow \min \sum_{i=1}^{n-1} w[v'_i][v'_{i+1}] \tag{2}$$

subject to the memory size $\le \delta_i$ for each vertex v'_i.

Since the memory size of each RFID tag is limited, in order to ensure the locating and navigating performance in the RFID-based delay tolerant network, appropriate number of tags should be deployed. We use tag density ρ as the guide to deploy tags in the system. Let Ω be the volume of the indoor environment and $\overline{\delta}$ be the average memory size of tag. Obviously, we have $\rho * \Omega * \overline{\delta} = \sum_{v_i \in V} \delta_i$. To determine the value of ρ, we introduce a new concept *steady-state*. Considering an ideal situation, the memory size of each RFID tag is infinity, as the navigation system running, the total occupied memory size of tags $\phi(t)$ changes as well. We define that the system reaches its steady-state when the value $\phi(t)$ no longer increases. Then, if the system can reach its steady-state, the following conditions must be satisfied:

$$\sum_{v_i \in V} \delta_i \ge \max_{0<t<\infty} \phi(t) \tag{3}$$

Then we obtain

$$\rho \ge \frac{\max_{0<t<\infty} \phi(t)}{\Omega * \overline{\delta}} \tag{4}$$

However, it still remains as an open problem how $\max_{0<t<\infty}\phi(t)$ varies in accordance with network parameters, such as the number of people N in the network, the movement speed of each person ϑ, and the range of each person' activity area γ. Theoretically analyzing the exact relationship between them is very difficult. In Section 5, we use experiment result to reveal the above quantitative relation. In this paper, average searching time will be used to measure the performance of our solution. Intuitively, we can qualitatively describe the relationship between them as follows:

- The larger the number of people N is, the larger the tag density ρ is. Because more users means more event messages need to be stored on tags which leads to appropriate number of tags need to be deployed.
- The faster the speed of users' movement ϑ and the larger the range of users' activity area γ are, the larger the tag density ρ is. Because more event messages and request messages are generated in the same time which leads to more tags that are needed in the system.
- The tag density ρ is also related to which kind of writing and replacing strategy is chosen. For example, MRME strategy needs more tags than OROE strategy.

4.6. Discussion on practical issues

In realistic settings of the applications, several practical issues like the tag collisions and the localization accuracy might impact the actual performance of our indoor navigation solution. We thus provide a detailed discussion on these issues as follows:

In regard to the issue of tag collisions in MAC layer, since in our application settings, the RFID tags are not deployed in a rather dense approach, i.e., one tag is deployed for every 4–10 m, as we suppose to use a mobile reader (like smart phone) to continuously scan the surrounding tags, the scanning range is usually less than 3 m, the possibility of tag collision is very low in our application scenarios. Therefore, we mainly consider the performance in terms of application layer, the collision in physical layer will not have a great effect on the system performance.

In regard to the issue of localization accuracy, our main goal is to conduct efficient indoor navigation instead of indoor localization based on RFID systems. In our application settings, we deploy at least one tag for every 4–10 m, each tag is labeled with an exact position, the scanning range of a mobile reader is less than 3 m, hence the localization error is at most 3 m. As our indoor navigation solution is designed towards a large scale application scenario, for example, indoor navigation in a shopping mall or a large office building, this localization scheme is accurate enough for our application scenarios.

5. Performance evaluation

5.1. Large-scale experiment under classical human mobility model

We implement a simulator in Java to simulate the navigation process, thus to evaluate the performance of our solution. We conduct the experiment based on a 30×30 grid-graph. Each vertex on the graph represents a tag in the RFID-based delay tolerant network. Except the boundary vertices, each vertex has four neighbor vertices. We consider the simulation as an event-driven scenario, that is, when a user is approaching the vertex (tag) within a distance of less than 1 m, we allow the user to leave a trace or send a query to the surrounding tags. To simulate the movement behaviors of users on the graph, we use two classical human mobility models (Camp et al., 2002; Lee et al., 2009).

We compare the performance of our solution with the following solutions in regard to the average searching time:

- *Blind navigation*: The searcher randomly selects a vertex on the graph and moves to the vertex with the shortest path length. If the searcher does not find the target on the path, then the searcher will repeat the above process until he finds the target.
- *Centralized navigation*: 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.

- *Intuitive navigation*: The searcher leverages the RFID-based delay tolerant network to search the target, once the searcher gets one record of the target from the surrounding tags, he goes to the corresponding position directly. If he does not find the target at the specified position, he then further queries the surrounding tags for the target.
- *Escort* (Constandache et al., 2010): Once a user meets other users, they will exchange their own information with each other to help find the target. As the number of users increases, the probability to meet the target increases, this helps the searcher to quickly collect enough information to search the target.

Here, in the following we call our solution as *CrowdSensing*, we regard the *centralized navigation* as the approximate optimal solution and the *blind navigation* as the worst solution. The *intuitive navigation* is usually proposed in recent literatures for adaptive navigation, which is greedy by focusing on the present record, and usually efficient to find a target moving regularly. The *Escort* should be efficient in searching when the number of users is fairly large.

5.1.1. Experiment settings

At the beginning of the experiment, 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 to 2 m/s and other users' to 1 m/s. If searcher and target happen to be on the same edge, we think that the searcher has found the target.

To simulate the movement behaviors of users, we use a simplified Random Walk (RW) Mobility Model and a Random Way-Point (RWP) Mobility Model. 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 the shortest path length. 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 $p(t) = C \times t^{-2.5}(s)$, $(5 \le t \le 60)$. Parameters used in the experiment are shown in Table 1. For all figures presented, we run the experiment 1000 times to get average values, and provide the 95% confidence interval of the experiment results.

5.1.2. Experiments result

Average searching time: In Fig. 4(a) and (b), 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 figures, with the number of users increasing, the average searching time of *CrowdSensing* is reduced gradually. When increasing the number of users from 50 to 500, *CrowdSensing*'s average searching time is reduced from 722/499 s to 216/230 s. 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.

Specifically, the *blind navigation* just randomly chooses a way to find the target, which is irrelevant to the number of users and leads to stable average searching time. The *centralized navigation* always achieves the least searching time, since it is always close to the optimal solution. With the number of users increasing, the average searching time of *CrowdSensing*, *Escort* and *Intuitive Navigation* are all reduced gradually. *Escort* suffers from rare users at first, which leads to long searching time. However, as the number of user increases, the number of users is close to the intersections in the graph, which leads to close searching time of *CrowdSensing*. The *intuitive navigation* can only follow the one historical record of the target, which leads to relatively high searching time in RW model, because the target keeps on moving. However, in RWP model, the target will not move so often, which makes the searching time smaller than the RW model. Since *CrowdSensing* can make use of all the crowd-sourcing information, it achieves better performance than the above two solutions.

In Fig. 4(c) and (d), we plot the average searching time of *CrowdSensing* 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 s to 200 s, the average searching time is reduced more than 30%. Different from θ_R, the impact of θ_k is little. When the values of θ_k are 1, 2, 6, and 8 under RW or 1, 3 and 4 under RWP, the average searching time is slightly smaller than others.

In Fig. 4(e), we plot the average searching time of *CrowdSensing* 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 weak, say $\mu \ge 30$, 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.

Number of messages: In Fig. 4(f), 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 and 350 s. After 400 s, the number of event messages is essentially unchanged. Because no more request messages are generated after that.

Table 1
Parameters used in the experiment.

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]

5.2. Realistic experiment under shopping mall mobility model

To evaluate the performance of *CrowdSensing* under real world environment, we conduct our experiment using a mobility model for shopping mall environments founded on real-world human traces which is presented by Galati et al. (2013). They ran a field trial to collect Bluetooth contact data from shop employees and clerks in a shopping mall over six days. The field trial started on Monday the 15th of June 2009 and went on till Saturday the 20th of June 2009. The traceset can be accessed from community resource for archiving wireless data at Dartmouth (CRAWDAD) (Galati and Greenhalgh, 2013). They analyzed the collected contact traces and designed the shopping mall mobility model.

5.2.1. Shopping mall mobility model

In the shopping mall environment, there are twenty-five mobile devices running symbianOS and using Bluetooth, eighteen of which

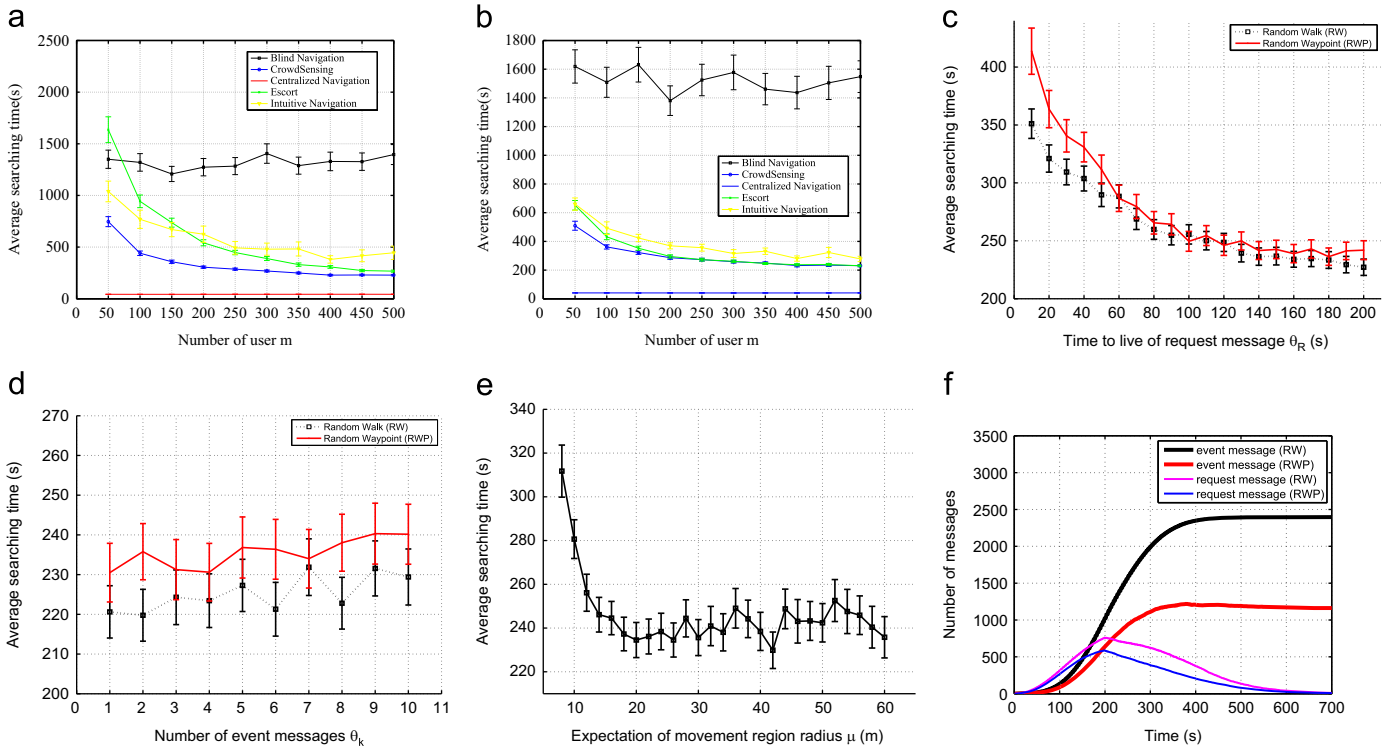

Fig. 4. Comparison of the average searching time under Random Walk (RW) Mobility Model and Random Way-Point (RWP) Mobility Model with different parameters (number of users m; time to live of request message θ_R; number of event messages θ_k; expectation of movement region radius μ). (a) RW, $\theta_R = 200$ 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.

Table 2
Parameters of shopping mall mobility model (Galati et al., 2013).

Entity	Distribution	Parameters	K–S test distance
Sellers within their workplace	Lognorm	$\mu = 7.086$, $\sigma = 1.876$	0.1405
Sellers out of their workplace	Lognorm	$\mu = 6.504$, $\sigma = 0.51$	0.1902
Customers' interarrival time	Exponential	$\beta = 1.771e-03$	0.1144
Customers within the mall	Weibull	$\alpha = 0.935$, $\beta = 2.579e+03$	0.0639
Customers within a single shop	Weibull	$\alpha = 1.002$, $\beta = 3.059e+02$	0.3519

were carried by the shopkeepers and shop employees and seven of which were static, placed in fixed locations (Galati et al., 2013). All of the shop employees working in two large stores, eleven smaller shops and one bar. The surface area of the shopping center is 10,880 m^2. During the six-days' experiment, 749 customers were detected by the phones carried by sellers. The shopping mall mobility model uses exponential, lognormal and Weibull probability density function to mathematically model the movement of individuals in a shopping mall. The optimal parameters for the empirical distribution were inferred by the Maximum-Likelihood Method for each distribution (Galati et al., 2013). Table 2 is the parameters of shopping mall mobility model. In our experiment, we set the speed of searcher to 2 m/s. The speed and the pause of individuals were randomly generated according to a uniform distribution respectively between 1.65–1.15 m/s and 0–2 s. Similarly, in the experiment we consider the simulation as an event-driven scenario, that is, when a user is approaching the vertex (tag) within a distance of less than 1 m, we allow the user to leave a trace or send a query to the surrounding tags. For all figures presented, we run the experiment 1000 times to get average values, and provide the 95% confidence interval of the experiment results.

5.2.2. Experiment result

Average searching time: In Fig. 5(a), we plot the average searching time of three solutions under the shopping mall mobility model. We guarantee that the inputs of users' movement behaviors for each solution are the same. As we can see the figures, under the shopping mall mobility model, *CrowdSensing* can significantly reduce the average searching time than the *blind navigation*. The result falls in line with the previous experiment result under classical human mobility model RW and RWP. The average searching time of *CrowdSensing* is 126 s which is 20% of *blind navigation* 694 s.

In Fig. 5(b), we plot the average searching time of *CrowdSensing* under shopping mall mobility models with different values of θ_R. Similar results have been seen in classical mobility model, 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 5 s to 100 s, the average searching time is reduced from 220 s to 137 s.

Number of messages: In Fig. 5(c), we plot the average number of messages stored in the delay tolerant network as time grows.

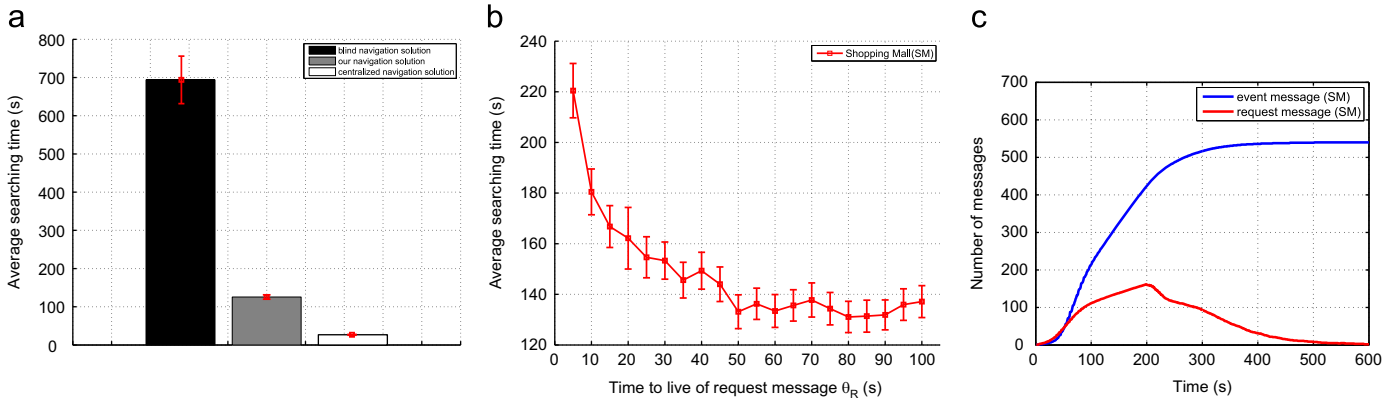

Fig. 5. Evaluation under shopping mall mobility model (SM) with different parameters. (a) SM, 25 sellers, $\theta_R = 200(s)$, $\theta_k = 10$. (b) SM, 25 sellers, ORME, $\theta_k = 10$. (c) SM, 25 sellers, $\theta_R = 200(s)$, $\theta_k = 10$.

When the time is about 200 s, the number of request messages reaches its maximum 162. The number decreases as the request message's time to live is reduced to zero. The number of event messages grows fast between 50 s to 250 s. After 350 s, the number of event messages is essentially unchanged. Because no more request messages are generated after that.

5.3. Discussion

From the above large-scale experiment under classical human mobility model and the realistic experiment under shopping mall mobility model, we find that the number of users and the time to live of request message are two critical factors affecting the performance of *CrowdSensing*. In order to reduce the average searching time, appropriate number of tags and value of time to live of request message should be given in the environment. The experimental results can be summarized in more detail as follows:

- *Tag density should adjust with the degree of user's movement activity and user scale*: The degree of user's movement activity reflects the speed of generating event messages and request messages of each user. The user scale reflects the speed of generating messages of all users. Tag density must be qualified for the storage requirement of these messages. For example, in our experiments, with the number of users increasing, the average searching time of our navigation solution is reduced gradually. But when the number of users increased to one-third of the number of tags, the decrease of average searching time is not obvious. Thus increasing the tag density is a method to improve the performance of navigation, but it may not dramatically reduce the average searching time.
- *Appropriate value of time to live of request message should be set*: Time to live of request message has great influence on system performance of average searching time. But the value of time to live of request message is not as bigger as better, because too large time to live of request message will lead to longer survival time of messages in the system and require a much larger storage space. On the contrary, too small time to live of request message will lead to few helpers who can receive request messages and the average searching time increases. For example, in our experiments, the value of time to live of request message should not be less than 40 s.
- *The proportion of memory space for event messages and request messages is determined by writing and replacing strategy of tags*: For example, in our experiments, at the beginning, the number of event messages is less than the number of request messages. But the number of event messages increases quickly after 50 s in realistic experiment under shopping mall mobility model. The peak number of event messages is 3 times as many as the peak number of request messages. The parameter λ used in writing and replacing strategy *One Request Many Event* (ORME) is in fact 3.

6. Conclusion

This paper proposes a framework called *CrowdSensing*, using RFID-based delay tolerant network for indoor navigation. By sufficiently leveraging the "store-forward" properties of delay tolerant network, *CrowdSensing* provides an effective mechanism for indoor navigation. Experiment results show that *CrowdSensing* can efficiently reduce the average searching time of navigation. While comparing with recent approaches like *Escort* (Constandache et al., 2010) and *Intuitive Navigation*, *CrowdSensing* can effectively reduce the average search time by 24% and 31% respectively. Furthermore, *CrowdSensing* is very effective in controlling the number of event and request messages by sufficiently leveraging the RFID tags' limited memory space.

Acknowledgment

This work is partially supported by National Natural Science Foundation of China under Grant nos. 61100196, 61472185, 61321491, 91218302, 61373129; Jiangsu Natural Science Foundation under Grant no. BK2011559; Key Project of Jiangsu Research Program under Grant no. BE2013116; EU FP7 IRSES MobileCloud Project under Grant no. 612212.

References

Azizyan M, Constandache I, Roy Choudhury R. SurroundSense: mobile phone localization via ambience fingerprinting. In: Proceedings of ACM MobiCom; 2009. p. 261–72.
Biswas J, Veloso M. WiFi localization and navigation for autonomous indoor mobile robots. In: Proceedings of IEEE international conference on robotics and automation (ICRA); 2010. p. 4379–84.
Bogo F, Peserico E. Optimal throughput and delay in delay-tolerant networks with ballistic mobility. In: Proceedings of ACM MobiCom; 2013. p. 303–14.
Camp T, Boleng J, Davies V. A survey of mobility models for ad hoc network research. Wirel Commun Mob Comput 2002;2(5):483–502.
Constandache I, Bao X, Azizyan M, Choudhury RR. Did you see bob?: human localization using mobile phones. In: Proceedings of ACM MobiCom; 2010. p. 149–60.
Dousse O, Mannersalo P, Thiran P. Latency of wireless sensor networks with uncoordinated power saving mechanisms. In: Proceedings of ACM MobiHoc; 2004. p. 109–20.
Fischer G, Dietrich B, Winkler F. Bluetooth indoor localization system. In: Proceedings of workshop on positioning, navigation and communication; 2004. p. 147–56.
Galati A, Greenhalgh C. Crawdad data set Nottingham/mall (v. 2013-02-05); 2013.
Galati A, Djemame K, Greenhalgh C. A mobility model for shopping mall environments founded on real traces. Netw Sci 2013;2(1–2):1–11.

Jiang X, Liang C-JM, Zhao F, Chen K, Hsu J, Zhang B, et al. Demo: Creating interactive virtual zones in physical space with magnetic-induction. In: Proceedings of ACM SenSys; 2011. p. 431–2.
Jiang X, Liang C-JM, Chen K, Zhang B, Hsu J, Liu J, et al. Design and evaluation of a wireless magnetic-based proximity detection platform for indoor applications. In: Proceedings of ACM IPSN; 2012. p. 221–32.
Ji H, Xie L, Yin Y, Lu S. An efficient indoor navigation scheme using RFID-based delay tolerant network. In: Proceedings of IEEE GLOBECOM; 2013. p. 183–8.
Kim C-M, Kang I-S, Han Y-H, Jeong Y-S. An efficient routing scheme based on social relations in delay-tolerant networks. In: Ubiquitous information technologies and applications; 2014.
Kong Z, Yeh EM. On the latency for information dissemination in mobile wireless networks. In: Proceedings of ACM MobiHoc; 2008. p. 139–48.
Lee H-J, Lee MC. Localization of mobile robot based on radio frequency identification devices. In: International joint conference SICE-ICASE; 2006. p. 5934–9.
Lee K, Hong S, Kim SJ, Rhee I, Chong S. Slaw: a new mobility model for human walks. In: Proceedings of IEEE INFOCOM; 2009. p. 855–63.
Minami M, Fukuju Y, Hirasawa K, Yokoyama S, Mizumachi M, Morikawa H, et al. Dolphin: a practical approach for implementing a fully distributed indoor ultrasonic positioning system. In: Proceedings of ACM UbiComp; 2004. p. 347–65.
Ng WW, Ding H-L, Chan PP, Yeung DS. Efficiency of applying virtual reference tag to neural network based RFID indoor positioning method. In: Proceedings of international conference on machine learning and cybernetics (ICMLC), vol. 1; 2011. p. 447–53.
Ni LM, Liu Y, Lau YC, Patil AP. Landmarc: indoor location sensing using active RFID. Wirel Netw 2004;10(6):701–10.
Priyantha NB, Miu AK, Balakrishnan H, Teller S. The cricket compass for context-aware mobile applications. In: Proceedings of ACM MobiCom; 2001. p. 1–14.
Saad SS, Nakad ZS. A standalone RFID indoor positioning system using passive tags. IEEE Trans Ind Electron 2011;58(5):1961–70.
Tong HC, Wang D. A novel RFID indoor positioning system based on doppler effect. Appl Mech Mater 2014;513:3292–5.
Xie L, Sheng B, Tan CC, Han H, Li Q, Chen D. Efficient tag identification in mobile RFID systems. In: Proceedings of IEEE INFOCOM; 2010. p. 1–9.
Xie L, Li Q, Chen X, Lu S, Chen D. Continuous scanning with mobile reader in RFID systems: an experimental study. In: Proceedings of ACM MobiHoc; 2013. p. 11–20.
Yang Z, Ning T, Wu H. Efficient rostering of mobile nodes in intermittently connected passive RFID networks. IEEE Trans Mob Comput 2013;12(10):2012–23.
Zhao S, Fu L, Wang X, Zhang Q. Fundamental relationship between nodedensity and delay in wireless ad hoc networks with unreliable links. In: Proceedings of ACM MobiCom; 2011. p. 337–48.
Zhu W, Cao J, Xu Y, Yang L, Kong J. Fault-tolerant RFID reader localization based on passive RFID tags. In: Proceedings of IEEE INFOCOM; 2012. p. 2183–91.
Zou H, Wang H, Xie L, Jia Q-S. An RFID indoor positioning system by using weighted path loss and extreme learning machine. In: Proceedings of cyber-physical systems, networks, and applications (CPSNA); 2013. p. 66–71.