中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-06-014

Design and Implementation of RFID-Based Object Locator

T. S. Chou and J. W. S. Liu

November 29, 2006 || Technical Report No. TR-IIS-06-014

http://www.iis.sinica.edu.tw/LIB/TechReport/tr2006/tr06.html

Design and Implementation of RFID-Based Object Locator

T. S. Chou and J. W. S. Liu, *Fellow, IEEE*

***Abstract*—An object locator is a device designed to assist its user in finding misplaced household and personal objects in a home. This paper describes alternative designs and a proof-of-concept prototype of object locators based on the RFID technology. Advantages of such locators include extensibility and low maintenance. The numeric model provided here can be used to determine figures of merits, including costs, search time and energy consumption. The results of analysis based on the model can serve as design guides.**

I. Introduction

In the coming decades, an increasingly larger number of baby boomers will grow into old age. This trend has led to an increasing demand for products (e.g., [1-6]) and services designed to help the elderly live independently. One of such products is *object locator*. An object locator can assist its users in finding misplaced household and personal objects in a house. Specialty stores and websites now offer object locators such as the one shown in Fig. 1 for $ 50 US each. A locator contains an interrogator with several buttons of different colors and a tag of the color matching the color of each button. By attaching a tag to an object to be tracked, the user can look for the object by pressing the button of matching color on the interrogator. The tag attached to the object beeps and flashes in response and thus enables the user to find the object.

Fig. 1 An object locator

The existing object locator is not ideal in many aspects: The number of buttons on the interrogator is fixed; extending the locator to track more than that number of objects is impossible. Tags are battery-powered. A tag might become a lost object itself after it runs out of battery. When a tag breaks, a user must purchase a replacement tag of the same color as the broken one. If a user were to use two tags of the same color, both tags would respond to the search signal from the interrogator. This situation is clearly not desirable.

This paper describes three designs and a proof-of-concept prototype of object locators based on the *RFID (Radio Frequency Identification)* technology. Our object locators do not have the drawbacks of the existing object locator. In particular, RFID-based object locators are extensible, reusable, and low maintenance. They are extensible in the sense that the maximum number of tracked objects is practically unlimited and that a RFID-based object locator can support multiple interrogators. The interrogator software can run on a variety of platforms (e.g. desktop PC, PDA, smart phone and so on). Reusability results from the fact that all RFID tags used for object locators can have globally unique ids. Hence, tags never conflict, and a tag can be used in more than one object locators. Low maintenance is one of the advantages of RFID technology. One of the designs uses only passive RFID tags; the user is never burdened by the concern that a tag may be out of battery.

The object locator designs described here call for different hardware components and hence have different overall cost. The differences in their hardware capabilities and object search schemes lead to differences in search time and energy consumption. We provide a numeric model that can be used to determine the tradeoff between these figures of merit. The results of analysis based on the model can serve as design guides to developers of RFID-based object locators. Through this analysis, we identify the design that is most practical for the current state of RFID technology. Today, object locators based on all designs are too costly. We project what it takes to make RFID-based object locators as affordable as the locators one can now find in stores.

Our object locator resembles location detection systems in its goal: assists users to locate objects. Many different location detection systems are available today. Global Positioning System (GPS) [7] is the most well known. Priced at about $ 100 US each, GPS navigators are widely used in cars, buses and so on. However GPS has its limitations. Reflection, occlusion and multipath effects seriously interfere with distance measurement and make GPS ineffective indoors. For this reason, indoor location detection systems use a variety of other technologies, including infra-red (e.g. Active Badge [8]), ultrasound (e.g. Bat [9, 10] and Cricket [11]), and radio frequency (e.g. RADAR [12]). Mote is a well-known sensor. It is used in MoteTrack [13] for the indoor location detection purpose. WLAN can be used to build location detection system, also. SpotON [14] and Nibble [15] are examples. Compared with these location detection systems, an object locator must be a far more low

T. S. Chou is with Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-email: g934343@oz.nthu.edu.tw).

J. W. S. Liu is with Institute of Information Science, Academia Sinica, Academia Road, Section 2, Box 128, Nankang, Taipei, Taiwan (Phone: +886-2-2788-3799, e-mail: janeliu@iis.sinica.edu.tw)

cost solution and must be ultra easy to set up and use. Many indoor location detection systems (e.g. Bat and Active Badge) rely on a big infrastructure or a pre-computed database (e.g. RADAR) to support location estimation. These systems are too costly to deploy and maintain and hence, unsuitable for home use. Cricket system provides low cost location-aware service. An object with a receiver can resolve its location. This is not what an object locator does. A misplaced object does not need to know its own location; the user looking for it needs to know.

The rest of this paper is organized as follows. Section II describes scenarios that illustrate how a RFID-based object locator may be used. Section III presents three designs of RFID-based object locators. Section IV describes the implementation of a proof-of-concept prototype based on one of the designs. It also describes the reader collision problem [16] encountered in the prototype and the solution we use to deal with to the problem. Section V describes a numeric model for computing energy consumption and search time and compares the merits of the designs. Section VI concludes this paper and discusses future works.

II. User Scenarios

The routine usage of an object locator requires only three operations: Add, Delete and Query. We describe these operations here to illustrate how a locator may be used. Without loss of generality, we assume that a new object locator kit contains a portable interrogator, a dozen of RFID tags and agents. The interrogator resembles a smart phone. Like smart phones, it has a small non-volatile storage and a RF transceiver together with a network address. We will return in the next section to describe how the RF transceiver is used, as well as agents and their functions. Unlike common smart phones, however, the interrogator has a RFID reader. The reader is used for the Add operation described below.

Fig. 2 shows parts of the user interface on an interrogator with a LCD touch screen and two buttons. The LCD touch screen is used as both input and output user interface. A user can use a pen to select an item among the items displayed on the screen, the button at the bottom left corner to confirm a selection, and the button at the bottom right corner to cancel the selection. Some operations need text input. The virtual keyboard shown on right is for this purpose.

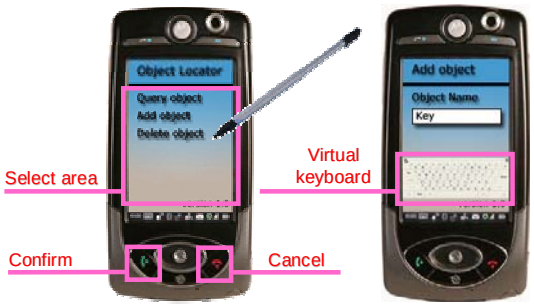


Fig. 2 The user interface

Add operation works in a similar way as the address book of a smart phone. Using this operation, the user can add the registration of an object to be tracked into the interrogator. By registration, we mean a mapping between the ID of the tag attached to an object and the name of the object. The user queries the locations of objects by their names. In response to a query, the interrogator uses the object-name-tag-id mappings to resolve which one of the registered objects to search. Fig. 3 shows a scenario: The user picks an unused tag and attaches it to an object to be tracked as shown in Fig. 3(a) and (b). Then, the user puts the tag close to the interrogator and selects Add object. This step is shown in Fig. 3(c). In response to Add object command, the interrogator reads the id of the tag, displays a new text field and prompts the user to enter a name (e.g., Key). When the user confirms the name, the interrogator creates a mapping associating the name with the id of the tag attached to the object, and stores the mapping in its local non-volatile memory. This is illustrated in Fig. 3(d). The user repeats the above steps to register each object until all objects to be tracked are registered.

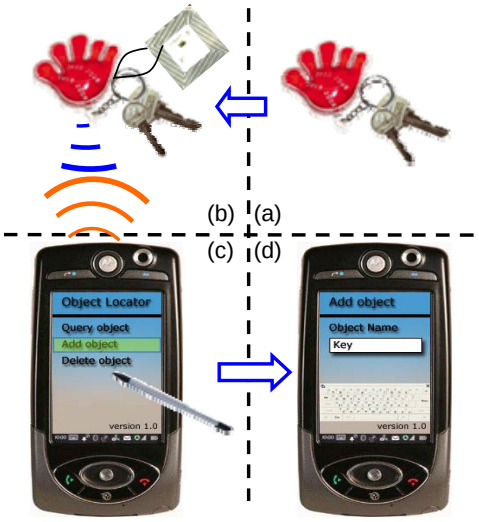


Fig. 3 Add operation

Delete operation removes the registration of an object, i.e., the object-name-tag-id mapping stored in the interrogator: The user can invoke the operation by pressing Delete object on the touch screen. In response, the interrogator displays the list of registered objects, allowing the user to select the object (e.g. Key) to be deleted. The interrogator deletes the mapping after the user confirms the selection. Delete operation frees the tag attached to the now unregistered object and makes the tag free for use to track some other object.

Query operation, illustrated by Fig. 4, is the work horse. The user presses Query object on the touch screen, as illustrated by Fig. 4(a), to invoke this operation for assistance in finding misplaced objects. When the names of registered objects are displayed, the user selects the object to be searched; in this example, it is Key. After the user confirms the selection (as shown in Fig. 4(b)), the interrogator retrieves from its local storage the id of the tag attached to the object

with the selected name and starts a search for the tag with that id. Hereafter, we call the tag being searched the *queried tag* and the object attached to the tag the *queried object*. We will return to describe the search process in the next section. Object locators of different designs present the result of Query operation in different ways. As examples, Fig. 4(c) and (d) shows two different responses. In Fig. 4(c), the queried tag beeps, allowing the user to look for it by following the sound. This version works like the existing locator described in Section 1. In Fig. 4(d), the interrogator directs the user to the place (e.g. bedroom 1) where the queried object is.

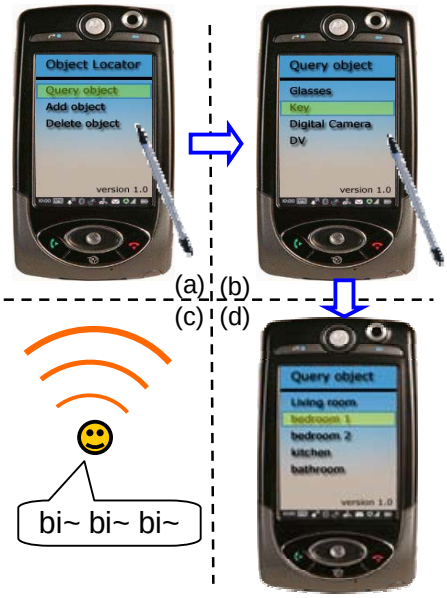


Fig. 4 Query operation

III. Alternative Designs

The three designs of object locator are called *Room-level Agents, Interrogator and Tags* (RAIT) locator, *Desk-level Agents, Interrogator and Tags* (DAIT) locator and *Desk-level and Room-level Agents, Interrogator and Tags* (DRAIT) locator. As their names imply, each of the locator consists of tags, agents and at least one interrogator. The adjectives room-level and desk-level describe the ranges of RFID readers used by the designs. The ranges of room-level readers and desk-level readers are sufficient large to cover a typical-size room or desk, respectively.

The term tag refers specifically to RFID tags. Each tag has a unique id, hereafter called TID. One of the designs uses only passive tags. The other designs call for tags that can beep upon receiving query messages containing their TIDs. It is possible to implement such tags using semi-passive RFID tags since the battery in such a tag can be used not only to improve read range but also to drive a beeper.

An *agent* is a device that aids the interrogator in locating the queried object (i.e., the queried tag). Each agent has a RF transceiver, together with a programmable network address, a RFID reader, and a RFID tag. As stated in Section 2, the interrogator also has a RF transceiver with a network address. This allows the interrogator and all agents to form a wireless local area network (LAN). The network address of the interrogator (or each interrogator in a multiple-interrogator system) is unique and so is the network address of each agent. We assume that the network provides reliable communication. Other aspects of the wireless LAN are irrelevant to our discussion and hence are omitted here.

The interrogator requests assistance from an agent by sending the TID of the queried tag to the agent via the wireless LAN. The RFID reader in the agent enables the agent to search for the tag within its coverage area.

A. RAIT Locator

A disadvantage of the existing locator is that a user needs to walk around the house when searching an object and the interrogator needs to repeatedly send the query signal until the user hears the queried tag or gives up the search. RAIT locator is designed to eliminate this disadvantage. As shown in Fig. 5, RAIT locator uses one or more agents to cover each room, and the house is fully covered by agents. When the user invokes Query operation, the interrogator sends a query message containing the TID of the queried tag to agents and thus requests the agents to search the queried tag on its behalf. Each agent broadcasts an addressed mode read request with the TID retrieved from the query message to read the tags within range. The tag with id matching the TID beeps upon receiving a read request as well as responding to the agent. The agent finding the queried tag reports its network address to the interrogator. This information enables the interrogator to display the results as shown in Fig. 4(d), telling the user to go to a specified room in the home.

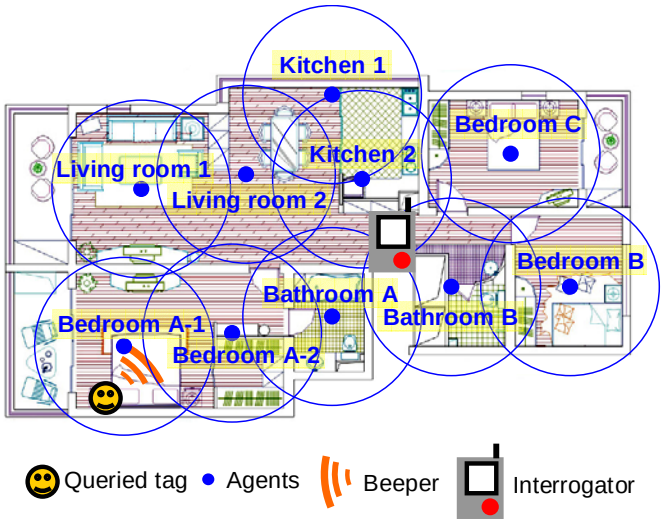


Fig. 5 The configuration of RAIT locator

Obviously, the agents must be set up for a RAIT locator to be usable. Fig. 6 lists the steps carried out by the user and work done by the system during the set up process. The goal of Steps 3-5 is to make sure that there is no blind region. A *blind region* is an area where tags cannot be read by any agent. The corners of a room are the most probable blind regions. This is the rationale behind Step 3. When the TEST READ

RANGE switch of an agent is on, the agent repeatedly broadcasts non-address mode read messages. In this way, the agent enables the user to determine whether any of the corners is a blind region.

1. Choose a location near middle of a room and temporarily attach an agent to the ceiling or furniture at the location.
2. Turn on *TEST READ RANGE* switch on the agent.
3. Pick up a tag and check whether the tag beeps at each corner of the room.
4. If no, adjust the location of the agent or add one more agent at another location in the room and turn on *TEST READ RANGE* switch on the additional agent. Then go back to Step 3. If yes, turn off *TEST READ RANGE* switch.
5. Securely attach the agents tested in Step 2-5 at their respective locations.
6. Put the interrogator near the agent and execute *Register Agent operation*.
7. Repeat step 1 to 6 until all agents covering the house are registered.

Fig. 6 Agent set-up process

The Register Agent operation in Step 6 is similar to Add operation described in Section II. Its goal is to assign a human-readable location name to an agent, so that the interrogator can later generate query results illustrated by the example in Fig. 4(d). During the operation, the interrogator prompts the user to provide a unique name for the location of each agent. For example, if the living room needs two agents, Living Room R(ight) and Living Room L(eft) are good names for them.

The interrogator also assigns a unique network address to the agent being registered. The id of the tag in an agent is the product serial number of the agent. The interrogator uses the id to distinguish the agent from previously registered agents. By assigning successive network addresses to agents as they are registered and initialized one by one, successive Register Agent operations enable each initialized agent to join the LAN and later compute the addresses of other agents by adding or substituting some number from its own address.

Fig. 7 depicts the format of messages in a RAIT locator. This format supports multiple interrogators: the src_addr allows agents to identify the interrogator issuing the query message, the dest_addr allows them to address their responses to a specified interrogator, and the offset and data field allow interrogators to synchronize their databases created by Add and Register Agent operations.

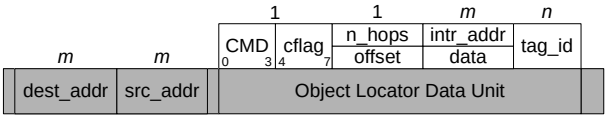


Fig. 7 The message format

A RAIT locator can search a queried object in three ways: broadcast, relay and polling. The *broadcast scheme* is the most straightforward. The interrogator broadcasts a query message with the tag_id field filled with TID of the queried tag. The agents finding the queried tag report their agent ids to the interrogator and the others do not reply.

The knowledge on the agent network addresses and the number of agents enables an interrogator to request assistance from agents one at a time using the *relay scheme*: To search for a queried tag, the interrogator sends a query message containing its own address in intr_addr field, the number of agents to be queried in n_hops and the TID of the queried tag in tag_id to the first agent. The simplest choice is the agent with the smallest address. In response to a query, each agent searches for the tag with the TID in its own cover area. The agent reports its own address to the interrogator if it finds the tag; otherwise it decreases n_hops by one, increments its own network address by one to get the address of the next agent and then forwards the query message to the next agent.

According to the *polling scheme*, the interrogator also sends a query message to the first agent in its polling list, provides the agent with the TID of the queried tag and waits for response from the agent. The agent replies to the interrogator no matter whether it finds the tag or not. If the response from an agent is negative, the interrogator sends the query message to the next agent in its polling list. Advantage of the polling scheme over the relay scheme is that the interrogator can dynamically alter the search sequence.

B. *DAIT and DRAIT locators*

DAIT locator, shown in Fig. 8, is an extension of RAIT locator. The designs are similar in how the Query operation is handled by the interrogator and agents. A DAIT locator can also use any of the search schemes mentioned above. DAIT differs from RAIT primarily in the required read ranges of agents. The read range of agents used in a DAIT locator is less than one meter. Agents with such a small range offer higher accuracy in locations of queried tags. Information on the agent that finds the queried tag tells the user the location of the searched object within a small vicinity of the agent. Tags in DAIT locators are passive; they do not beep because a user can easily find the misplaced object even though the tag does not beep. Because tags do not need to beep, they can be battery free. This is a major advantage of DAIT locator.

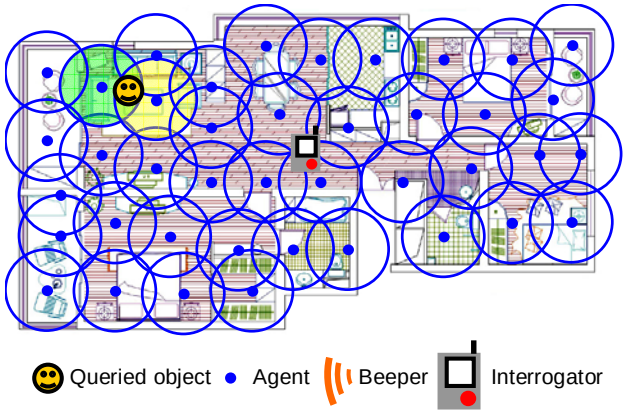


Fig. 8 The configuration of DAIT locator

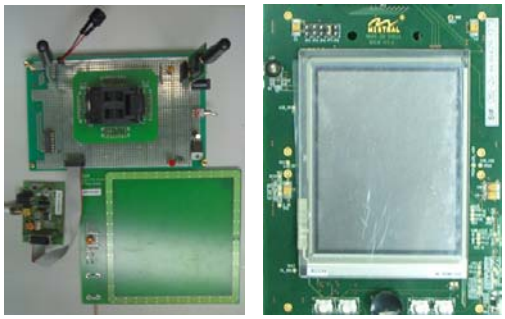
However, it is significantly more complicated to set up desk-level agents. Blind regions of RAIT locator are easy to detect and eliminate because a blind region is typically created by walls and is near the read boundary of an agent. In the case of DAIT locator, a room cannot be fully covered by one or two agents. Any three adjacent agents may create a blind region. Our solution is to give a user a circular thread

whose circumference is less than $3\sqrt{3}$ (i.e., the circumference of a regular triangle whose center is one unit away from its corners) times their read range and instruct a user to set any three adjacent agents within the circular thread. By doing so, blind regions never occur.

DRAIT locator is a hybrid design of DAIT and RAIT locators. A DRAIT locator contains both room-level and desk-level agents. When such a locator is used, the interrogator asks desk-level agents to search first. The interrogator asks room-level agents only when no desk-level agent finds the queried object. We set up several desk-level agents on furniture in addition to setting up room-level agents as described above. Because misplaced objects are often on furniture or the vicinities of them, in the majority of the cases, the queried object can be found by a desk-level agent, and the tag on it does not need to beep. Thus, this design extends the life time of a semi-passive tag.

IV. Prototype Implementation

We implemented a proof-of-concept prototype of DAIT locator. We chose to implement this design because it does not require customized semi-passive tags. Indeed, all components used in our prototype are readily available today. Parts (a) and (b) of Fig. 9 shows an agent and the portable interrogator of our prototype, respectively. The agent is composed of a microcontroller, a RF transmitter, a RF receiver and a RFID reader module. The microcontroller is ATMEL ATmega128. It runs at 8MHz and has 128k bytes flash / 4k bytes EPPROM. The RF transmitters and receivers interconnecting interrogator(s) and agents are LINX TXM(RXM)-433-LR, which use 433MHz ASK. RFID reader modules are MELEXIS EVB90121, which is ISO15693-compliant and uses a directional antenna. We use TI OMAP5912 and NEC Q-VGA to implement the portable interrogator. The current version of our prototype supports the three operations described in Section II and uses the polling search scheme.



(a) (b)

Fig. 9 The agent and interrogator

The lack of customized antenna design for tags and readers and the reader collision problem seriously affects the performance of our prototype. When the antennae of tags and readers are directional, the read performance of agents depends on the orientation of the antennae. Clearly, it is impossible to ensure optimal or near optimal alignment of the tag antennae towards the agents covering their locations. Hence tags in a DAIT object locator should have omni-directional antennae. Our DAIT prototype uses only tags with directional antennae. (Again, the reason is that such tags are readily available.) Agents with omni-directional antennae can be simply set on the furniture as shown in Fig. 10(a). Agents with directional antennae should be attached to the ceiling as shown in Fig. 10(b). This arrangement requires a read range of 2-3 meters. With readers of a sufficiently large read range, RAIT locators can use tags with directional antennae without performance concern.

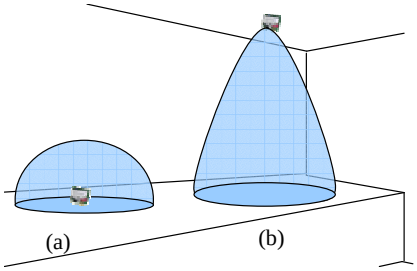


Fig. 10 The arrangement of agents

Close proximity of readers (i.e., agents) is necessary in order to avoid blind regions. Our DAIT prototype is no exception. When RFID readers have overlap coverage areas, signals sent at the same time from them to tags in the overlap region interfere with each other. This is called the reader collision problem [16]. Fortunately, only the broadcast scheme suffers this problem. Our prototype uses the polling scheme to avoid the problem: According to the polling schemes (or the relay scheme), agents search the queried tag in sequence; signals from readers never interfere.

We will experiment with the broadcast scheme in the future. There are many ways to circumvent the reader collision problem. For example, the DAIT prototype can let each agent delay transmitting its query signal by an amount of time that is a function of its network address. In this way, agents try to avoid transmitting query signals at the same time. This solution is practical and easy to implement and is the solution we plan to implement. Another solution requires each agent to know the network addresses of its neighbors. Each agent can be viewed as a node in a connected graph. There is an edge between two nodes when the agents represented by them have overlapping coverage regions. A graph coloring algorithm can be used to assign different colors to adjacent nodes. The reader collision problem never occurs as long as agents labeled by different colors do not transmit query signals concurrently. This solution is likely to have a better response time than the solution mentioned above or the relay and polling schemes. However it requires additional hardware for each agent to automatically detect its neighbors or connectivity information entered by the user manually. The additional hardware makes agents more costly, and complicated operations by the users make an object locator hard to use.

V. Relative Merits

We use search time and energy consumption of a single query to measure the relative merits of object locator designs. Search time and energy consumption per query depend on many factors including the number of agents, search scheme, search sequence and locations of misplaced objects.

A. Search Time and Energy Consumption

The expressions of energy consumption and search time per query according to broadcast, relay and polling schemes are listed in Table 1. The expressions assume that agents and interrogator(s) are battery powered and communicate in the manners described in Section III. The notations used in the expression are defined in Table 2.

TABLE 1
Expressions for search time and energy consumption

broadcast	E_{total}	$E_{IA} + N_A(x,y,r)E_{Arfid} + E_{AI}$
	T_{avg}	$D_{IA} + pA_1(D_{Arfid} + D_{AI}) + \sum_{i=2}^{n}(\prod_{k=1}^{i-1}1 - pA_k)pA_i(iD_{Arfid} + D_{AI})$
relay	E_{avg}	$E_{IA} + pA_1(E_{Arfid} + E_{AI}) + \sum_{i=2}^{n}(\prod_{k=1}^{i-1}1 - pA_k)pA_i(iE_{Arfid} + (i-1)E_{AA} + E_{AI})$
	T_{avg}	$D_{IA} + pA_1(D_{Arfid} + D_{AI}) + \sum_{i=2}^{n}(\prod_{k=1}^{i-1}1 - pA_k)pA_i(iD_{Arfid} + (i-1)D_{AA} + D_{AI})$
polling	E_{avg}	$E_{IA} + pA_1(E_{Arfid} + E_{AI}) + \sum_{i=2}^{n}(\prod_{k=1}^{i-1}1 - pA_k)pA_i(iE_{Arfid} + (i-1)E_{IA} + iE_{AI})$
	T_{avg}	$D_{IA} + pA_1(D_{Arfid} + D_{AI}) + \sum_{i=2}^{n}(\prod_{k=1}^{i-1}1 - pA_k)pA_i(iD_{Arfid} + (i-1)D_{IA} + iD_{AI})$

TABLE 2 Notations

• D_{IA}:	Delay of a message transmission from an interrogator to an agent
• E_{IA}:	Energy consumption of a message transmission from an interrogator to an agent
• D_{AA}:	Delay of a message transmission from one agent to another
• E_{AA}:	Energy consumption of a message transmission from one agent to another
• D_{AI}:	Delay of a message transmission from an agent to an interrogator
• E_{AI}:	Energy consumption of a message transmission from an agent to an interrogator
• D_{Arfid}:	Time for an agent to use its RFID reader to search a queried tag
• E_{Arfid}:	Energy consumption for an agent to use its RFID reader to search a queried tag

The total energy consumed by the object locator for processing a Query operation according to the broadcast scheme is the sum of the three terms in the first row of Table 1. In this case, the interrogator transmits only one query message when processing a Query operation. The energy it consumes is E_{IA}. The energy consumed by each agent in the search is E_{Arfid}. The total energy consumed by all agents is $N_A(x, y, r)E_{Arfid}$, where $N_A(x, y, r)$ is the number of agents with range r in a rectangular space of dimensions x and y. The agent finding the queried tag consumes E_{AI} to send a response back to the interrogator.

In the expressions, pA_i denotes the probability that the i-th agent in the search sequence finds the queried tag. In general, this probability is a function of the number and location distribution of objects (i.e., tags) in the house. (To keep the expressions simple, our notations do not show this dependency.) The expression of the expected time taken by the locator using the broadcast scheme to respond to a Query operation assumes that agents search the queried tag in sequence in order to avoid the reader collision problem. The first term in the expression is the time taken by the query message from the interrogator to reach all the agents. If the first agent finds the queried tag, which occurs with probability pA_1, the addition delay is $D_{Arfid} + D_{AI}$. This is the reason for the second term in the expression of T_{avg}. In general, the probability that the queried tag is found by the i-th agent is $\prod_{k=1}^{i-1}(1 - pA_k)\,pA_i$. When this occurs, each of the i agents spends D_{Arfid} amount of time to search for the queried tag before the i-th agent can respond to the interrogator. Hence, the delay is $iD_{Arfid} + D_{AI}$.

The average search time of an object locator that uses the relay and polling scheme are estimated by the expressions in the fourth and sixth rows, respectively. Relay and polling scheme also lets all agents search the queried tag in sequence. This is why the coefficients in these expressions are the same as the coefficients in the expression of T_{avg} for the broadcast scheme. The expressions of the average energy consumption can be derived from the expressions of the average search time by substituting energy consumption for message transmission delay because sending a message cause both transmission delay and energy consumption.

As stated earlier, Table 1 is based on the assumption that agents, like the interrogator, are battery powered. Hence, the expressions for energy consumption listed in the table include energy consumption of both agents and an interrogator. However, agents can be connected to wall plugs, especially when the number of agents is smaller, as in the case of RAIT locators. The interrogator using relay and broadcast scheme consumes exactly E_{IA} to search a queried tag. The interrogator using polling scheme consumes at least E_{IA} to search a queried tag. Thus, the polling scheme is suitable for stationary interrogator(s) and the relay and broadcast scheme are suitable for portable interrogator(s) if we do not need to account for the energy consumption of agents.

B. Model of Object Locality

The probability pA_i of that an agent A_i finds the queried tag, and hence the misplaced object, depends on where the object is at the time. To calculate this probability, we use a locality model of tracked objects. The model gives the spatial probability density of the locations of each object. For the sake of simplicity and without noticeable lose of accuracy, we partitions the space in the search area into unit squares, rather than treating the coordinates of a location as continuous

variables. (Except for where it is stated otherwise, the dimension of a unit square is 1 cm by 1 cm.) This allows us to model a house as a finite, discrete and planar search space. We denote the space by $Z = \{Z_{x,y}\} \subseteq N \times N$. Each element $Z_{x,y}$ of the space is a unit square; its location is given by the coordinate (x, y) where both x and y are integers. All agents are at fixed and known locations. A misplaced object may be placed anywhere within the search space.

We call the probability of finding a queried object at $Z_{x,y}$ the (*existence*) *probability* of the object at $Z_{x,y}$. (For example, if we find an object at $Z_{x,y}$ on the average 10 times in 100 searches for the object, the (existence) probability of the object at $Z_{x,y}$ is approximately 0.10. We use $pZ_{x,y}(j)$ to denote the existence probability of an object with a tag of id = j at $Z_{x,y}$. We ignore the probability of a registered object being outside the search space when the object is being searched. (In other words, we do not consider the situation where someone has taken some registered object shopping, for example, while someone else is searching for it in the house.) For every object, the sum of probabilities of it at all locations in the search space equals to 1.

Fig. 11 gives an illustrative example. The figure is not drawn in scale, and each unit square in this example is 10 cm by 10 cm in dimension. Two agents A_1 and A_2 are at their locations. The id of A_1 is 1 and the id of A_2 is 2. The rectangle models a desk. It contains 15 unit squares. The number in each square gives the probability of a queried object being at the location. Since the numbers add up to 1, they tell us that the object is surely some where in the rectangle. We want to calculate pA_i, the probability that the agent with id = i can find the queried tag. Using Fig. 11 for example, we see that pA_1 equals to the sum of all existence probabilities within the read range of the agent A_1; in other words, pA_1 is about 0.87. Similarly, we find that pA_2 is about 0.68.

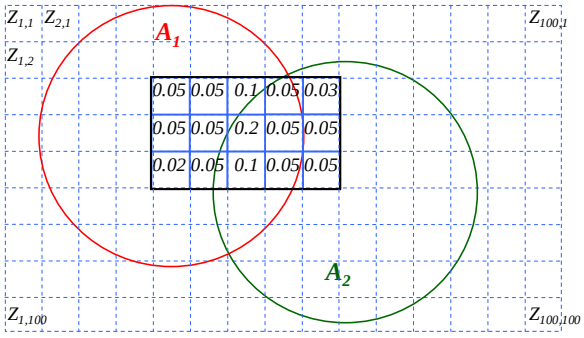


Fig. 11 The locality model

We call the area where a misplaced object might be placed an *object region*. The size of an object region is the total area of the region in number of unit squares. We characterize the locality of a misplaced object by the size and shape of its object region and its existence probabilities of being at each unit square within the region. Once we know the locality parameters of an object and coverage area of each agent A_i, the terms pA_i can easily be calculated. We can then calculate the average search time and energy consumption of the object based on the probability pA_i for all agents.

C. Evaluation Environment and Results

The environment we used to evaluate the relative performance of our designs has a 10m by 10m search space, containing 1000 × 1000 unit squares of size 1 cm by 1 cm. Agents are placed according to the arrangement in Fig. 12(a). The number of agents is $N_A(1000, 1000, r)$. Again, r is the read range of an agent. The ranges of desk-level and room-level agents are 100 and 350, respectively, the typical number of room-level agents in a RAIT locator is $N_A(1000,1000,350) = 6$, and the typical number of agents in a DAIT locator is equal to $N_A(1000,1000,100) = 42$.

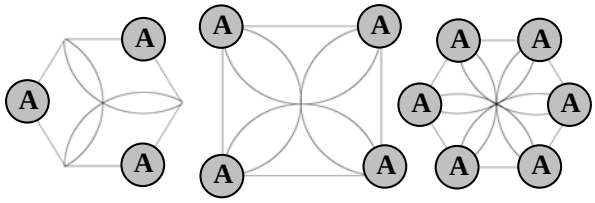


Fig. 12 The possible arrangement of agents

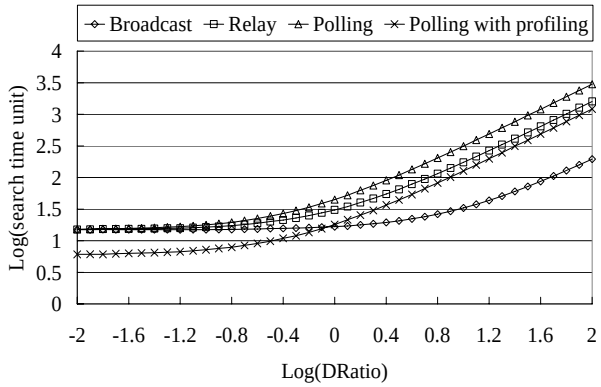
In Section III, we said that the agent with the smallest network address is the first agent and the other agents are asked to search for the object one by one in order of agent ids. We call this search order *sequential*. Alternatively, we can ask the agents in non-increasing order of their empirical existence probabilities. This search sequence is called *profiling*.

Our evaluation program assumes that object regions are circular for the sake of simplicity. The center and radius of an object region are randomly generated. The variables D_{IA}, D_{AA} and D_{AI} in Table 2 have the same values because both interrogators and agents use the same kind of RF transceiver. For the same reason, E_{IA}, E_{AA} and E_{AI} have the same value. For convenience, we use D_{Arfid} and E_{Arfid} as base units of delay and energy consumption. The ratio of D_{IA}/D_{Arfid} (D_{AI}/D_{Arfid} and D_{AA}/D_{Arfid}) is called *DRatio* and the ratio of E_{IA}/E_{Arfid} is called *ERatio*. The evaluation program needs only these two parameters rather than all variables.

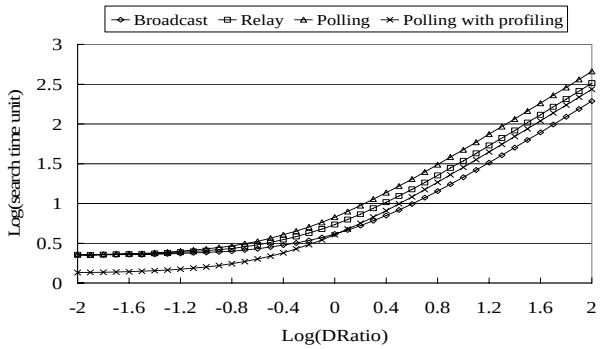
Fig. 13(a) and (b) show the average search time for broadcast scheme, relay scheme, and polling scheme (i.e., polling in sequential order), as well as polling scheme with profiling. The search time of relay and polling schemes is higher than broadcast scheme for all values of DRatio. The search time of polling scheme with profiling is less than that of broadcast scheme when DRatio is less than about 1 (10^0) for N_A = 42 and 1.25 ($10^{0.1}$) for N_A = 6.

Fig. 14 shows the average energy consumption consumed by agents when N_A is 42 and 6. The energy consumption consumed by agents is the same, when the relay and polling scheme is used. As Fig. 14(a) depicts, the energy consumption of relay scheme, polling scheme and polling scheme with profiling is less than that of broadcast scheme

when ERatio is less than 1.99 ($10^{0.3}$) and 7.94 ($10^{0.9}$), respectively. ERatio at the intersections of the curves in Fig. 14(b) are about 3.16 ($10^{0.5}$) and 15.85($10^{1.2}$).

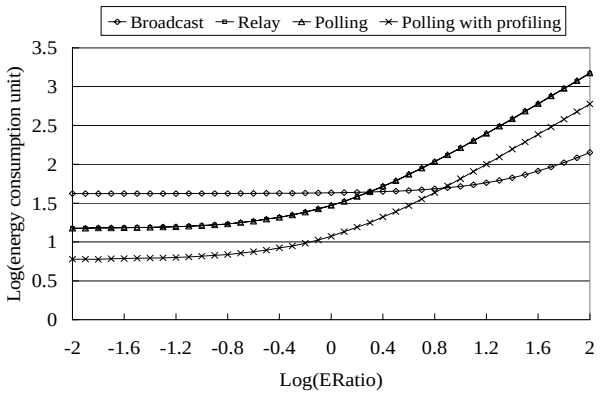


(a) N_A = 42

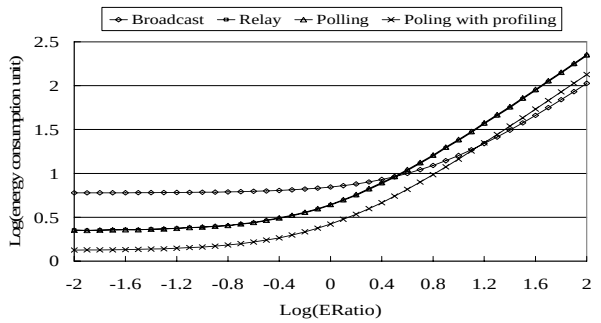


(b) N_A = 6

Fig. 13 Search time Vs DRatio



(a) N_A = 42



(b) N_A = 6

Fig. 14 Energy consumption of agents Vs ERatio

Table 3 gives a summary. The table suggests the broadcast scheme when DRatio is high and search time is more important than energy consumption. When DRatio is low, the differences among the search times of all search schemes are small. Energy consumption becomes the dominant factor for comparison. It is possible for agents in a RAIT locator to connect to power source. For energy saving on interrogators, we suggest polling scheme with profiling for stationary interrogators and relay or broadcast scheme for portable interrogators. As for DAIT locators, we consider energy consumption of an interrogator and agents. We suggest polling scheme with profiling when ERatio is low and the same suggestions as that for a RAIT locator if ERatio is high.

TABLE 3 Summary of results

	Low DRatio Low ERatio	Low DRatio High ERatio	High DRatio Low ERatio	High DRatio High ERatio
RAIT locator (fewer agents)	PI: ***b*** or ***r*** SI: ***pp***	PI: ***b*** or ***r*** SI: ***pp***	***broadcast***	***broadcast***
DAIT locator (more agents)	***pp***	PI: ***b*** or ***r*** SI: ***pp***	***broadcast***	***broadcast***

PI: portable interrogator
SI: stationary interrogator

b: broadcast ***r***: relay
pp: polling with profiling

VI. CONCLUSION

We described here three alternative RFID-based object locator designs. RFID-based object locators are extensible, reusable and low maintenance. They are very easy for users to use and set up. Our analysis shows that search time and energy consumption for all designs and search schemes depend the capabilities of RFID readers and RF transceivers used by agents. Roughly speaking, polling and relay scheme is competitive to broadcast scheme only when DRatio or ERatio are less than 10.

We implemented a proof-of-concept DAIT prototype object locator to demonstrate the object locator concept and designs. The prototype uses only readily available hardware components, including readers and tags with directional antennae. The performance of the prototype is far from ideal, primarily for this reason. Because it is impossible to control the orientation of tag antennae, omni-directional antennae are better suited for our application.

The total cost of an object locator depends on many factors. The total hardware cost of a minimum object locator is the sum of the costs of an interrogator and required number of agents and tags. Compared with the costs of interrogator and agent, the hardware cost of tags is significantly lower and, for the discussion here, can be neglected.

Currently, the total hardware cost of an object locator is dominated by the total cost of agents, and the cost of an agent is dominated by the RFID reader in the agent. The number of agents required to fully cover a house depends on dimensions *x* and *y* of the house, the read range of the agents and the way agents are placed. To get a rough estimate, we assume that the

coverage area of each agent is a circle. Fig. 12 depicts three ways to place agents. Putting agents further apart than locations shown in Fig. 12(a) can create blind regions. Putting more agents closer than those indicated in Fig. 12(c) is not necessary since the space is covered by at least two agents. We need six room-level agents to cover a 10m x 10m space even when we place agents as shown in Fig. 12(a) (i.e., as far as possible without creating blind regions). The existing object locator costs $ 50 US. A RAIT locator is not competitive to the existing locator unless the cost per room-level agent is about $ 10 US. As for DAIT locator, the cost per desk-level agent must be much lower. We are optimistic that the cost of agents will become sufficiently lower in the coming decade as the need for more and more products (e.g., Smart pantry [1], dispenser in [4]) containing RFID readers are developed to take advantage of this technology [17].

ACKNOWLEDGMENT

This work is partially supported by the Taiwan Academia Sinica thematic project SISARL (http://www.sisarl.org).

REFERENCES

[1] C. F. Hsu, H. Y. Liao, P. C. Hsiu, Y. S. Lin, C. S. Shih, T. W. Kuo and Jane W. S. Liu, "Smart Pantries for Homes," In Proc. IEEE SMC, October 2006

[2] H. C. Yeh, P. C. Hsiu, P. H. Tsai, C. S. Shih and Jane W. S. Liu, "APAMAT: A Prescription Algebra for Medication Authoring Tool," In Proc. IEEE SMC, October 2006

[3] Y. T. Hsu, S. F. Hsiao, C. E. Chiang, Y. H. Chien, H. W. Tseng, A. C. Pang, T.W. Kuo and K. H. Chiang "Walker's Buddy: An Ultrasonic Dangerous Terrain Detection System," In Proc. IEEE SMC, October 2006

[4] P. H. Tsai, H. C. Yeh, C. Y. Yu, P. C. Hsiu, C. S. Shih and Jane W. S. Liu, "Compliance Enforcement of Temporal and Dosage Constrains," In Proc. IEEE RTSS, December 2006

[5] http://www.sharperimage.com/us/en/catalog/product/sku__SI676FUN "Now You Can Find It!"

[6] http://www.epill.com/ e-pill Medication Reminders

[7] Getting I. "The Global Positioning System," IEEE Spectrum 30, 12 (December 1993), 36–47.

[8] Want, R., Hopper, A., Falcao, V. and Gibbons J., "The Active Badge Location System," ACM Transactions on Information Systems 10, 1 (January 1992), 91–102.

[9] Harter A. and Hopper A., "A New Location Technique for the Active Office," IEEE Personal Communications 4, 5(October 1997), 43–47.

[10] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward and Paul Webster, "The Anatomy of a Context-Aware Application," In Proc. ACM MOBICOM, August 1999

[11] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan, "The Cricket Location-Support System," In Proc. ACM MOBICOM, August 2000

[12] Bahl P., and Padmanabhan V., "RADAR: An In-Building RF-based User Location and Tracking System," In Proc. IEEE INFOCOM, March 2000

[13] Konrad Lorincz and Matt Welsh, "MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking," In Proceedings of the International Workshop on Location and Context-Awareness at Pervasive, May 2005

[14] J. Hightower, G. Borriello, and R.Want, "SpotON: An indoor 3-D location sensing technology based on RF signal strength," Univ.Washington, Seattle, Tech. Rep. 2000-02-02, February 2000

[15] P. Castro, P. Chiu, T. Rremenek, and R. R. Muntz, "A probabilistic room location service for wireless networked environments", In Proc. ACM UBICOMP, 2001

[16] Leong Kin Seong, Ng Mun Leng and Cole Peter H., "The Reader Collision Problem in RFID Systems", Auto-ID Labs, August 2005

[17] http://www.abiresearch.com/products/market_research/RFID_Readers "The Market for RFID Readers", ABI research