Design and Implementation of RFID-Based Object Locators

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

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/53576

1. 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 devices and services (e.g., [1-8]) that can help elderly individuals to live well and independently. *Object locator* is such a device. The device can assist its users in finding misplaced household and personal objects in a home or office. Figure 1 shows several object locators offered today by specialty stores and websites. Each of these locators contains an interrogator with a few buttons and an equal number of tags: Even the largest one, the leftmost one in the figure, offers only 8 buttons. The buttons are of different colors, and there is 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. Other locators work similarly.

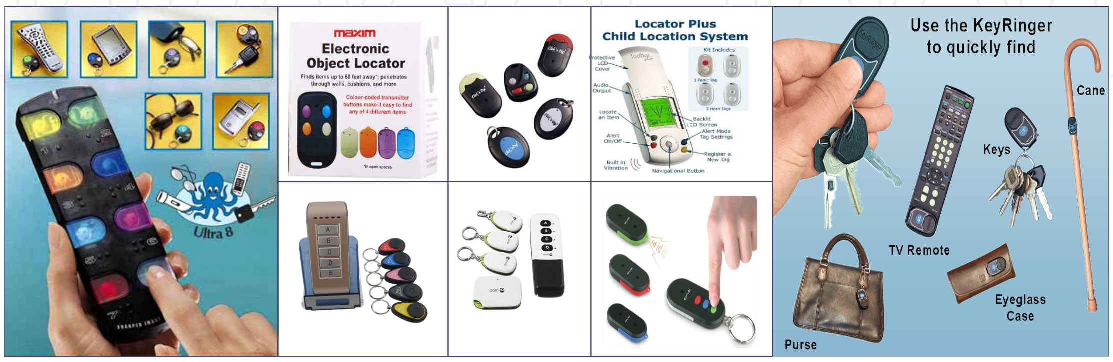


Figure 1. Existing object locators

Existing object locators are not ideal in many aspects: The number of buttons on the interrogator and tags is fixed, and the number is small. Extending the locator to track more objects is impossible. – If the user were to use more than one tag of the same color, the tags would all respond to the search signal for tag(s) of the color from the interrogator. This situation is clearly not desirable. – When a tag breaks, the user must purchase a replacement tag of the same color as the broken one. Tags are battery-powered. A tag might become a lost object itself after it runs out of battery. More seriously, the interrogator itself can be misplaced. Obviously, these are serious shortcomings.

This chapter describes three designs and a proof-of-concept prototype of object locators based on the *RFID (Radio Frequency Identification)* technology. RFID-based object locators do not have the drawbacks of existing object locators. 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). A mobile interrogator can be tagged and thus, can be searched via other interrogators when it is misplaced. 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 RFID tags without batteries; the user is never burdened by the concern that a tag may be out of battery.

This chapter makes two contributions: The first is the object locator designs presented here. The designs use different hardware components and have different hardware-dependent software requirements. The information provided by the chapter on these aspects should enable a developer to build a suitable object locator platform, or an extension to one of the commonly used computer and smart mobile device platforms. The functionality of hardware-independent object locator software is well defined, and a C-like pseudo code description can be found in [9].

The hardware capabilities and object search schemes used by the designs lead to differences in search time and energy consumption. We provide here a numeric model that can be used to determine the tradeoffs between these figures of merit. Developers of RFID-based object locators can use the results of the analysis as design guides. Today, object locators based on all designs are too costly: Typical RFID readers have capabilities not needed by our application and cost far more than what is suitable for the application. Through this analysis, we identify the design that is the most practical for the current state of RFID technology and project the advances in the technology required to make RFID-based object locators affordable (i.e., with prices comparable with some of the locators one can now find in stores.) This is the second contribution of the chapter.

The rest of this chapter is organized as follows. Section 2 describes closely related works. Section 3 describes use scenarios that illustrate how a RFID-based object locator may be used. Section 4 presents three designs of RFID-based object locators. Section 5 describes the implementation of a proof-of-concept prototype based on one of the designs. It also describes the reader collision problem [10] encountered in the prototype and the solution we use to deal with the problem. Section 6 describes a numeric model for computing energy consumption and search time and compares the merits of the designs. Section 7 concludes the chapter and discusses future works.

2. Background and related work

This section first presents a brief overview of RFID technology as a way to state the assumptions made in subsequent chapters on state-of-the-art readers and tags. Our object locator resembles location detection systems in its goal: assisting users to locate objects. The section describes existing location systems and compare and contrast them with our object locators.

2.1. RFID technology

RFID technology is now applied to a wide spectrum of applications. As an example, personal identification application is used to provide authentication and authorization to individuals carrying their RFID tags so that they can be automatically identified by a central computer. Card-like RFID tags used as smart cards in public transports is another example: Information on money stored in a tag is automatically deducted when the card holder presents the card in front of a reader while getting on or off a transporter. Other applications include using RFID tags as markings of books for more efficient library management, shipping containers for tracking them by retail industry, and so on.

Figure 2 shows a typical system that uses RFID technology. The host machine uses one or more RFID readers to retrieve digital information stored in RFID tags and processes the information according to the needs of one or more applications. In general, a RFID tag contains a globally unique identification (UID) as well as data fields organized in a standard way [11]. A RFID-based object locator only needs the UID information; other data fields are not used.

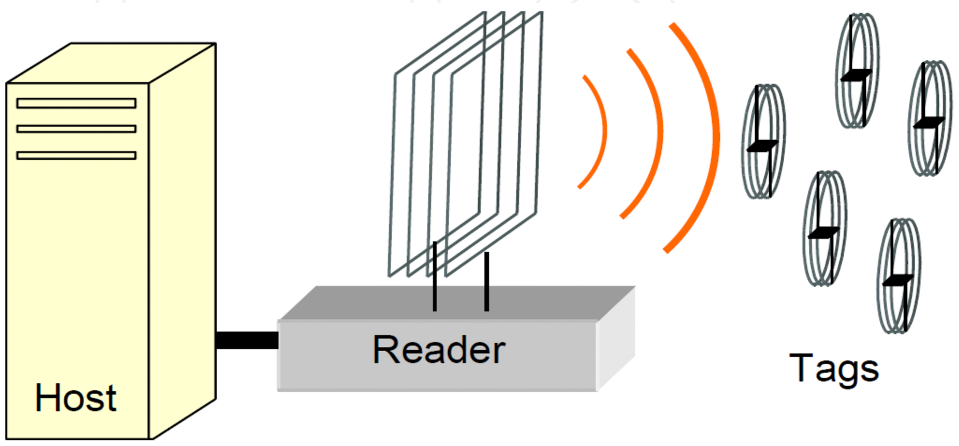


Figure 2. A configuration of RFID system

There are three types of RFID tags: passive, semi-passive and active. A *passive tag* has no internal power source: It gets the power it needs to operate from the incident RF signal radiated by a reader. The readable distance of such a tag ranges from 10 cm to a few meters depending on the frequency of the incident RF signal and its antenna design. In contrast, *semi-passive* and *active tags* have internal power source. Semi-passive tags can increase their readable distances by leveraging internal power. Like passive tags, semi-passive tags respond only after receiving some command from the reader. An active tag, on the other hand, can send RF signals to a reader even when it is not commanded by the reader. Being battery free and having long lifetime (in tens of years) are the major advantages of passive tags over other types of tags for our application.

Each message sent from a reader to tags contains a command code. Among the sets of commands defined by ISO15693 [12], our object locators use only *mandatory* commands and *custom* commands. Standard-compliant tags support all commands in the mandatory set. Commands in the custom set are defined by tag IC manufacturer according to application needs.

The command used to read UID of a tag is the *inventory* command in the mandatory set. This command has only the non-addressed mode, while the other commands have both non-addressed and addressed modes. A command in the *non-addressed mode* is processed by all tags which receive it. A command in the *addressed mode* consists of the command code followed by a UID. When a tag receives an addressed-mode command, it first checks whether the UID is its own. The tag processes and responds to the command only when it is the tag addressed by the UID.

2.2. Location detection systems

Many different location detection systems are available today. Global Positioning System (GPS) [13] 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.

Active Badge [14] is representative of infrared-based location detection systems. A badge containing an infrared transmitter is attached to each object to be tracked by the system. The transmitter sends periodically messages containing the unique identification of the badge. The messages are caught by some infrared receivers at fixed known locations and relayed by the receivers to a central computer. The central computer resolves the position of the badge based on the locations of the receivers. Shortcomings of systems such as Active Badge arise from the fact that infrared signals cannot penetrate most materials in a building and are easily interfered by other infrared sources.

Ultrasound is used to assist with distance measurement in Bat [15] and Cricket [16]. These systems use both ultrasound and RF signals to measure distances between beacons (transmitters) and listeners (receivers): When a beacon at a known location transmits an ultrasound signal and a RF signal concurrently, a listener can calculate the distance to the beacon from the difference between the arrival times of the signals.

Many indoor location detection systems use RF-based technology to take advantage of the fact that RF signals penetrate most non-metallic materials. RADAR [17] is an example. The system estimates distance by estimating the strength of RF signals. Specifically, the system measures in the initialization phase at a set of fixed locations the strengths of a RF signal sent by a location-known transmitter. The measured strengths are stored in a database to be used later as yardsticks during the working phase. In the working phase, each receiver measures the strength of a RF signal transmitted from a tracked object and sends the strength to a central computer. The computer compares the measured strengths with the information stored in the database and then resolves the possible position of the transmitter (i.e., the tracked object). MoteTrack [18], similar to RADAR, uses empirical distance measurement to estimate positions of objects. WLAN (wireless local area network) can be used to build location detection systems also. SpotON [19] and Nibble [20] are examples.

Compared with the above mentioned location detection systems, an object locator must be a far more low 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 a low cost location-aware service. An object with a receiver can determine 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.

3. 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. As illustrated by Figure 3, the interrogator resembles a smart phone. 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 what agents are and do. Unlike common smart phones, however, the interrogator has a RFID reader. The reader is used for the Add operation described below.

Specifically, Figure 3 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 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.

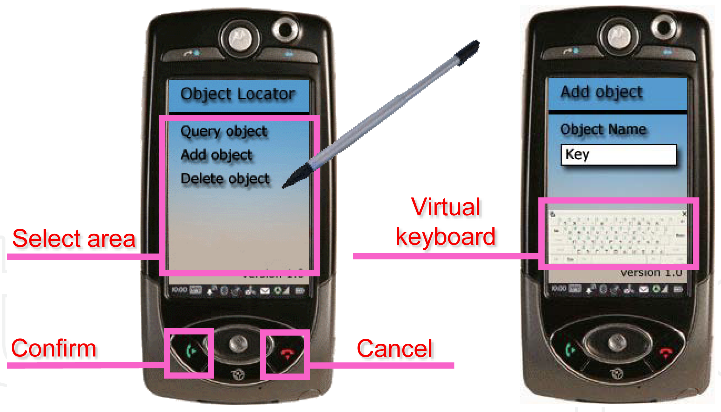


Figure 3. Object locator user interface

Figures 4 and 5 illustrate Add and Query operations, respectively. 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. Figure 4 shows a scenario: The user picks an unused tag and attaches it to an object to be tracked as shown in Figure 4(a) and (b). Then, the user puts the tag close to the interrogator and selects Add object. This step is shown in Figure 4(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 Figure 4(d). The user repeats the above steps to register each object until all objects to be tracked are registered.

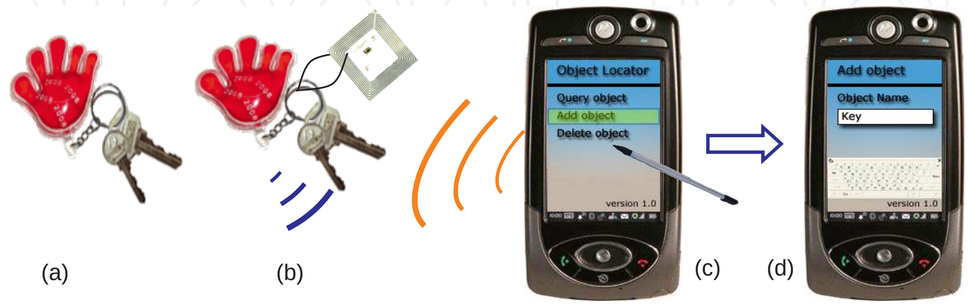


Figure 4. Add operation

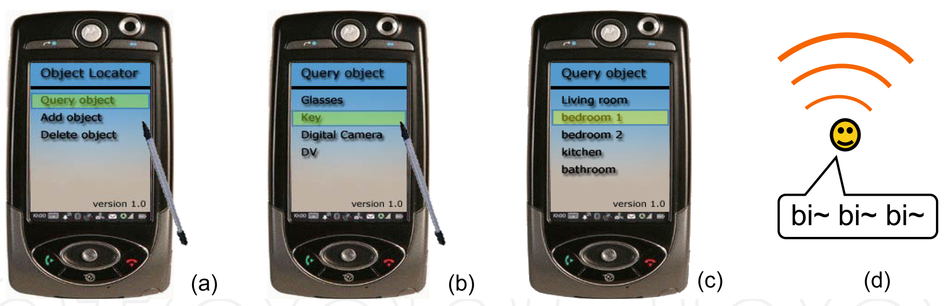


Figure 5. Query operation

Query operation is the work horse of the object locator. The user presses Query object on the touch screen, as illustrated by Figure 5(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 Figure 5(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 describe the search process in the next section.

Object locators of different designs present the result of Query operation in different ways. As examples, Figure 5(c) and (d) shows two different responses. In Figure 5(c), the interrogator directs the user to the place (e.g. bedroom 1) where the queried object is found. In Figure 5(d), 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 I.

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.

4. 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 sufficiently 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. The RFID reader in the agent enables the agent to search for the tags within its coverage area. As stated in Section III, 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 (WLAN). 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. The interrogator requests assistance from an agent by sending the TID of the queried tag to the agent via the WLAN. We assume that the network provides reliable communication. We do not mention other aspects of the WLAN because they are not relevant to our discussion.

4.1. 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.

RAIT locator uses one or more agents to cover each room, and the house is fully covered by agents as shown in Figure 6. When the user invokes a 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, in addition to 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 illustrated by Figure 5(c), telling the user to go to the specified room where the queried object has been found.

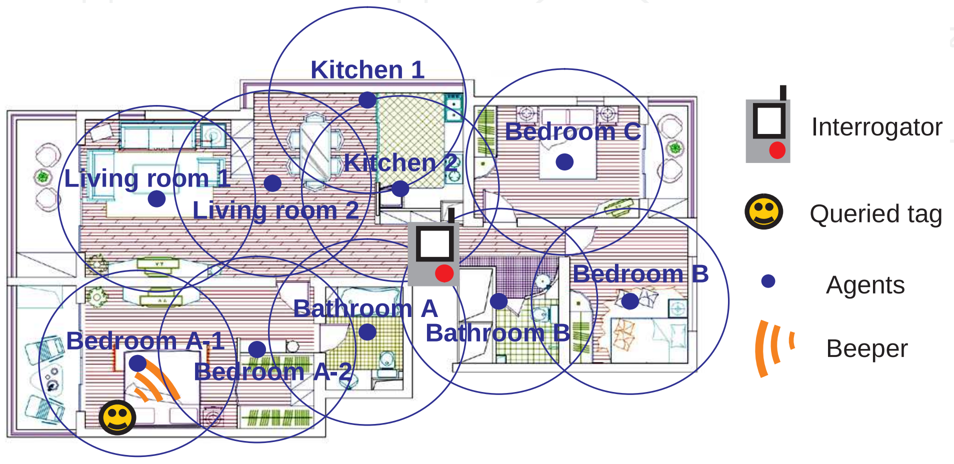
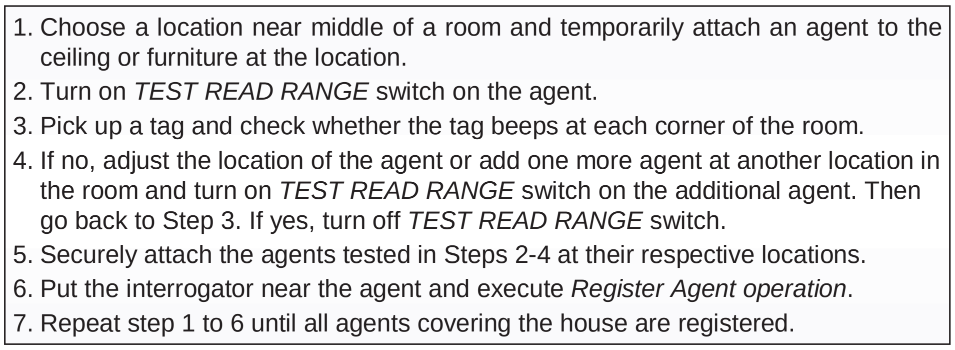


Figure 6. Configuration of RAIT locator

Obviously, the agents must be set up before a RAIT locator can be used. Figure 7 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 likely to be 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 in Step 3.



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 Steps 2-4 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.

Figure 7. Agent set-up process

The Register Agent operation in Step 6 is similar to Add operation described in Section 3. 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 Figure 5(c). 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 WLAN and later compute the addresses of other agents by adding or substituting some number from its own address.

Figure 8 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. Data field allows interrogators to synchronize their databases created by Add and Register Agent operations. We will discuss how the other fields are used shortly.

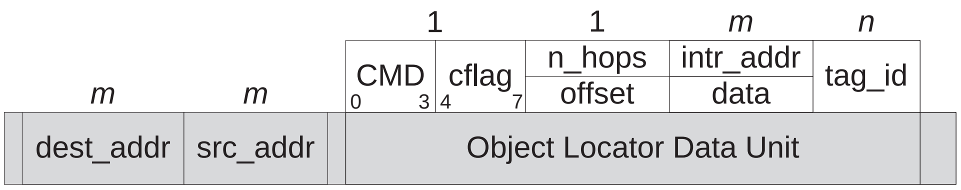


Figure 8. RAIT locator message format

4.2. DAIT and DRAIT locators

DAIT locator, shown in Figure 9, is an extension of RAIT locator. The designs are similar in how the Query operation is handled by the interrogator and agents. 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.

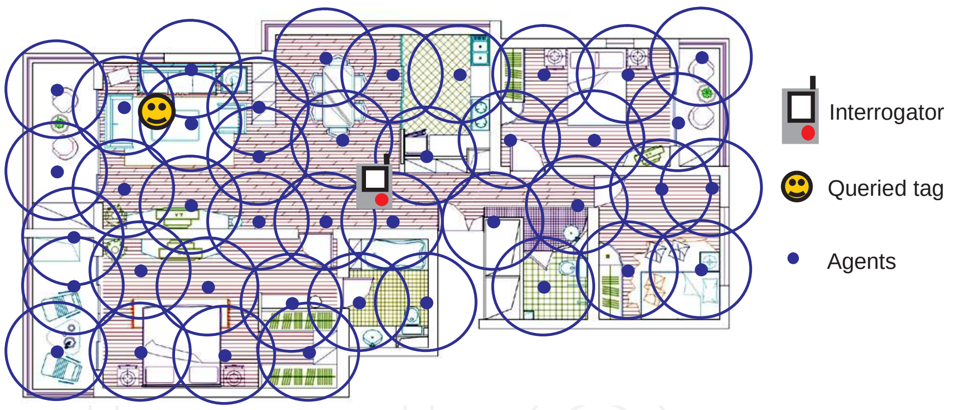


Figure 9. Configuration of DAIT locator

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 has a hybrid design that aims to extend the lifetime of semi-passive tags. A DRAIT locator contains both room-level and desk-level agents. Its 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 desk-level agents on furniture in addition to setting up room-level agents as described above. Because misplaced objects are often on furniture or in vicinities of them, the queried object can often be found by a desk-level agent, and the tag on it does not need to beep.

4.3. Search schemes

A queried object can be searched 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.

5. Prototype implementation

We implemented a proof-of-concept prototype of DAIT locator, the design that does not require customized semi-passive tags. Indeed, all components used in our prototype are readily available today. Parts (a) and (b) of Figure 10 show 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.

The lack of customized antenna design for tags and readers and the reader collision problem seriously affects the performance of our prototype. Our DAIT prototype uses only tags with directional antennae. (Again, the reason is that such tags are readily available.) When the antennae of tags and readers are directional, the read performance of agents depends on the orientation of the antennae. Clearly, tagged objects may be placed in arbitrary orientations. As a consequence, it is impossible to ensure optimal or near optimal alignment of the tag antennae towards the agents covering their locations. This is the reason that tags in a DAIT object locator should have omni-directional antennae. Agents with omni-directional antennae can be simply set on furniture as shown in Figure 11(a). Agents with directional antennae should be attached to the ceiling as shown in Figure 11(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.

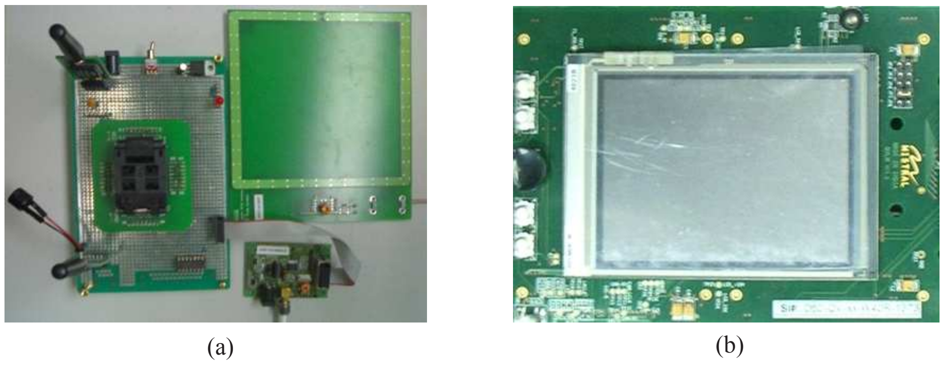


Figure 10. Agent and interrogator

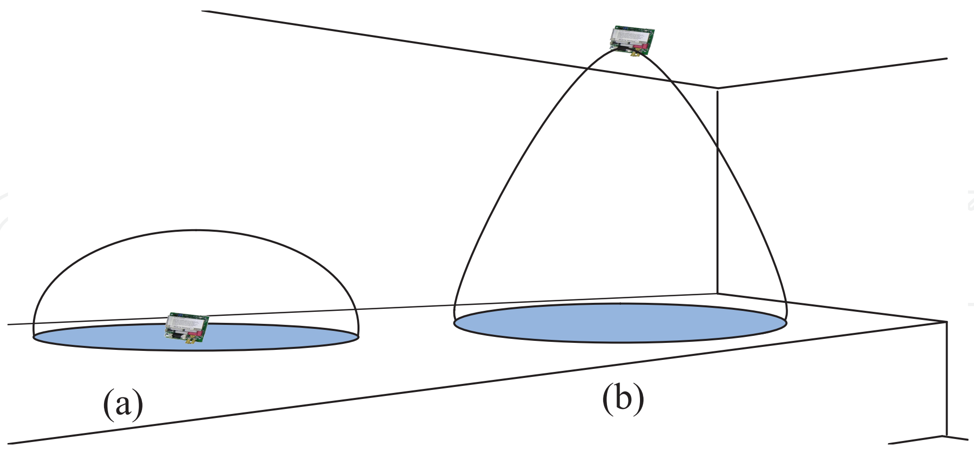


Figure 11. Arrangement of agents

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* [10]. 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.

A DAIT locator that uses the broadcast scheme can circumvent the reader collision problem in many ways. 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.

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 user make an object locator hard to use.

6. Relative merits

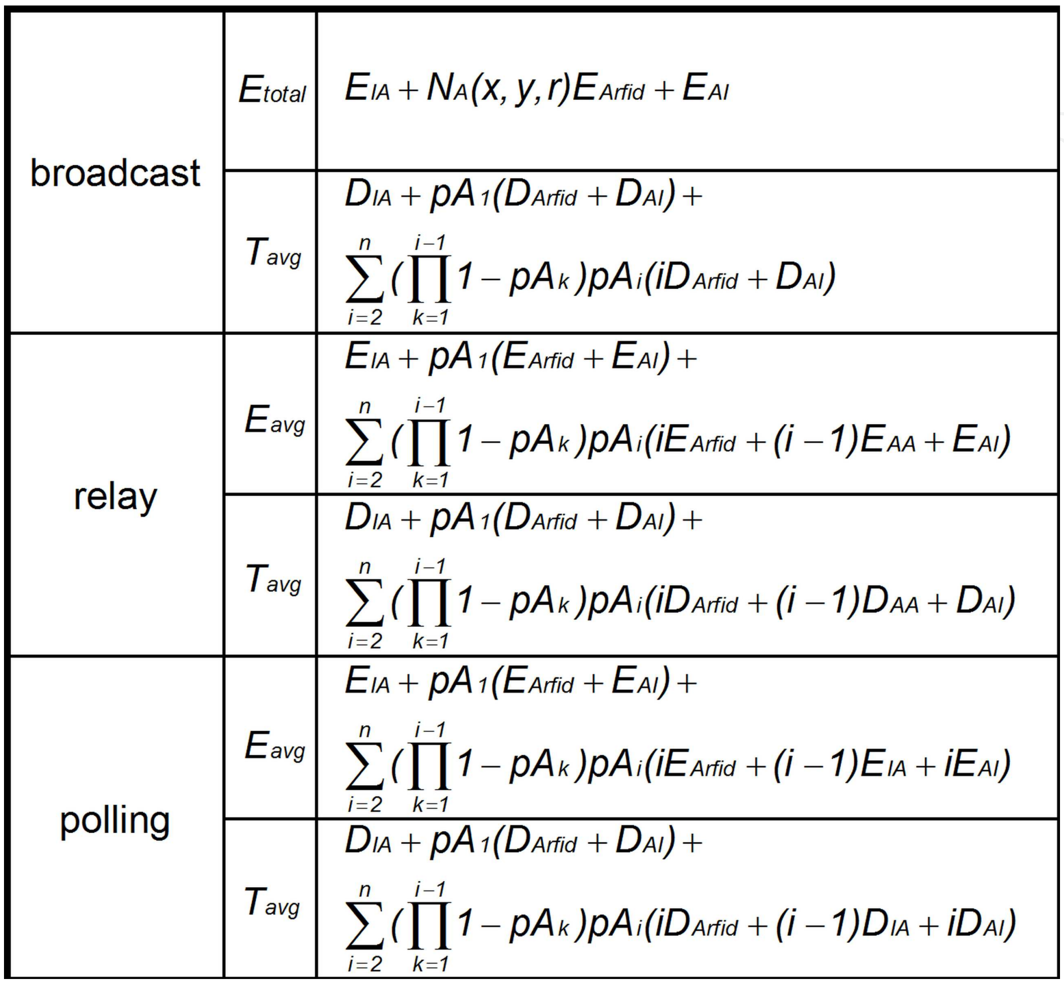
We use search time and energy consumption of a single query to measure the relative merits of object locator designs. Search time and energy consumption per query depend on many factors including the number of agents, search scheme, search sequence and locations of misplaced objects.

6.1. 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 4. The notations used in the expression are defined in Table 2.

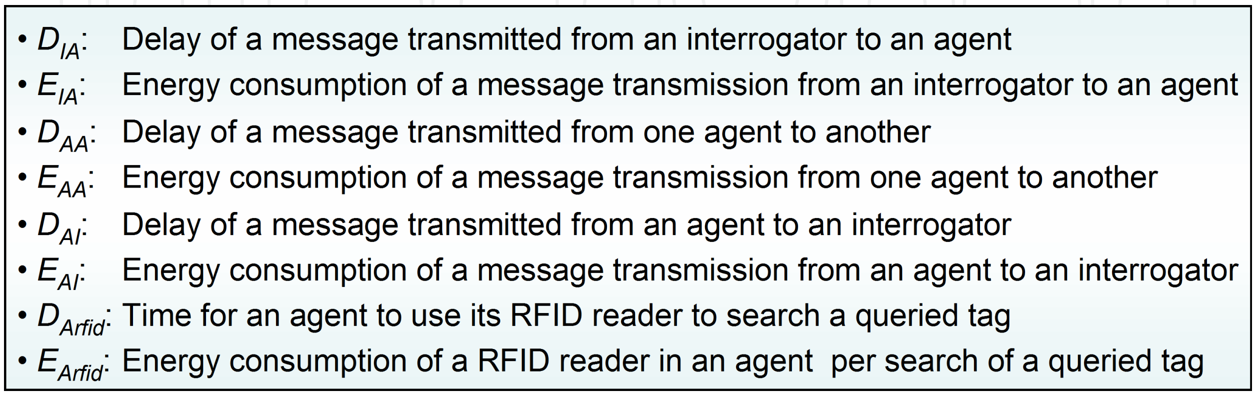
The total energy consumed by the object locator for processing a Query operation according to the broadcast scheme is the sum of the three terms in the first row of Table 1. In this case, the interrogator transmits only one query message per Query operation. The energy it consumes is E_{IA}. The energy 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.)



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 1. Expressions for search time and energy consumption



• D_{IA}:	Delay of a message transmitted 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 transmitted from one agent to another
• E_{AA}:	Energy consumption of a message transmission from one agent to another
• D_{AI}:	Delay of a message transmitted 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 of a RFID reader in an agent per search of a queried tag

Table 2. Notations

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 other 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 in Table 1, 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 and the interrogator are battery powered. Hence, the total energy consumption includes energy consumptions of agents and an interrogator. However, agents can be connected to wall plugs, especially when the number of agents is small, 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. Therefore, 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.

6.2. 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 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. Hence, for every object being searched, the sum of the probabilities of it being at all locations in the search space equals to 1.

Figure 12 gives an illustrative example. The figure is not drawn 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 somewhere in the rectangle. We want to calculate pA_i, the probability that the agent with id = i can find the queried tag. Using Figure 12 as an illustrative 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.

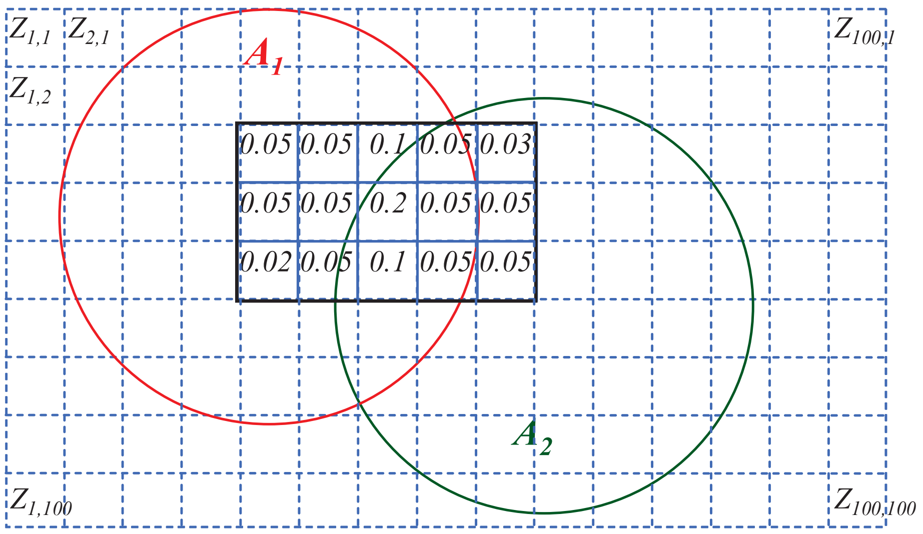


Figure 12. Locality model

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.

6.3. 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 Figure 13(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 $N_A(1000,1000,100) = 42$.

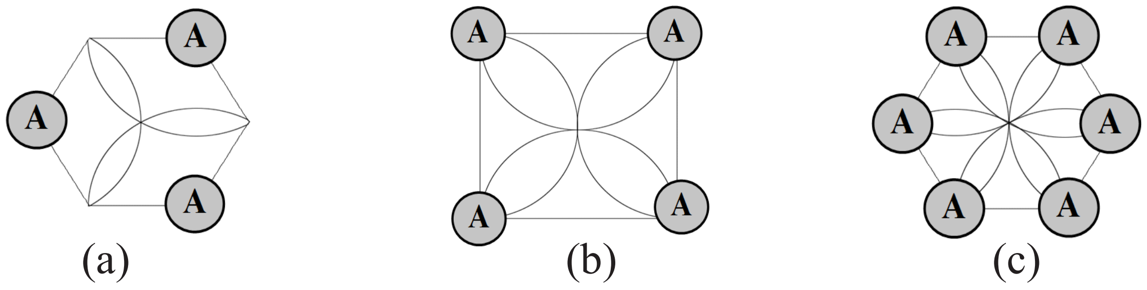


Figure 13. Possible arrangement of agents

In Section 4, we said that the agent with the smallest network address is the first agent and the other agents are asked one by one in order of agent ids to search for the queried object. 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.

Figure 14(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.

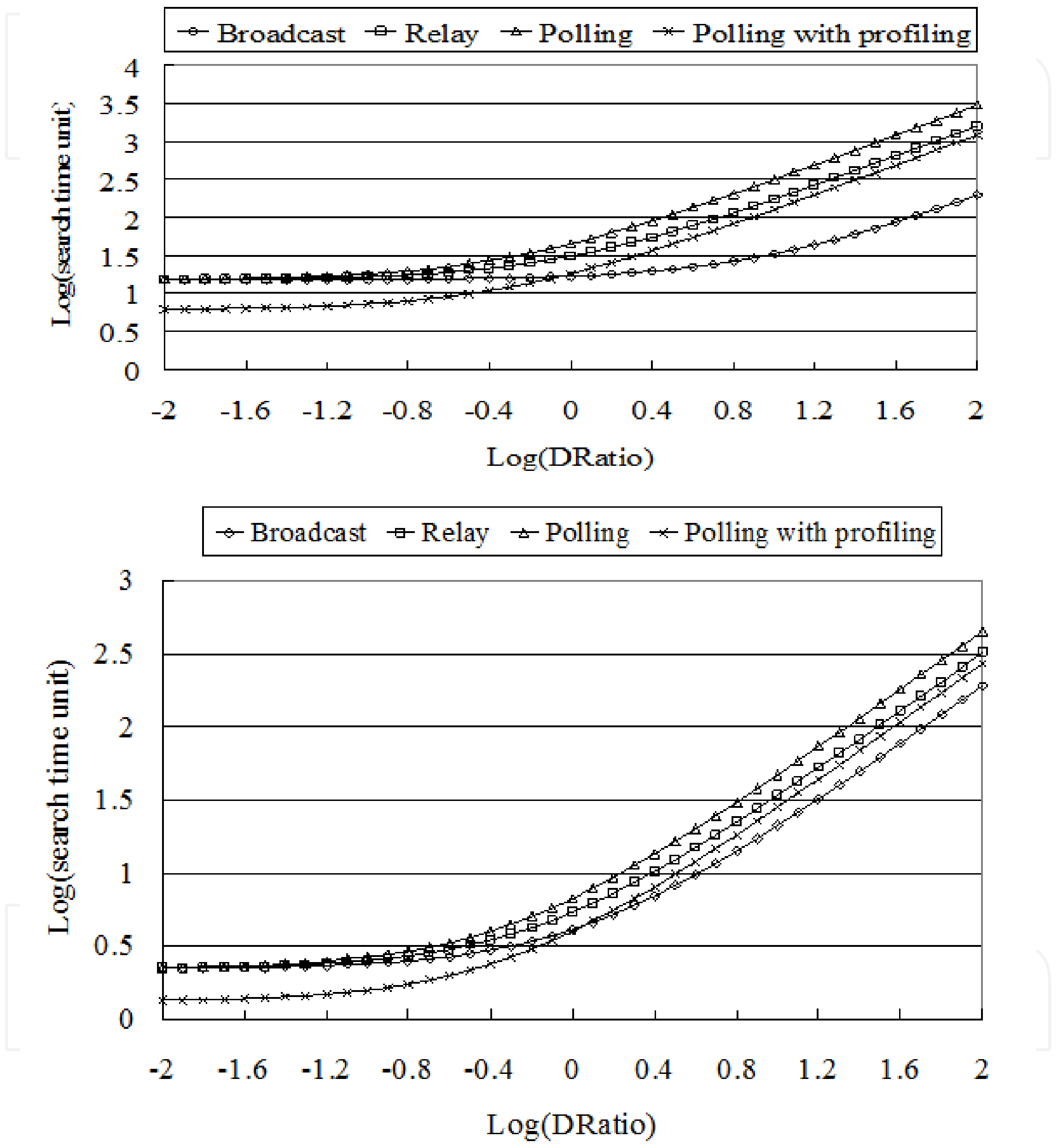


Figure 14. Search time Vs DRatio: (a) top NA = 42; (b) bottom NA = 6

Figure 15 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 Figure 15(a) depicts, the energy consumption of relay scheme and polling scheme are the same. Their consumptions and that of 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. Values of ERatio at the intersections of the curves in Figure 15(b) are about 3.16 ($10^{0.5}$) and 15.85($10^{1.2}$).

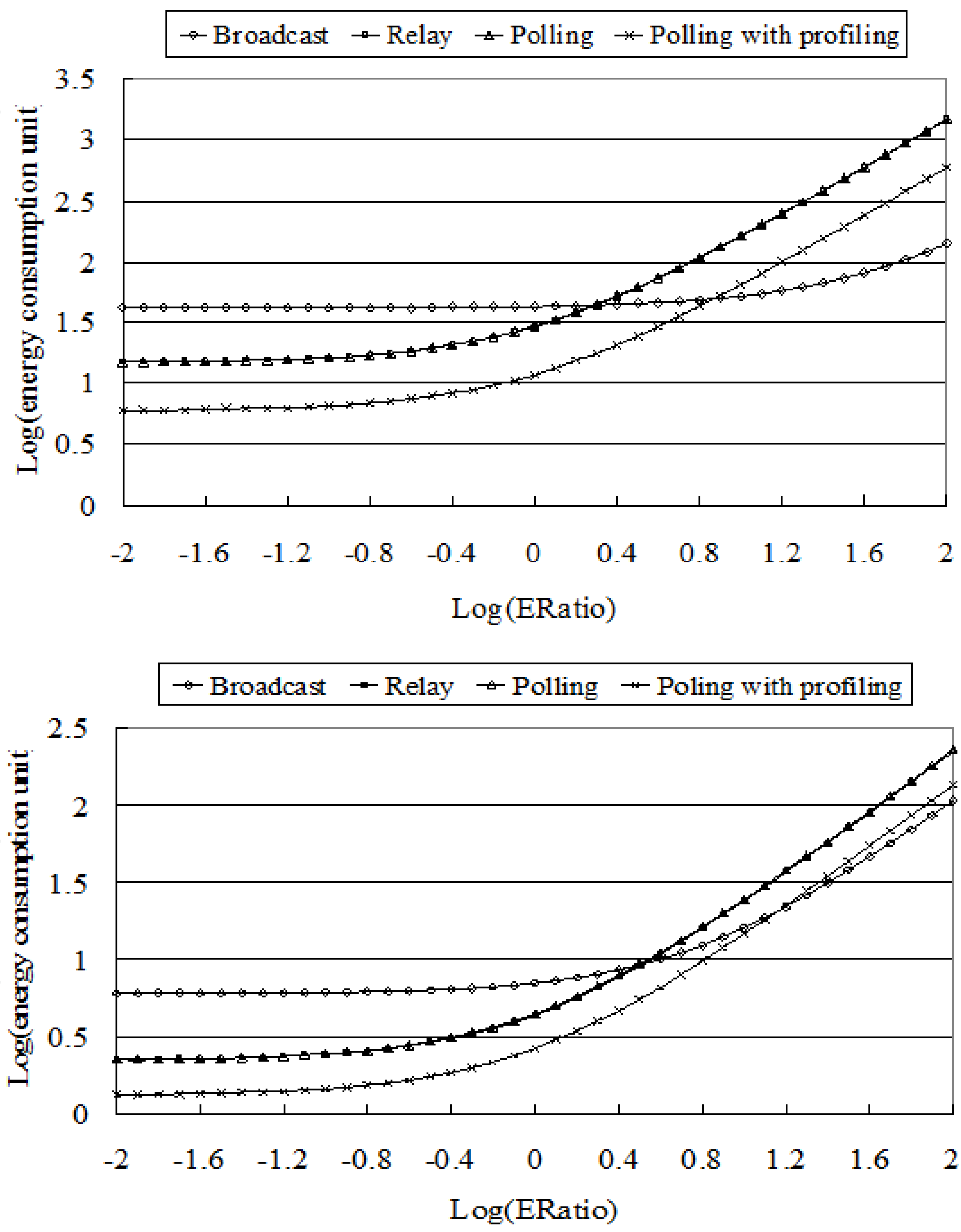
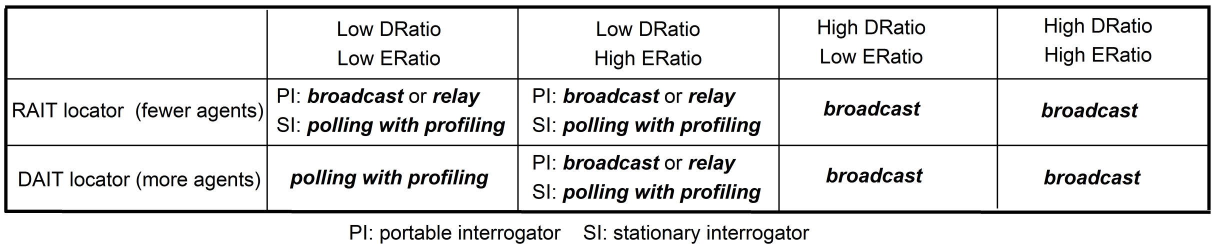


Figure 15. Energy consumption of agents Vs ERatio: (a) top NA = 42; (b) bottom NA = 6

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.



	Low DRatio Low ERatio	Low DRatio High ERatio	High DRatio Low ERatio	High DRatio High ERatio
RAIT locator (fewer agents)	PI: ***broadcast*** or ***relay*** SI: ***polling with profiling***	PI: ***broadcast*** or ***relay*** SI: ***polling with profiling***	***broadcast***	***broadcast***
DAIT locator (more agents)	***polling with profiling***	PI: ***broadcast*** or ***relay*** SI: ***polling with profiling***	***broadcast***	***broadcast***

PI: portable interrogator SI: stationary interrogator

PI: portable interrogator; SI: stationary interrogator

Table 3. Summary of suggested search schemes

7. Conclusion

We described here three alternative designs for RFID-based object locator. These object locators are extensible, reusable and low maintenance. They are easy for users to set up and use. 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 schemes are 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. Figure 13 depicts three ways to place agents. Putting agents further apart than locations shown in Figure 13(a) can create blind regions. Putting more agents closer than those indicated in Figure 13(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 Figure 13(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.

Acknowledgment

This work is partially supported by the Taiwan Academia Sinica thematic project SISARL (Sensor Information System for Active Retirees and Assisted Living (http://www.sisarl.org).

Author details

T. S. Chou[1] and J. W. S. Liu[2]

1 Computer Science Department, University of California at Irvine, Irvine, CA, USA

2 Institute of Information Science, Academia Sinica, Nangang, Taipei, Taiwan

References

[1] Hsu CF., Liao HY., Hsiu PC., Lin YS., Shih CS., Kuo TW., Liu JWS., Smart Pantries for Homes: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, SMC2006, 8-11 October 2006, Taipei, Taiwan; 2006. p4276-4283.

[2] Yeh HC., Hsiu PC., Tsai PH., Shih CS. and Liu JWS. APAMAT: A Prescription Algebra for Medication Authoring Tool: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, SMC2006, 8-11 October 2006, Taipei, Taiwan; 2006. p3676-3681.

[3] Hsu YT., Hsiao SF., Chiang CE, Chien YH., Tseng HW., Pang AC., Kuo TW., Chiang KH. Walker's Buddy: An Ultrasonic Dangerous Terrain Detection System: Proceed-

ings of IEEE International Conference on Systems, Man and Cybernetics, SMC2006, 8-11 October 2006, Taipei, Taiwan; 2006. p4292 – 4296.

[4] Tsai PH., Yu CY., Shih CS., Liu JWS. Smart Medication Dispensers: Design, Architecture and Implementation. IEEE Systems Journal 2011; 5(1). p99-110.

[5] Medication Reminders: http://www.epill.com/ (accessed 01 July 2012)

[6] Senior Care Products: http://www.agingcare.com/Products (accessed 01 July 2012)

[7] Alababa.com, showroom for object locators: http://www.alibaba.com/showroom/object-locator.html (accessed 01 July 2012)

[8] Hsu CC., Chen JH. A novel sensor assisted and RFID-based indoor tracking system for elderly living alone: Sensor 2011; 11. p10094-10113.

[9] Chou TS. Design and Implementation of Object Locator. MS thesis, Taiwan National Tsing Hua University, 2006

[10] Leong Kin Seong LK., Ng ML., Cole PH. The Reader Collision Problem in RFID Systems: Proceedings of IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, MAPE 2005, August 2005. p 658 – 661.

[11] EPCglobal standards overview: http://www.gs1.org/gsmp/kc/epcglobal (accessed 11 July 2012)

[12] ISO15693 Standard, OpenPCD: http://www.openpcd.org/ISO15693 (accessed 12 July 2012)

[13] Getting I. The Global Positioning System: IEEE Spectrum 1993; 30(12). p36–47.

[14] Want R., Hopper, A., Falcao V., Gibbons J. The Active Badge Location System: ACM Transactions on Information Systems. 1992 10(1). p91–102.

[15] Harter A., Hopper A. A New Location Technique for the Active Office. IEEE Personal Communications; 1997 4(5). p43–47.

[16] Nissanka BP., Chakraborty A., Balakrishnan A. The Cricket Location-Support System: Proceedings of ACM International Conference on Mobile Computing and Networking (MoBiCOM), 6 – 11 August, 2000, Boston MA USA. 2000. p32-43.

[17] Bahl P., Padmanabhan V. RADAR: An In-Building RF-based User Location and Tracking System: Proceedings of IEEE The Conference on Computer Communications (INFOCOM), 26 - 30 March, 2000, Tel-Aviv, Israel. 2000. p775-784.

[18] Konrad L., Welsh M. MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking: Proceedings of the International Workshop on Location and Context-Awareness at Pervasive, 12 – 13 May 2005, Oberpfaffenhofen near Munich, Germany. 2005. p489-503.

[19] Hightower J., Borriello G., Want R. SpotON: An Indoor 3-D Location Sensing Technology Based on RF Signal Strength. University of Washington, Seattle, Tech. Rep. 2000-02-02, February 2000.

[20] Castro P., Chiu P., Rremenek T., Muntz RR. A Probabilistic Room Location Service for Wireless Networked Environments: Proceedings of ACM International Conference on Ubiquitous Computing (UBICOMP), 30 September – 2 October 2, 2001, Atlanta, Georgia USA; 2001. p18-34.