2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware

Information-Gradient based Decentralized Data Management over RFID Tag Clouds

Tejas.A.Bapat, K. Selçuk Candan, V. Snehith Cherukuri, Hari Sundaram
Computer Science Department, Arizona State University, Tempe, Arizona, USA
Email: {tbapat,candan,venkata.cherukuri,hari.sundaram}@asu.edu

Abstract

The tasks in the physical environments are mainly information centric processes, such as search and exploration of physical objects. We have developed an informational environment, AURA *that supports object searches in the physical world*[1]*. The goal of AURA is to enable individuals to use the environment in which they function as a living (short-term) memory of their activities and of the objects with which they interact in this environment. To support physical searches, the environment that the user is occupying must be transparently embedded with relevant information and made accessible by in-situ search mechanisms. We achieve this through innovative algorithms that re-imagine a collection of environmentally distributed RFID tags to act as a distributed storage cloud that encodes the required information for attribute-based object search. Since RFID tags lack radio transmitters and, thus, cannot communicate among each other,* auraProp *and* auraSearch *leverage the movements of the humans in the environment to propagate information: as they move in the environment, users not only leave traces (or* aura*s) of their own activities, but also help further disseminate* aura*s of prior activities in the same space. This scheme creates an* information-gradient *in the physical environment which AURA then leverages to direct the user toward the object of interest.* auraSearch *significantly reduces the number of steps that the user has to walk while searching for a given object.*

1. Introduction

Consider the situation of Ben, who is, at home, trying to find a book that his wife had mentioned last week. The book does not appear to be in the bookshelf, so his wife must have taken it to read. At this point, Ben has three choices: to look for all the places his wife may potentially have taken the book to read, call her at the office where she works, or pick another book to read. If only the environment could have highlighted for him where his wife took the book, then he would not need to worry about calling her to ask about the book or having to choose another book to read.

1. This project is supported by the NSF grant "NSF, Design of Dense RFID Systems for Indexing in the Physical World across Space, Time, and Human Experience"

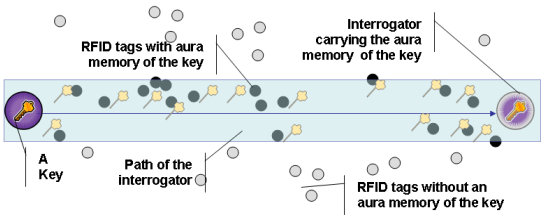
Human beings are functioning in environments that are shared by multiple individuals. In more public environments, such as shopping malls and book stores. While these environments are designed to support and, in fact encourage, synchronous social interactions, asynchronous social interactions in physical environments are left to secondary means (such as post-it notes), which fall short when the interaction is serendipitous. In [1], we provided a first look at AURA, an RFID-based informational environment to support searches and interactions in the physical world. The goal of AURA is to let the individuals to use the environment in which they function as a (short-term) memory of their activities and of the objects with which they interact. AURA achieves this by making use of RFID technology, involving small, cheap, and passive (no battery) radio tags which are attached to objects in the environment. In particular, AURA re-imagines the collection of RFID tags as a de-centralized, distributed storage cloud which can store information about the objects and activities within the environment.

RFID technology, involving small, cheap, and passive (no battery) radio tags, is rapidly gaining popularity and replacing other types of electomagnetic tags. Since RFID is gaining importance in supply chain management, with large corporations pushing for the use of RFID technology, new protocols (e.g. EPC GEN 2[2]) have been developed which allow for faster read/write rates and access control. One of the prominent features of EPC GEN 2 is that it lets the so called *interrogator* devices to write user supplied data on RFID tags. While such storage-enabled RFID tags are becoming increasingly cheaper and being incorporated into everyday objects, the following limitations of these devices make their use as a distributed storage space challenging and interesting for further investigation:

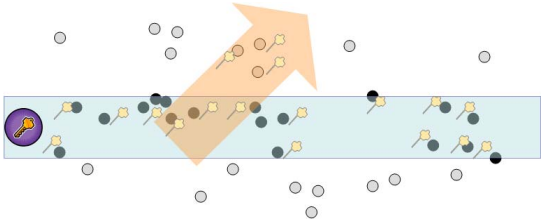
- The storage space available on each RFID tag is exceedingly small (only 100s of bytes).
- The lack of transmitters on the RFID tags makes it impossible for these devices to exchange information, without the help of an interrogator device that the users carry with them.
- Thus, the user herself has to be part of the dissemination and search processes.

In this paper, we describe the algorithms underlying *AURA* to tackle the above challenges: a novel *auraProp* algorithm disseminates information in the environment and a

978-0-7695-3650-7/09 $25.00 © 2009 IEEE
DOI 10.1109/MDM.2009.18



(a) *Aura memory propagation:* Interrogator-based information dissemination of auras along the user's path



(b) *Aura memory cross-propagation:* Potentially, aura's in the environment are further disseminated when other users' paths cross the original path

Figure 1. *AURA* dissemination

complementary *auraSearch* algorithm implements spatial searches for physical objects in the environment. In particular, *auraSearch* disseminates information along the most popular routes in the environment to ensure that most search queries can be answered with a low walk overhead.

2. Overview of AURA

In most environments people use existing pathways. *AURA* takes advantage of the walk patterns of the users to transparently disseminate the information into the entire environment. Specifically, *AURA* piggy-backs the information dissemination task onto the interrogator devices that the users carry with themselves and leverages people's mobility in the environment to *transparently* spread information into the environment: as the user moves from one region to the other, the interrogator device that she carries, picks up information about objects in the current vicinity and spreads this information along the path the user follows (Figure 1).

auraProp relies on a signature based encoding scheme to reduce the footprint of the attribute-value pairs on the extremely limited storage space available on the RFID tags [3]: each attribute-value pair is hashed to a a unique bit string (i.e., a signature); if a tag is to store more than one attribute-value pair, all these hashed bit-strings are logically *OR*ed to obtain a combined signature. The lookup is then performed by creating a search signature for the attribute-value pairs given by the user and then matching the bits in the search signature against the bits in the data signature encoded in the given tag. If any of the search bits is not found in a given tag, then that tag can be eliminated from further consideration.

A naive superimposition of propagated signatures with the local signatures would soon make the combined signature unusable because most bits will eventually be set to 1. To address this, AURA uses a novel erasure scheme, which turns a number of 1's in the combined *signature* to '0'. The *erasure rate*, ϵ, determines the probability with which a '1' in a given signature will be turned to a '0'. A side effect of the random bit-erasure process is that after the erasure some data on the tag will become unintelligible. However, since each signature is written redundantly and since the bit-erasure process is randomized, there will be other tags in the environment which will still have that particular data. We define *query match rate* as the number of tags which answer a given query to the total number of tags in a given region. Note that, as the distance of propagation increases (i.e., as user moves away from the original object), the probability that the 1's corresponding to this object signature are flipped to 0 increases. Thus, the number of tags that can answer to queries about this object decreases until it reaches a region where none of the tags can identify the object.

More importantly, however, with the erasure scheme ensures that the number of tags which identify a particular object decreases as the user walks away from the object. This creates an *information gradient* in space: AURA leverages the differences in the *query match rate* to guess the direction of information propagation and directs the user toward the source of information, essentially walking against the "erasure". We refer to this as the *information gradient* based search. In other words, *auraSearch* is exploiting the *erasure* process (which is also needed for preventing information overload) for creating reverse pointers in space that can be followed by the user.

The rest of the paper is organized in the following manner: Section 3 gives the background details of RFID. In Section 4, we give a detailed explanation of the encoding, dissemination and the search schemes. Section 5 gives a detailed explanation of the experimental setup and discusses the results. In Section 6, we present the related work. In Section 7, we present our conclusions.

3. RFID Technology Overview

Radio Frequency Identification (RFID) technology uses radio enabled tags attached to the objects (each with a unique serial number) to identify the objects. This unique serial number is called the *Electronic Product Code*(EPC). Reader/writer devices are used to interact with one or more tags within their range. Most RFID tags do not have a on chip battery and use the Radio Frequency signals from the interrogator device to power them up and reply by The EPC GEN 2 protocol provides a user memory bank for storing user specific data. It also allows the user to define the organization of this memory bank. This flexibility allows us to develop algorithms that encode the data on this memory location in an efficient manner. The interrogator manages

the tag population using three basic operations:

- *Select*: The device selects the tag population on which further operations will take place, based on a user specified criterion. At this phase, those tags that match the criterion are all activated (since RFID tags are not self-powered, being activated simply means that a specific bit on these tags is set to "1"). Tags can be selected by applying masks on the memory banks.
- *Inventory*: This operation identifies all the tags that have been activated in the *select* phase. At the end of this phase, the device knows about all the matching tags.
- *Access*: Once the tags have been identified, the device is ready to issue data management commands, *READ* and *WRITE*, to exchange data with the tags.

In AURA, as the user moves in the environment, the device goes through these three phases repeatedly to discover and interact with the new tags in the vicinity. The *select* operation is very fast since it involves a single broadcast command. On the other hand, the *inventory* and *access* operations are implementation dependent and are relatively slower: inventory is on the order of 100s to 1000s of tags per second [4], whereas access operations are on the order of 1000s of bytes per reads and writes [5]. The interaction range is different for reads and writes. Generally the read range is larger than the write range. These values depend on the interrogator device being used and the surrounding environment. Another factor which depends on the interrogator device is the directionality of the device. If the device is directional, it means that the tags should be facing(or within the specified directions) the interrogator and also are within its range to be detected. If the device is not directional, it means that the interrogator will detect tags in all directions.

4. AURA

AURA seamlessly embeds information about the environment on the RFID tags, so that this information can be used to answer queries. The two main components of AURA are (a) *auraProp*, which uses a erasure based scheme to propagate information and (b)*auraSearch*, which uses the local information to direct the user to the object of interest.

AURA's primary goal while disseminating information is to spread the information as far as possible. Since the communication in a RFID environment is interrogator driven, we need to develop a scheme in which information is transparently propagated while the user (carrying the interrogator) walks through the environment. The direction in which the user walks is the only direction in which information can be propagated. Once the user has moved from one region to the next, tags previously seen will now no longer be accessible. Thus previously seen information can be written on the newly seen tags, but not vice versa.

Table 1 shows the various system parameters. *N* is the number of tags that the interrogator can communicate with at any given point in time. Redundancy(*r*) is the number of tags used for encoding a local object signature. *L* is the bit length of the signature which is used to represent the objects and *m* is the number bits that are set to 1 in each signature. *Erasure rate*(ϵ) is the probability with which a bit originally set to 1 is turned to 0 to create an information gradient.

Table 1. AURA system parameters

Symbol	Definition
N	Number of tags in a region
r	Redundancy in storing local information
L	Length of signature stored at each tag
m	Number of bits set to 1 in each object signature
ϵ	Probability of turning a bit from 1 to 0(Erasure Rate)

4.1. Signature-based Data Management

A data signature is a bit vector of length L generated by first hashing the attribute values and/or contents of the document into bit strings and then superimposing (bitwise OR'ing) them together [3]. For searching, a search signature is generated using the same hash function for the search criteria (e.g., keywords) specified by the user. By matching the search signature against all the existing data signatures by performing a bitwise comparison, one can quickly eliminate the records that do match the search criteria. If any of the search bits is not found in the signature, then that data signature can be eliminated from further consideration. When using signatures for search, there can also be cases where all the bits match but the document itself does not match the search criteria. This case referred to as a *false positive* occurs due to the superimposing of various hash codes generated for particular attribute values. Let us assume that a sentence with n terms has been combined to form a signature of length L-bits, where exactly m-bits are randomly set ($m \leq L$), for each word in the sentence. It is known that the rate of false positives is $(1-e^{-\frac{nm}{L}})^m$. Thus, in general, it is possible to pick m such that the number of false positives will be small.

4.2. AURA Signatures

In *AURA*, each object has a set of descriptive attribute-value pairs. For example, the book that Ben was looking in the Introduction may have the following set of attribute-value pairs: $\{type ='' book'', keyword ='' economy'', usertag ='' favorite''\}$.Each attribute-value pair is hashed to a a unique bit string (i.e., a signature or the *aura*). If a tag is to store more than one attribute-value pair, all these hashed bit-strings are logically *OR*ed to obtain a combined aura signature.

4.3. AuraProp: Information Dissemination

AuraProp piggy-backs information dissemination onto the interrogator devices to leverage people's mobility in the

environment to *transparently* spread information: as the user moves among regions, the interrogator device that she carries, picks up information about objects in the current vicinity and spreads this information along user's path.

4.3.1. Aura Storage. Each RFID tag is split into two memory banks: *local aura bank* and the *aura memory bank*. The local aura bank contains only auras of the attribute-value pairs specific to that tag (or the object on which the tag is attached). The aura memory bank, however, may contain propagated auras of objects that are physically elsewhere.

As the user moves in the environment, the interrogator device reads the data signatures in the environment and creates a set of combined *local object signatures*. Each combined local object signature is created by selecting a subset of the objects randomly and merging their signatures. The membership of each set is determined by using a consistent hashing function [6]. The fraction of the total number of objects which participate in the creation of a *local object signature* is referred to as the *redundancy factor (r)*. These local object signatures are written back to the available tags in the environment.

4.3.2. Propagation Models. As the user continues her movement, the interrogator repeatedly reads *local auras* written in the environment and combines them with previously seen aura signatures into a so-called *aura memory* signatures. These *aura memories* are stored in the interrogator and are carried to the next region along the walk. The aura memories are also distributed onto the RFID tags in the new region (Figure 1). Based on what information is carried between regions and how the information is subsequently written, AURA can use different propagation models:

- *Overwrite - No Cross-propagation:* In the simplest model, the interrogator only collects local aura signatures from the tags and ignores the aura memories stored on them. At each step[2], the interrogator completely overwrites the aura memories on the tags it meets with previously collected aura-memories.
- *OR only at junctions - No Cross-propagation:* This is a special case of the previous model. At pathway crossings and junctions, instead of overwriting the aura-memories on the tags, the interrogator ORs its aura-memories with the aura-memories on these tags.
- *OR - No Cross-propagation* As the user walks, the interrogator collects the local aura signatures and superimposes these collected signatures with the existing signatures while writing them into a region. When the environment is covered with sufficient walks along all the directions, the information about an object is disseminated in the entire space as users seeing this object walk to different parts of the environment.

2. A *step* is a user's movement from one region in the space to the next one. The interrogator reads and writes once per step.

The above propagation models can propagate auras to only places where people who met the original objects later visit. (Figure 1(a)). The following model takes advantage of users' crossing paths to help disseminate information:

- *OR with Cross-propagation* As the user walks, the interrogator collects both local aura signatures and the aura memories stored on the tags. The aura memories an interrogator picks up from the environment may belong to objects that the interrogator never met.

The last model helps propagate information faster. This is the default propagation model in *AURA*. A naive OR-based superimposition of propagated aura memories would soon make the aura memories stored on the tags unusable because most bits will eventually be set to 1. Since naive superimposition will eventually lead to information overload, we introduce a novel erasure scheme, which turns a select number of 1's in the collected *aura-memories* to '0'.

4.3.3. Controlled Erasure of Aura Memories. In order to prevent information overload (and to create an information gradient as described in Section 4.4.1) the interrogator devices randomly erase portions of the aura memories they carry with themselves. To achieve this, the interrogator turns a number of 1's in the combined *aura memory* it carries to '0'. The *erasure rate*, ϵ, determines the probability with which a '1' in a given signature will be turned to a '0'.

Note that, if performed naively, the *erasure* scheme will degrade with high number of walks along a given path. If different walks along the same path carry different erasure-variants of the same signature, then superimposition of these variants will eventually overload the signature. Hence *auraProp* ensures that the same bits are turned from 1 to 0 for different walks by using the *tag Id* on each tag as the seed for the consistent hash function [6]. This ensures that the same bits are turned from 1 to 0 in the local aura memory of a tag for all the walks across a given region.

A side effect of the random bit-erasure process is that after the erasure some data on the tag will become unintelligible. However, since each signature is written redundantly and since the bit-erasure process is randomized, there will be other tags in the environment which will still have that particular data. We define *query match rate* as the number of tags which answer a given query to the total number of tags in a given region. Note that, as the distance of propagation increases (i.e., as user moves away from the original object), the probability that the 1's corresponding to this object signature are flipped to 0 increases. Thus, the number of tags that can answer to queries about this object decreases until it reaches a region where none of the tags can identify the object. As discussed in the next section, AURA leverages the correlation between distance and *query match rate* to implement *auraSearch*.

The outline of the information dissemination algorithms is provided in Algorithm 4.1.

Algorithm 4.1 Information Dissemination

procedure $disseminateInformation()$
1: $PCS[1...N] \leftarrow null$ {Initializing the combined signatures to null }
2: **for** $currentStep = 1$ to $pathLength$ **do**
3: LOS[1...N] $\leftarrow$ *collectLocalInformation(currentStep)*
4: PCS'[1...N]$\leftarrow$ EraseBits(PCS[1...N], errorRate)
5: **for** $i = 1$ to N **do**
6: PCS[i] $\leftarrow PCS[i]$ OR $PCS'[i]$
7: Write PCS[i] to tag i
8: **end for**
9: **end for**

procedure $collectLocalInformation(currentStep)$
1: totalTags $\leftarrow$ number of tags in the current radio range
2: **for** $tag\_no = 1$ to $totalTags$ **do**
3: **if** current tag contains previously written information **then**
4: LS[tag_no] $\leftarrow$ previously written information on the tag
5: **else**
6: LS[tag_no] $\leftarrow$ generateNeighborhoodSignature(tag_no)
7: **end if**
8: **end for**
9: **return** LS[1...N]

procedure $generateNeighborhoodSignature(index\_no)$
1: $S_{index\_no} \leftarrow$ *H*(index_no,redundancy)
2: LOS $\leftarrow S_{index\_no}$[1] OR S_{index_no}[2] OR ... S_{index_no}[m]
3: **return** LOS

procedure $H(index\_no, redundancy)$
1: $S_{index\_no} \leftarrow$ Object Signatures of a subset of tags, using a consistent hashing function.
2: **return** S_{index_no}

Algorithm 4.2 DFS Search

input: Attributes of the object of interest.
output: Directions which the user follows to reach the target.
procedure $DFSSearch()$
Require: Information gradient formed by the Information dissemination process
1: QS $\leftarrow$ generate query signature for given attribute(s)
2: $WalkDirection \leftarrow getNextBestRegiontoWalk()$
3: Direct the user to walk in *Walk Direction*
4: Initiate stack to empty
5: **while** $walkdirection \neq Cantproceed$ && $\neg pathWalkedTooLong$ **do**
6: **if** desired object is in the current region **then**
7: **return** searchSuccessful {User reached the object}
8: **else**
9: Walk Direction $\leftarrow$ getNextBestRegiontoWalk(stack)
10: Direct the user to walk in the *Walk Direction*
11: **end if**
12: **end while**
13: **return** searchUnsuccessful

procedure $getNextBestRegiontoWalk(stack)$
1: PreferredRegion $\leftarrow$ getRankedNeighboringRegions()
2: **if** PreferredRegion $\neq$ null **then**
3: Stack.push (all elements of *PreferredRegion*)
4: **end if**
5: nextRegion $\leftarrow$ Stack.pop
6: **return** nextRegion

procedure $getRankedNeighboringRegions()$
1: Neighbors $\leftarrow$ directions where tags match the query.
2: Sort *Neighbors* DESC on match rate
3: **return** *Neighbours*

4.4. AuraSearch

The user will use the interrogator device to specify the attribute-value pairs of the object which she is interested in searching. *auraSearch* uses the same hashing mechanism as used in *auraProp* to generate a query signature. At any given location, the interrogator can read only the tags in the vicinity. Thus, the information that is available to direct the user to the object of interest is limited. *AuraSearch* leverages the *information gradient* created by the erasure scheme to aid in the search process.

4.4.1. Erasure based Information Gradient. During the dissemination process, *AURA* uses lower power levels of the interrogators directional antennas to identify the tags for writing. During search, on the other hand, *AURA* uses antennas at higher power levels to read from the current region as well as from adjoining regions in a particular direction. This asymmetric design enables AURA to leverage the differences in the *query match rate* to guess the direction of information propagation and to direct the user toward the source of information, essentially walking against the "erasure". In other words, *AURA* leverages the *erasure* process (which was also needed for preventing information overload) for creating reverse pointers in space that can be followed by the user.

Since, for information gradient based search, we only need the count of the number of objects which match the query signature, we can do this by making use of the *Select* and *Inventory* phases of GEN2. The *Select* command takes a bit-mask as a parameter and applies this mask on the user memory(also specified as a parameter) of all the tags in the current range of the interrogator. When the bit-mask used is the query signature, all those objects which match this query are selected. The *Inventory* phase can be used to get a count of all the objects that were selected in the *Select* phase. As explained in Section 3, since there is no data exchange involved, the time required to collect the required information is minimal.

Ideally the next best region should have a higher match ratio than the current region, but this might not always be true. This arises when the current region has a higher match rate than ideal because of the false positives. In the rest of this section, we describe three information-gradient based search schemes (depth first, restricted depth first, and confidence-based)for guiding the user toward the object of interest. The underlying common policy employed by all these algorithms is to select the next region in the direction where the ratio of tags that answer a query to the total number of tags is maximum. Different schemes recover from errors differently to achieve high *success rates*, while keeping the *walk length* back to the source close to optimal.

4.4.2. Depth First Search. In this strategy, the user picks the best direction (with highest information gradient) to follow in a greedy fashion. At each step, the user also stores the other possible options available in a stack with the worst possible direction (with lowest information gradient) in the bottom. When the user reaches a point where none of his neighbors (except for the one from where he previously came) can answer the query, the searcher pops the next best closest region from the stack and the interrogator will guide the user back to this new location. As shown in

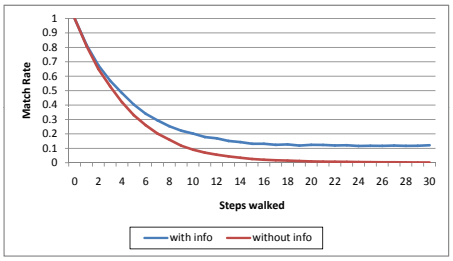


Figure 2. The degree of matches (the number of tags that have signatures matching the query) at different distances; ambient aura noise in the environment reduces erasure rate

Algorithm 4.2, this "depth-first" like search ends when the user has reached the region where the object is located (i.e., search terminated successfully) or when the user failed to locate the object (i.e., search terminated with failure).

4.4.3. Restricted Confidence-based Search. In the previous scheme, by following the gradient strictly, the user may end up following long paths. To overcome this, we introduce a *confidence score* which helps us to reduce the walk length: While searching for an object the user increments the *confidence score* each time she moves from a region with lower match rate to a region with higher match rate. The scheme decrements the *confidence score* every time the user is forced to walk to a region which has a lower match rate than the current one. This scheme may result in a lower success rate than depth first search, but (as the experiments in Section 5 show) the user walks tend to be shorter.

4.4.4. Confidence Based Search. In this scheme, only the *confidence score* is used as the stopping condition to stop looking for an object in a particular direction. The user can walk through regions even when none of the tags in the region match the query. This scheme is useful in cases where the searcher is starting from a region far away from the object of interest. In such a scenario, especially when an overwrite-based propagation scheme (Section 4.3.2) is used, there could be regions along the correct path where none of the tags match the query signature. A DFS based scheme will always be unsuccessful in such cases, but a confidence based scheme may help overcome gaps in gradient.

4.5. Ambient Aura Noise

The *erasure scheme* limits the distance up to which the information about an object can be disseminated in the environment. Figure 2 shows this information loss along a single walk. As seen in this figure, as the user walks away from the source object, the number of tags that answer for that object query drops (the lower curve in the figure).

For any given object signature, with m bits set to 1, erasure of one of these m bits in the combined signature, leads to the creation of at least $m-1$ dormant bits set to 1. These dormant bits do not aid in this object's identification, while they still remain in the environment. While not being immediately useful, these dormant bits may become active (i.e., useful for search) when superimpositions with other auras set the erased bit back to 1. The upper curve in Figure 2 shows that dormant bits (also referred to *ambient aura noise*) reduces the erasure rate and help carry information further.

4.5.1. Propagation Distance without Ambient Noise. To study the effect of ambient aura noise on information propagation, we first model the information propagation along a path ignoring the noise present in the environment. The bit at location β of aura memory signature is turned from 1 to 0 only by erasure. Therefore the probability that β which is set to '1' remains a '1' after d steps is given by $(1-\epsilon)^d$. The propagated signature matches the query signature is when all the m bits in the query signature are also set in the data signature. Hence the probability that a tag at distance d matches a query signature with m bits set to '1' is given by $((1-\epsilon)^d)^m$.

As we discussed in Section 4.3.1, *auraProp* introduces replication before propagating signatures by superimposing $N \times r$ auras into a single aura memory signature. Thus, information about a signature from a region is lost only when neither of the $N \times r$ signatures match the query signature any longer. Thus, the probability with which a signature is propagated d steps away can be calculated as[3]

$$P_{(\epsilon,d)} = 1 - (1 - (1-\epsilon)^{d \times m})^{N \times r}.$$

4.5.2. Propagation Distance with Ambient Noise. Let T be the total objects in the information space and each object be described by an aura which has m out L bits set to '1'. The probability that some bit position β is set in exactly A auras is given by

$$p(A) = \binom{T}{A} \times (\frac{m}{L})^A \times (\frac{1-m}{L})^{T-A}$$

When the noise source[4] is at distance d steps from the current location of the searcher, the probability that the bit is set to 1 because of noise is $(1-\epsilon)^d$. As the number, S, of noise sources in the environment increases, the average distance Δ_{avg} between any region and the nearest noise source decreases. Therefore the probability by which the bit β in some aura signature is set to '1' because of S other auras (acting as sources of noise) is given by

$$P_{noise} = \frac{1}{1-p_0} \times \sum (p(S+1) \times (1-\epsilon)^{\Delta_{avg}}).$$

3. The derivations of the various models presented in this paper are omitted due to space constraints.

4. To simplify the model, when there is more than one noise source in the environment, we consider the effect of only the nearest noise source as it is the most predominant source of noise.

For a query signature with m bits set to 1, the false positive match rate because of noise is given by $(P_{noise})^m$. A bit which was 1 is set to 0, only when it is erased by the erasure scheme and not set back to 1 by the noise. Given this, probability that information is propagated to a region d steps away from an object is given by

$$P_{(P_{noise},\epsilon,d)} = 1 - (1 - (1 - (\quad (1 - (1 - \epsilon)^d) \times (1 - P_{noise})))^m)^{N \times r}$$

We evaluate this model in the experiments section; Section 5.

4.6. Pathways vs, Junctions

Most physical spaces consist of pathways and junctions. In *AURA*, junctions carry additional significance: they are the critical points at which the directional decisions are to be made during search. This has an impact on how the erasures are done: along the straight pathways, we need to introduce just enough *information gradient* to distinguish between front and backward directions; on the other hand at junctions, we need to have sufficient erasure to prevent overload, while providing good *information gradient*.

Given a query with m bits set to 1, the false positive rate is given $(1 - P)^m$, where P is the probability that a bit is set to 0 in the tag. Let us assume that there are N objects belonging to the *information space* in each region along the pathway and let the redundancy factor be r. The probability that a given bit is set to 0 in the combined *aura signature* is $p = (1 - \frac{m}{L})^{N \times r}$, where $N \times r$ is the number of aura signatures superimposed. At any given region along the walk, a bit, β, in the *aura memory signature* stored in a tag is 0 when β is 0 in the *aura-signature* being carried by the interrogator and

- β was 0 in the previous *aura-memory signature* stored on the tag, or
- β was 1 in the previous *aura-memory* but was turned to a 0 by the *erasure* scheme.

Let us assume that the length of a pathway from one junction, a, to the other, b, is k steps. We can estimate the probability, $P_{(a,b),\delta}$, that a bit in an *aura-memory signature* stored δ steps away from the source junction, a, (and $k - \delta$ steps away from junction, b) is set to 0 as a function of the *erasure rate*, ϵ_{path}, along the path, as follows:

$$p_{(a,b),\delta} \times \prod_{x \in \{-1,+1\}} \left(P_{(a,b),\delta+x} + (1 - P_{(a,b),\delta+x}) \times \epsilon_{path}\right).$$

Here, $p_{(a,b),\delta}$ is the probability that a given bit is set to 0 in the *aura-memory* the interrogator carries with. Also, $P_{(a,b),0} = P_a$ and $P_{(a,b),k+1} = P_b$. These can be computed as follows: Let $\epsilon_{junction}$ be the erasure rate AURA employs immediately before a pathway joins to a junction: Let us consider a junction, a, with u pathways leading to other junctions, $b_1, b_2, \ldots, b_u$, respectively. We can compute the probability of a bit in the *aura-memory* stored at the junction j_a is set to 0, as a function of the probabilities, $P_{(a,b_i),1}$, for the closest regions along each of the pathways, as

$$P_a = p_a \times \prod_{i=1}^{u} \left(P_{(a,b_i),1} + \left(1 - P_{(a,b_i),1}\right) \times \epsilon_{junction}\right).$$

The above two equations link the probability that a bit is set to 0 in any given region to the probability that a bit is set to 0 in any neighbor. In general, erasures along a path is set lower than the junction erasures as the need for information gradient is much lower along straight paths. We evaluate this in the experiments presented in the next section.

5. Experiments

The performance of AURA can be analyzed by considering two measures: the *success rate* and the *walk overhead*. The *success rate* is defined as the ratio of the queries for which the user is able to finds the object successfully using *auraSearch*. Let optimal walk length be the number of steps the user has to walk to reach the object of interest is she knew its location. The *walk overhead* is defined as the ratio of the difference between the number of steps that the user walks using the *auraSearch* scheme and the optimal walk length to the optimal walk length. The *successful walk overhead* on the other hand is the walk overhead only for those queries that succeeded.

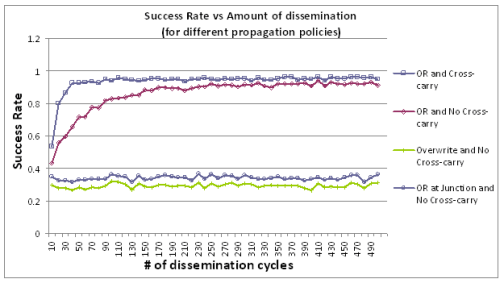
5.1. Experimental Setup

Table 2 shows the various parameters used and also how these parameters were varied. Based on a real world RFID reader range, we set the read range of the interrogators at least 3 meters and the write range 1 meter.

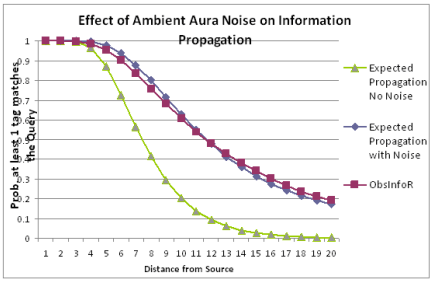
Table 2. Parameter Settings for the Experiments

Parameter Name	Range of Values
Length of the Signature(L) in bits	1000,2000
Number of bits set to 1 in each Signature(m)	2,4
Number of objects in a WRITE region	10, 20
Erasure Rate(ϵ)	0.05,0.1,0.2
Replication Factor(r)	100%

Layout Models. In order to simulate the physical world, we used a number of floor plans in which users walk to model a real world scenario. We varied the sizes of the floor plans and randomly generated user paths for each of these floor plans. We model a home environment as a 20m x 20m region (10x10 cell grid). For this home setup, we create two different path models (a) 2 horizontal and 2 vertical paths (b) 4 horizontal and 4 vertical paths. Each path is 2 meters wide corresponding to a single cell in the grid and runs from one end of the home to the other end. The locations of these paths in the home are chosen randomly. For the home environment, we assume that all the tags in the environment have auras. In addition, we also experimented with a mall-like configuration: a 50 x 50 size grid, where each cell in



(a)



(b)

Figure 3. The performance of *auraProp* for the home setup(m=4,L=1000,ϵ = 0.1, N=10, r=100%) (a) for different propagation schemes and (b) with noise

the grid is 2 x 2 meters. We used 5 vertical and 5 horizontal path ways to cover the region. In each region we assume 10% of the tags participate with *aura*s.

Walk Model. Users walk along randomly selected paths. When arriving to a junction, the user continues to walk straight with probability 0.6 or walks in the other available directions with equal probability. The walk terminates when the user reaches the boundary.

Query Model. We generate random queries for the objects. Each of these queries is issued by the user starting at a randomly selected point on the paths.

5.2. Results and Discussion

Dissemination Effectiveness. Figure 3(a) shows the effect of the various information propagation models on the success rate. The "OR with cross propagation" takes advantage of the previously written information at the junctions and hence disseminates the information faster. After 40 dissemination cycles in the region, the system can answer 95% of the queries successfully. Since "ORing without cross-propagation" scheme only superimposes the previously seen information with the current information, it takes more walks for all the information to be disseminated in environment. With sufficient propagation cycle, this dissemination scheme eventually performs similar to the default scheme. The other two schemes, which overwrite previously seen information, achieve significantly lower success rates.

Dissemination and Ambient Aura Noise. Figure 3(b) shows three curves: the observed propagation, propagation predicted without a noise model, and propagation predicted

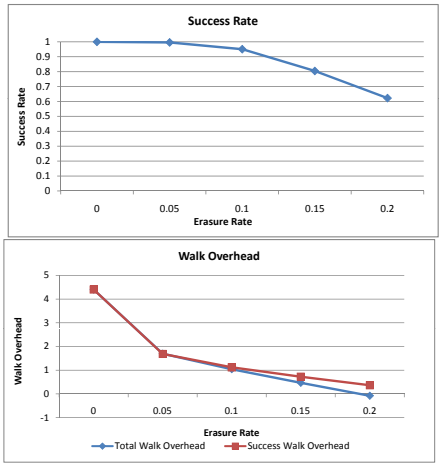


Figure 4. Effect of ϵ (m=4,L=2000,N=10, r=100%)

Table 3. Performance of different search protocols(2x2, m=2,L=1000,N=10, r=100%)

	DFS	Rest-Conf	Conf	No Conf/No DFS
Success rate	0.955	0.947	0.967	0.699
Walk overhead	0.71	0.59	0.8	0.06
Successful walk overhead	0.76	0.60	0.79	0.01

based on the noise model. This figure shows that the observed degree of propagation matches the amount expected based on the noise model and that, indeed, the ambient aura noise reduces the erasure rate (thus may necessitates interrogators that have wider read ranges that can leverage small information gradients) and carries information further.

Erasure and Search. Figure 4 shows the effect of the *erasure Rate* ϵ on the success rate as well as the walk overhead (for the above setup). For $\epsilon = 0$ there is no information gradient created and hence the user has to perform an exhaustive search to find his object of interest. Keeping a stack for DFS ensures that the success rate is 100%, but the walk overhead is very large. As the erasure rate increases the information gradient is formed and hence the successful walk overhead reduces. On the other hand, for higher values of ϵ, since the information is forgotten very quickly, the success rates drops[5].

Effect of Different Search Protocols. Table 3 shows the walk overhead in this configuration is low and the success rate is high for all protocols. The restricted confidence based scheme has the least walk overhead since it stops searching (along non-promising paths) earlier than a DFS scheme. This results in slightly lower success rates than the DFS and the confidence based scheme. While the restricted confidence based scheme is a good solution in this setup where success rate is already high, DFS might be appropriate in situations where users may get lost more easily. In this setup, the purely confidence based scheme achieves a slightly higher

5. Note that, when $\epsilon = 0.2$, the walk overhead is negative (while the succesful walk overhead is positive). This highlights that, due to the failures, the user has walked less than the necessary walk length to find the object.

Table 4. Varying path patterns(m=2,N=10, L=2000, r=100%)

DFS	Success Rate		
Config.	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
2x2	0.983	0.793	0.509
4x4	0.986	0.984	0.937
DFS	**Walk Overhead**		
Config.	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
2x2	1.12	0.26	0.13
4x4	5.77	3.35	1.67
DFS	**Successful Walk Overhead**		
Config.	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
2x2	1.16	0.56	0.09
4x4	5.43	3.27	2.48

Table 5. Effects of the path and junction erasure rates (m=2,L=4000,N=10, r=100%; search stops after 200 steps)

DFS	ϵ_{Path}_$\epsilon_{Junction}$		
	0.0_0.3	**0.01**_0.3	**0.05**_0.3
Prob. bit is set	0.39	0.33	0.24
Success Rate	0.79	0.99	0.7
Walk Overhead	2.6	1.8	1.4
	0.01_**0.2**	0.01_**0.3**	0.01_**0.4**
Prob. bit is set	0.46	0.33	0.22
Success Rate	0.79	0.99	0.7
Walk Overhead	3.8	1.8	0.64

success rate, but this comes with an increase in the walk overhead. Table 3 also shows an extreme protocol where the user only walks along the path with non-decreasing match rates and does not rely on DFS to recover from errors. As can be seen here, the walk overhead rate drops to almost 0, but the success rate also drops to 70%. The DFS based search scheme performs well in terms of both the success rate and the walk overhead and hence we use this as our search scheme for the rest of the experiments.

Effect of Path Density. Table 4 shows the effect of path density on the success rate and walk overhead. For 2x2 paths, an erasure of $\epsilon = 0.1$ results in a high success rate and a low walk overhead. For a denser environment, the same erasure rate is not sufficient to form an information gradient, and thus the walk overhead is very large. This is because of the loss of erasure at the junctions, which results in poor directions at where having good directions is most important. Thus, for a higher erasure rate of $\epsilon = 0.3$, the 4x4 path setup also achieves a comparable result.

Effects of Path and Junction Erasure Rates. Table 5 shows the DFS search result for the mall environment. In this table, ϵ_{path} corresponds to the *erasure rate* introduced along the pathways and $\epsilon_{junction}$ corresponds to the *erasure rate* used at the junctions. As can be seen here, the number of 1s in the environment drop significantly as the e_{path} is increased. When $e_{path} = 0.0$, searches fail because there is no information gradient. For $e_{path} = 0.05$, the DFS success rate is low because information is not propagated sufficiently. As can be also seen in the table, the $\epsilon_{junction}$ parameter has a drastic effect on the number of 1s and, hence, the success and walk overheads. In this configuration, the best performance is achieved with a low path erasure rate (0.01) and a relatively high junction erasure rate (0.3). But, higher junction erasures do not help.

Table 6. Results for environments with similar objects(2x2, m=2,L=1000,N=10, r=100%, ϵ=0.1)

# of Repeated Attributes	Success Rate	% Closest Found	Walk Overhead from the Closest Instance
1	0.954	NA	NA
2	0.997	80	0.18
4	0.999	70.7	0.38

Effect of Multiple Matching Data Sources. It is common in any environment that multiple objects have the same attributes. For example there can be multiple pens, glasses, etc. in a home environment. In such cases, when the user queries for an object based on an attribute, it is ideal that the user is directed to that object which is closest to him from his current location. Table 6 shows that *auraSearch* guides the user to the closest object with high probability. Since the match ratio along the direction of the closest object is higher than for a farther object, the user will be guided in the direction of the closest object by our search scheme.

Effect of Tag Mobility in the Environment. Since the search scheme is based on the distribution of the tags, in environments where tags may move, this may result in undesirable changes in the *information gradient*. To study the effects of tag mobility on search, we disseminated information in the environment. After this, we selected 10%, 20% and 40% of the objects in the environment to be either completely moved out of the environment (as in items purchased in a shop) or to be moved to some other region in the same environment (e.g., re-decoration). These objects are moved without the presence of the interrogator device, so the environment is not aware of the changes. Figure 5 shows the success rate:

- When objects move out completely (up to 40% tags in the experiments), the search is not affected for objects which are currently present in the environment. This is because, when a random set of objects from a region move out completely, the ratio (i.e., density) of the tags which answer a given query remains constant, thereby not affecting the *information gradient*.
- When tags move within the environment, this fraction does not remain constant as tags moving into the environment skew the information gradients. Thus, large ($\geq$ 10%) restructuring of the environment should be done in the presence of interrogator devices.

6. Related Work

There is limited prior work on the use of RFID tags as distributed storage systems. Most existing works, such as Drishti [7] (which encodes information about nearby landmarks on RFID tags to help navigate user who are blind), assume

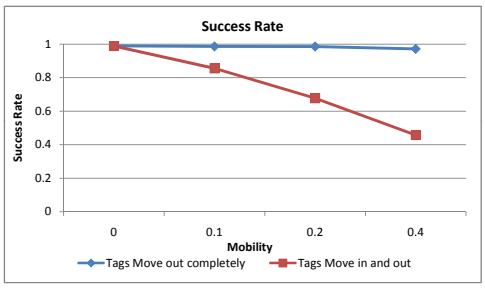


Figure 5. The performance of auraSearch with tag mobility(2x2, m=2,L=1000,N=10, r=100%, ϵ=0.1)

that the information is static. Others, such as Rfind [8], store information about the environment in a central database. [9], [10] use RFID tags to attached to objects to access relevant information. Both of these schemes, however, assume that a central server can provide the relevant information given the unique identifiers stored on the RFID tags. Unlike these solutions, AURA is neither read-only nor assumes that there is a central database. Instead, it embeds the information into the environment and disseminates it along the popular routes (with the help of those very individuals using those routes).

Organizing nodes and data in a distributed system and supporting efficient content based retrieval has recently gained attention [11], [12], [13]. Yet, most solutions assume that nodes can communicate with each other to establish an ordering over the search key. In *AURA*, we cannot rely on this assumption, because tags are passive and cannot communicate with each other. Thus, simple schemes for searching, such as random walks [14] or flooding based searches, are not applicable. [15] uses *attenuated Bloom filters* [15] along with an existing overlay for finding nearby replicas[16]. At each level, i, of the Bloom filter of depth d, one can store information about all nodes that are at a distance i from the current node. Such a filter would be associated with each neighbor link and would give the probability of finding the object being searched through that neighbor. For finding objects that are further apart, [16] relies on a separate deterministic algorithm. The paper shows that *signature* based approaches lead to more efficient search solutions at a lower storage cost than index based mechanisms. A shortcoming of these approaches, however, is that, they can only be useful for searching information up to a fixed distance, d. Our goal is to use *auraSearch* to search for objects as far as possible (keeping a long trail of any object) without restricting searches to a fixed number of steps. We also make efficient use of storage space since we store only a single signature for all the neighbors as compared to d signatures stored for each neighbor in [15].

7. Conclusion

In this paper, we described an RFID based environment, *AURA*, to support spatial searches for physical objects. AURA works using established RFID protocols and interrogator technologies and is based on innovative algorithms that re-imagines the collection of RFID tags to act as a distributed storage cloud. These innovative algorithms tackle significant challenges including the limited storage space on the tags and the fact that RFID tags cannot communicate with each other. In particular, we introduce the concept of aura which encodes object-memory into the environment. Using signature based techniques, *auraProp* distributes auras along the paths commonly followed by the users. An *erasure* scheme both prevents overload and creates an *information gradient* that enables users to locate objects in the physical space given attribute based searches.

References

[1] T. Bapat, K.S. Candan, V. Cherukuri, and H. Sundaram. "AURA: Enabling Attribute-based Spatial Search in RFID Rich Environments," in *ICDE*, 2009.

[2] EPCGlobal, "Specification for rfid air interface: Epc radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860 mhz - 960mhz," Tech. Rep., Jan. 2005.

[3] C. Faloutsos and S. Christodoulakis, "Signature files: an access method for documents and its analytical performance evaluation," *ACM TOIS*, vol. 2, no. 4, pp. 267–288, 1984.

[4] Wikipedia, "Singulation — wikipedia, the free encyclopedia," 2008, [Online; accessed 2-November-2008]. http://en.wikipedia.org/w/index.php?title=Singulation

[5] "Fasttrack series rfid tags singulation." http://www.datascansystems.com/upload/Data_Sheets/RFID/lrp125-250tags_.pdf

[6] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with consistent hashing," in *WWW '99*, 1999.

[7] S. Willis and S. Helal, "Rfid information grid for blind navigation and wayfinding," *Wearable Computers*, 2005.

[8] A. Saxena, S. Ganguly, S. Bhatnagar, and R. Izmailov, "Rfind: An rfid-based system to manage virtual spaces," in *PERCOMW '07*, 2007, pp. 382–387.

[9] Y.-J. Seo, D. K. Oum, and P. Park, "Bridging the real world with the rfid phone - focus on development of innovative man machine interface and abundant wireless internet service-," *cit*, vol. 0, p. 83, 2006.

[10] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison, "Bridging physical and virtual worlds with electronic tags," in *CHI*, 1999, pp. 370–377.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Middleware*, 2001, pp. 329–350.

[12] K. C. Zatloukal and N. J. A. Harvey, "Family trees: an ordered dictionary with optimal congestion, locality, degree, and search time," in *SODA '04*, 2004, pp. 308–317.

[13] M. T. Goodrich, M. J. Nelson, and J. Z. Sun, "The rainbow skip graph: a fault-tolerant constant-degree distributed data structure," in *SODA '06*, 2006, pp. 384–393.

[14] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Physical Review E*, vol. 64, 2001.

[15] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *CACM*, vol. 13, no. 7, pp. 422–426, 1970.

[16] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Neighborhood signatures for searching p2p networks," *Database Engineering and Applications Symposium*, pp. 149–158, 2003.