RFID Data Aggregation*

Dritan Bleco and Yannis Kotidis

Department of Informatics
Athens University of Economics and Business
76 Patission Street, Athens, Greece
{dritan,kotidis}@aueb.gr

Abstract. Radio frequency identification (RFID) technology is gaining popularity for many IT related applications. Nevertheless, an immediate adoption of RFID solutions by the existing IT infrastructure is a formidable task because of the volume of data that can be collected in a large-scale deployment of RFIDs. In this paper we present algorithms for temporal and spatial aggregation of RFID data streams, as a means to reduce their volume in an application controllable manner. We propose algorithms of increased complexity that can aggregate the temporal records indicating the presence of an RFID tag using an application-defined storage upper bound. We further present complementary techniques that exploit the spatial correlations among RFID tags. Our methods detect multiple tags that are moved as a group and replace them with a surrogate group id, in order to further reduce the size of the representation. We provide an experimental study using real RFID traces and demonstrate the effectiveness of our methods.

1 Introduction

Radio frequency identification (RFID) technology has gained significant attention in the past few years. In a nutshell, RFIDs allow us to sense and identify objects. RFIDs are by no means a new technology. Its origins can be traced back to World War II, where it was deployed in order to distinguish between friendly and enemy war planes [1]. Since then, RFIDs have seamlessly infiltrated our daily activities. In many cities around the word, RFIDs are used for toll collection, in roads, subways and public buses. Airport baggage handling and patient monitoring are more examples denoting the widespread adoption of RFIDs.

With their prices already in the range of a few cents, RFID tags are becoming a viable alternative to bar codes for retail industries. Large department stores like the Metro Group and Wal-Mart are pioneers in deploying RFID tags in their supply chain [2]. Individual products, pallets and containers are increasingly tagged with RFIDs. At the same time, RFID readers, are placed at warehouse entrances, rooms and distribution hubs. These readers compute and communicate the list of RFID tags sensed in their vicinity to a central station for further processing

* This work has been supported by the Basic Research Funding Program, Athens University of Economics and Business.

and archiving. The ability to automatically identify objects, without contact, through their RFID tags, allows for a much more efficient tracking in the supply chain, thus eliminating the need for human intervention (which for instance is typically required in the case of bar codes). This removal of latency between the appearance of an object at a certain location and its identification allows us to consider new large- or global- scale monitoring infrastructures, enabling a much more efficient planning and management of resources.

Nevertheless, an immediate adoption of RFID technology by existing IT infrastructure, consisting of systems such as enterprise resource planning, manufacturing execution, or supply chain management, is a formidable task. As an example, the typical architecture of a centralized data warehouse, used by decision support applications, assumes a periodic refresh schedule [3] that contradicts the need for currency by a supply chain management solution: when a product arrives at a distribution hub, it needs to be processed as quickly as possible. Moreover, existing systems have not been designed to cope with the voluminous data feeds that can be easily generated through a wide-use of RFID technology. A pallet of a few hundred products tagged with RFIDs generates hundreds of readings every time it is located within the sensing radius of a reader. A container with several hundred pallets throws tens of thousands of such readings. Moreover, these readings are continuous: the RFID reader will continuously report all tags that it senses at every time epoch. Obviously, some form of data reduction is required in order to manage these excessive volumes of data.

Fortunately, the type of data feeds generated by RFIDs are embedded with lots of redundancy. As an example, successive observations of the same tag by a reader can be easily encoded using a time interval indicating the starting and ending time of the observation. Unfortunately, this straightforward data representation is prone to data collection errors. Existing RFID deployments, routinely drop a significant amount of the tag-readings; often as much as 30% of the observations are lost [4]. This makes the previous solution practically ineffectual as it can not limit in a application-controllable manner the number of records required in order to represent an existing RFID data stream. In this paper, we investigate data reduction methods that can reduce the size of the RFID data streams into a manageable representation that can then be fed into existing data processing and archiving infrastructures such as a data warehouse. Key to our framework is the decision to move much of the processing near the locations where RFID streams are produced. This reduces network congestion and allows for large scale deployment of the monitoring infrastructure.

Our methods exploit the inherent temporal redundancy of RFID data streams. While an RFID tag remains at a certain location, its presence is recorded multiple times by the readers nearby. Based on this observation we propose algorithms of increased complexity that can aggregate the records indicating the presence of this tag using an application-defined storage upper bound. During this process some information might be lost resulting in *false positive* or *false negative* cases of identification. Our techniques minimize the inaccuracy of the reduced representation for a target space constraint. In addition to temporal, RFID data

streams exhibit spatial correlations as well. Packaged products within a pallet are read all together when near an RFID reader. This observation can be exploited by introducing a data representation that *groups* multiple RFID readings within the same record. While this observation has already been discussed in the literature [5], to our knowledge we are the first to propose a systematic method that can automatically identify and use such spatial correlations.

The contributions of our work are

- We propose a distributed framework for managing voluminous streams of RFID data in an supply-chain management system. Our methods push the logic required for reducing the size of the streams at the so-called Edgeware, near the RFID readers, in an attempt to reduce network congestion.
- We present a lossy aggregation scheme that exploits the temporal correlations in RFID data streams. For a given space constraint, our techniques compute the optimal temporal representation of the RFID data stream that reduces the expected error of the approximate representation, compared to the full, unaggregated data stream. We also consider alternative greedy algorithms that produce a near-optimal representation, at a fraction of the time required by the optimal algorithm.
- We present complementary techniques that further exploit the spatial correlations among RFID tags. Our methods detect multiple tags that are moved as a group and replace them with a surrogate group id, in order to further reduce the size of the representation.
- We provide an experimental evaluation of our techniques and algorithms using real RFID data traces. Our experiments demonstrate the utility and effectiveness of our proposed algorithms, in reducing the volume of the RFID data, by exploiting correlations both at the time and space.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce the system architecture we consider in this work, present the details of an RFID data stream and state our optimization problem. In Section 4 we present our algorithms for temporal aggregation of the RFID streams, while in Section 5 we describe our spatial aggregation process. Our experimental evaluation is presented in Section 6. Finally, Section 7 contains concluding remarks.

2 Related Work

There have been several recent proposals discussing RFID technology. These works analyze RFID systems from different points of view, including hardware [6], software [2], data processing [7–9] and privacy [10]. The work in [11] presents a RFID system deployed inside a building where the tagged participants walking through it produce a large amount of RFID data. The authors discuss the system, its performance, showcase analysis of higher-level information inferred from raw RFID data and comment on additional challenges, such as privacy.

The main characteristics of an RFID system, such as their temporal and dynamic behavior, the inaccuracy of data, the need for integration with existing IT systems, the streaming nature of the raw data and their large volumes are discussed in [12]. The paper presents a temporal data model, DRER, which exploits the specific fundamentals of an RFID application and of the primitive RFID data. This work shows the basic features that should be included in a RFID Middleware system, such as including effective query support and automatic data acquisition and transformation. The work in [9] introduces a deferred RFID data cleaning framework of using rules executed at query time.

The work in [13] demonstrates the significance of compression in RFID systems and discusses a graph-based model that captures possible object locations and their containment relationships. However, in order to provide accurate results, the graph models require high detection rates at the RFID readers. In [4], the authors highlight the inherent unreliability of RFID data streams. Software running at the Edgeware, typically corrects for dropped readings using a temporal smoothing filter based on a pre-defined sliding window over the reader's data stream that interpolates for lost readings from each tag within the time window. The work in [4] uses a statistical sampling-based approach that results in an adaptive smoothing filter, which determines the window-size automatically based on observed readings. This work nicely complements our techniques, as it may be used at a pre-processing step in order to clean the incoming RFID data stream, before applying our aggregation algorithms.

Our temporal aggregation process works by first transforming the individual readings produced by the readers into temporal segments and then reduces the number of segments while trying to minimize the error of the approximate temporal representation. At an abstract level, this process resembles the construction of a one-dimensional histogram on the frequency distribution of a data attribute, used in traditional database management systems [14–16]. Application of existing data compression algorithms such as wavelets [17, 18], their probabilistic counterparts [19], or even algorithms developed for sensory data [20, 21] on RFID data streams is an interesting research topic.

A model for data warehousing RFID data has been proposed in [5]. This work studies the movement of products from suppliers to points of sale taking advantage of bulky object movements, of data generalization and the merge or collapse of path segment that RFID objects follow. The authors introduce a basic RFID data compression scheme, based on the observation that tags move together in any stage of the movement path. However, there is no provision for missing or erroneous data tuples. The work in [22] introduced the Flowcube, a data cube computed for a large collection of paths. The Flowcube model analyzes item flows in an RFID system. The Flowcube differs from the traditional data cube [23] in that it does not compute aggregated measurements but, instead, movement trends of each specific item. The work in [24] introduced the notion of a service provisioning data warehouse, which organizes records produced by a service delivery process (such as a delivery network). The paper introduces pair-wise aggregate queries as a means to analyze massive datasets.

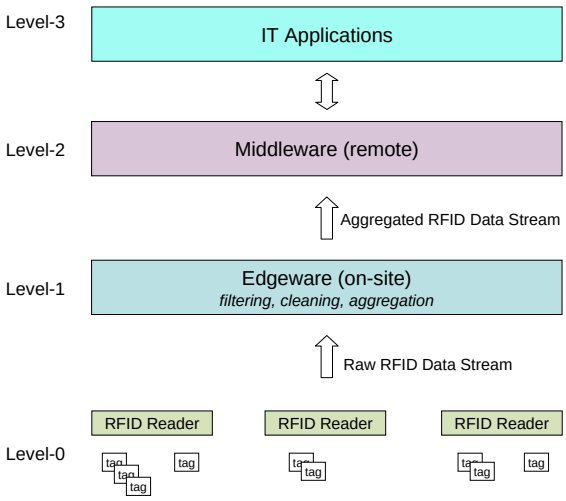

Fig. 1. System Overview

Such a query consists of a path expression over the delivery graph and a user defined aggregate function that consolidates the recorded data. Our RFID data aggregation framework can be used for the instrumentation of a large-scale ETL process while building such a data warehouse using data emanating from RFID readers along the delivery network.

3 Preliminaries

In this section we first present an overview of the architecture we assume for managing RFID data. We then discuss in more detail the contents of an RFID data stream and a simple relational mapping that exploits temporal correlations in the stream.

3.1 System Architecture

In our work we assume, but are not limited to, a layered system architecture, like the one depicted in Figure 1. The lower level of the architecture contains the hardware specific to the RFID infrastructure, which includes, at the minimum, the RFID tags and the readers. Raw RFID data streams generated by the Level-0 devices are transferred to the Edgeware (Level-1), which can be implemented, for instance, using an on-site data server. Depending on the hardware used, the Edgeware is often capable to perform data filtering, cleaning and manipulation. The Edgeware may also instantiate a local database server for buffering and managing the reported data. Finally, the processed RFID data stream is sent to

the Middleware (Level-2), whose purpose is to bridge the RFID infrastructure with the upper-level IT applications (Level-3) that rely on its data. The upper level of the architecture in Figure 1 may be further broken down in additional layers (for instance a Service layer and an Applications layer), however in the Figure we omit such details as they are not related to the problems we address with our techniques.

We note that unlike the Edgeware, which is typically implemented on-site, the Middleware may be instantiated at a central data processing center. This means that network data movement is required in order to transmit the processed RFID data streams to the Middleware and the applications on top. Furthermore, the Middleware may be responsible for multiple sites equipped with RFID infrastructure. Thus, in order to reduce the network congestion and not to overburden the servers implementing the Middleware, it is desirable that the processed RFID data streams are aggregated as much as possible.

3.2 RFID Data Description

In its simplest form a RFID tag stores a unique identifier called the Electronic Product Code (EPC). When the tag comes in proximity with a reader, the EPC code is read, using an RFID Air Interface protocol, which regulates communication between the reader and the tag. A reader is also equipped with a unique identifier, which in case of immobile readers relates to its location. Finally, an internal clock lets the reader mark the time of the observation. Thus, each time a tag is sensed a triplet of the form

$$(EPC_i, loc_i, t_i)$$

is generated, where EPC_i denotes the code of the tag, loc_i the location (or the id) of the reader and t_i the current time. It is typical in this setup to assume that the time is descritized: a reader reports tags at times t_1, t_2,... where the difference $t_{i+1} - t_i$ is called the *epoch* duration. The reader buffers multiple observations in its local memory and subsequently transmits them to the Edgeware for further processing. This data stream of triplets representing base observations is called the RFID data stream.

The data server at the Edgeware receives the RFID data streams by all readers that it manages in a continuous manner. These readings are filtered based on requirements prescribed by the applications of the upper level. For example EPC codes of locally tagged equipment that are of no interest to the supply-chain monitoring software may be dropped from the stream. A simple form of data reduction is possible at this level my merging occurrences of the same EPC in successive epochs. Let (EPC_i, loc_i, t_i), (EPC_i, loc_i, t_{i+1}),..., (EPC_i, loc_i, t_{i+m}) be part of the stream transmitted by the reader at location loc_i. A straightforward data size reduction is possible if we replace these records with a single quadruple of the form

$$(EPC_i, loc_i, t_{start}, t_{end})$$

with t_{start}=t_i and t_{end}=t_{i+m} indicating the interval containing all observations of tag EPC_i. This is a basic temporal aggregation service that helps reduce the volume of data in the network, when these readings are sent to the Middleware. Unfortunately, RFID readers routinely drop a significant amount of tag-readings, especially when a large number of tags are concurrently present within the sensing radius of the reader. Moreover, as items are moving withing the facility, the same RFID tag may appear in multiple time intervals in the stream produced by a reader. Both observations complicate management of the RFID data stream and limit the effectiveness of the basic temporal aggregation service.

In our work, we extend the format of the basic tuple generated at the Edgeware to also include a fifth attribute p indicating the percentage of epochs that a tag was observed withing the time interval $[t_{start}, t_{end}]$. A value of p equal to 1 indicates that the tag was spotted during all epochs between t_{start} and t_{end}. This case is equivalent to the format used in basic temporal aggregation. However, a value of p lesser than 1 indicates that only $p \times (t_{end} - t_{start} + 1)$ epochs within the time interval contain observations of the tag. This extension, thus, allows us to represent multiple occurrences of the same tag using fewer intervals. Of course using a single interval to describe all observations of the tag results in large inaccuracy, especially when there are many "holes", i.e. time intervals when the tag was never spotted, in the stream. Ideally, given an upper bound on the number of records that can be produced to describe the tag, one would like to find an allocation of intervals that best describe the presence of the tag at the reader.

3.3 Problem Formulation

We can now state our optimization problem formally:

Problem Statement: Given a data stream containing observations of EPC_i at epochs $t_{i_1} < t_{i_2} < \ldots < t_{i_n}$ find the best B-tuple representation $(EPC_i, loc_i, t_{s_1}, t_{e_1}, p_1)$, $\ldots$, $(EPC_i, loc_i, t_{s_B}, t_{e_B}, p_B)$ where

- $[t_{s_k}, t_{e_k}]$ and $[t_{s_l}, t_{e_l}]$ are non-overlapping intervals $(1 \leq k \neq l \leq B)$,
- X=$\{t_{i_1}, \ldots t_{i_n}\}$, Y=$\{t \in [t_{i_1}, t_{i_n}] | t \notin X\}$, $X \subseteq \cup_k [t_{s_k}, t_{e_k}]$
- p_k=$\frac{|X \cap [t_{s_k}, t_{e_k}]|}{t_{e_k} - t_{s_k} + 1}$

and the cumulative error of the representation

$$\sum_{t \in X} err_x(t) + \sum_{t \in Y} err_y(t)$$

is minimized. Where

$$err_x(t) = \begin{cases} 1 - p_j & , \exists j \text{ such that } t \in [t_{s_j}, t_{e_j}] \\ 1 & , \text{otherwise} \end{cases}$$

and,

$$err_y(t) = \begin{cases} p_j & , \exists j \text{ such that } t \in [t_{s_j}, t_{e_j}] \\ 0 & , \text{otherwise} \end{cases}$$

In this formulation X contains the set of epochs when the tag was spotted by the reader and Y the set of epochs (between its first and last observation) when the tag was not reported. If we use an interval $[t_s, t_e]$ the value of p is determined by the fraction of epochs in X that belong in $[t_s, t_e]$ over the size of the interval. Then the error in estimating the presence of the tag at an epoch t within the interval is (1-p), in case the tag was spotted, and, p in case the tag was not spotted by the reader, respectively. In the formulation of the error function $err_x()$ denotes the false negative error rate when a tag is spotted but we report a value of p less than 1. Similarly, $err_y()$ denotes the false positive error rate when the tag was not spotted by the reader but the epoch in question is inside the interval we report. Thus, our formulation takes into account both false positive and false negative reports of a tag in the computed representation. Given an initial RFID data stream we would like to compute the best representation, using only B tuples, that minimizes the aforementioned error. Obviously, the error is zero when we use as many intervals as the number of epochs in set X. Depending on the distribution of epochs within X, we may be able to derive a much smaller representation with small cumulative error.

In what follows we will discuss algorithms of increased complexity for solving this problem. We first present two observations that help limit the search space in finding the optimal set of tuples.

Lemma 1. *If tuple* $(EPC_i, loc_i, t_{s_k}, t_{e_k}, p_k)$ *belongs in the optimal* B*-tuple representation then* $t_{s_k}, t_{e_k} \in X$.

Lemma 2. *If tuple* $(EPC_i, loc_i, t_{s_k}, t_{e_k}, p_k)$ *belongs in the optimal* B*-tuple representation then* $t_{s_k} - 1, t_{e_k} + 1 \notin X$.

Lemma 1 states that we should only consider intervals where the starting and ending points both belong to set X. It is easy to see that if one or both end-points do not satisfy this condition, we can always compute a better representation by increasing t_{s_k} (resp. decreasing t_{e_k}) to the nearest epoch when the tag was spotted, as this always reduces the error. Lemma 2 states that when consecutive observations of a tag exist, it is always desirable to package them within the same interval.

4 Temporal RFID Data Aggregation

Recall that given a series of observations of EPC_i at a location, we would like to compute a B-tuple representation of the form $(EPC_i, loc_i, t_{s_1}, t_{e_1}, p_1)$, ..., $(EPC_i, loc_i, t_{s_B}, t_{e_B}, p_B)$. Due to Lemmas 1 and 2, the computation can be performed on the data stream produced after we apply the basic temporal aggregation service. Thus, if we ignore the EPC_i and loc_i values that are constant in this discussion, we are given a set of non-overlapping intervals $[t_{s_1}, t_{e_1}]$, ..., $[t_{s_n}, t_{e_n}]$ and would like to replace them with $B << n$ intervals $[t'_{s_k}, t'_{e_k}]$, $1 \leq k \leq B$ such that each input time interval is contained within exactly one of the output intervals. Given the definition of the error we presented in the previous section, we

can derive an analytical formula for computing the error associated with a candidate interval $T'=[t'_{s_k}, t'_{e_k}]$ as follows. Let $X(T')$ denote the number of epochs that the tag was reported within T'. Similarly, let $Y(T')$ denote the number of epochs in T', during which the tag was not reported by the reader. Then, the error induced by T' is ($p=\frac{X(T')}{X(T')+Y(T')}$)

$$error(T') = \frac{(1-p) \times X(T') + p \times Y(T')}{X(T') + Y(T')} = 2 \times \frac{X(T') \times Y(T')}{(X(T') + Y(T'))^2} \quad (1)$$

4.1 Sub-optimal Algorithms

A straightforward way to obtain a B-tuple representation from the initial n intervals if to first order them by their starting times t_{e_i} and then group them in B batches containing $\lceil \frac{n}{B} \rceil$ input intervals each, except possibly the last. For each batch we generate one interval T' with starting point the starting time of the first input interval in the batch and ending point the ending of the last interval in the batch. The complexity of this simple algorithm is $O(n)$ (linear), assuming that the input n intervals are already ordered by their timestamps. This is a valid assumption, since the basic temporal aggregation process that generates the input intervals operates by first ordering the incoming RFID data stream based on the timestamps of the observations.

The Linear algorithm merges consecutive input intervals, in an error-oblivious manner. A better approximation can be obtained as follows. Given n input intervals, we can consider merging each of the n-1 consecutive pairs in the input. Each candidate pair results in a new interval T' for which we can compute the error using equation 1. A greedy strategy can then be applied that selects the best such interval T' and replaces the corresponding two input intervals with T'. This reduces the number of input intervals by one, to n-1. The same process is then repeated until we are left with B intervals. We call this algorithm the Greedy algorithm. The complexity of the Greedy algorithm is $O((n-B) \times n)$.

4.2 An Optimal Dynamic Programming Algorithm

We now describe a algorithm based on dynamic programing that computes the best B intervals (equivalently best B-tuple representation) that minimize the error of the approximation. Recall that our input consists of n time-intervals that we would like to organize into B non-overlapping intervals, each containing one or more of the original ones. In what follows we would refer to the output intervals that the algorithm considers as *buckets* in order to distinguish them from the input ones. Based on Equation 1, we observe that the error produced by incorporating one or more input intervals within a bucket is independent of the assignments that we have made for other buckets. This observations allows us to state the computation problem using a dynamic programming formulation where the optimal result of a (sub)problem can be obtained by dividing it into two more subproblems and combining the optimal solutions to those subproblems.

In particular, let $E(i,k)$ denote the cumulative error of the representation that considers the best way to generate up to k buckets out of the first i input intervals. Obviously $E(i,k)$=0, for $i \leq k$. We can now compute $E(i,k)$ using the following recursion.

$$E(i,k) = \min_{j<i}(E(j,k-1) + err(j+1,i)) \quad (2)$$

$err(j+1,i)$ in this formula denotes the error of using a single bucket for merging all input intervals from $j+1$ up to i. This error is computed by equation 1 by setting T'=$[t_{s_{j+1}}, t_{e_i}]$. Informally, the dynamic programming setup calculates the best way to generate k buckets for the first i intervals by considering the best way to compute k-1 buckets for up to the j-th interval and using a single bucket for intervals $j+1$,$j+2$,...,i. The error of the optimal assignment is calculated by $E(n,B)$ and the optimal bucket configuration arises easily by backtracking the selections made at each step. The running time complexity of the algorithm is $O(n^2 \times B)$.

5 Spatial RFID Data Aggregation

Items tagged with RFIDs are typically moved in groups. For example, packaged products within a pallet are read all together when near an RFID reader. This observation can be exploited by introducing a data representation that *groups* multiple RFID readings within the same temporal record. Given the RFID data stream produced after we apply the temporal aggregation described in the previous Section, we now seek to exploit spatial correlations in its records. In particular, we order the incoming tuples based on their t_{s_k},t_{e_k} timestamps. Tuples with the same starting and ending timestamps can be encoded using the single common interval by creating a EPC *group-id*. This is a system generated unique code that we will using in order to refer to all EPC_i codes that have been identified during the same time interval. These group-ids are kept in a separate relational table at the Edgeware. This way, they may get possibly reused in order to refer to this subset of codes, when they participate at a larger group in other parts of the data stream.

As an example, consider the records of Table 1 produced after we apply the temporal aggregation process. We can observe that EPC codes I1 and I2 are observed simultaneously at location L1 between T1 and T5. Thus, we can generate a group-id G1 to refer to both products. When these products are later spotted at location $L2$ along with product I4, a new group G2 is generated, which contains both G1 and I4. Table 2 depicts the final set of produced tuples. Table 3 describes the assignment of product codes to group. We note that groups may include other groups (as is the case of G2 and G1). This information needs also be transmitted along with the spatially aggregated tuples in order to decode the groups at the Middleware.

An additional point that we need to clarify when we aggregate multiple records in a group is how to generate a new p-value for the composite record.

EPC	loc	t_s	t_e	p
I1	L1	T1	T5	78
I2	L1	T1	T5	69
I3	L1	T2	T5	90
I1	L2	T12	T22	67
I2	L2	T12	T22	62
I4	L2	T12	T22	66

Table 1. Input RFID Data Stream

EPC	loc	t_s	t_e	p
G1	L1	T1	T5	69
I3	L1	T2	T5	90
G2	L2	T12	T22	62

Table 2. Reduced RFID Data Stream

Group-id	EPC list
G1	I1,I2
G2	G1,I4

Table 3. Map Table

Recall that the p-values in the temporally aggregated RFID data stream indicate the percentage of epochs that the tag was spotted during the interval indicated in the record. When we aggregate several records we have different options in order to produce a new p-value for the group.

- If we seek to reduce the false negative rate of the representation, the p-value of the group should be the minimum of the p-values of all tuples composing the group.
- If we seek to reduce the false positive rate of the representation, we should keep the maximum value of p among the group.
- If we seek to reduce both the false negative and the false positive rates, then we can keep the average value.

In our running example, we used the minimum value, indicating that the application is more concerned about false negative identifications of a tag.

The basic spatial aggregation scheme can be extended in two ways. First, when we compose multiple records, we can set an upper bound on the false positive/negative error rate that we introduce. This may lead to fewer opportunities for spatial aggregation but with reduced error. Another possible extension is to allow an item to partially participate in a group. For example item I3 in Table 1 is spotted with items I1 and I2 in all but the first epoch (T1) at location L1. We could, thus, consider including all three items in a single group for this location.

6 Experiments

In this Section we provide an evaluation of our temporal and spatial aggregation schemes. All algorithms were implemented using Visual Studio 2005. The

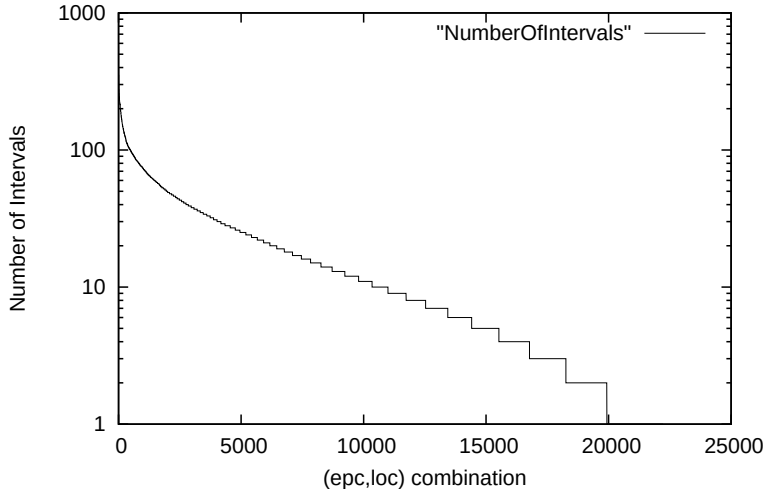

Fig. 2. Number of intervals for each (epc,loc) combination after application of the basic temporal aggregation

reported times are on a Intel Core Duo CPU running at 1.83GHz with 1GB of memory. We used as input dataset, the publicly available trace of RFID data obtained during the 2008 Hope Conference in New York, sampled at 30sec intervals. The trace contains 1.9M records of the form (EPC_i, loc_i, t_i). At a first step, we applied the basic temporal aggregation service that identifies continuous tag observations in order to create tuples of the form (EPC_i, loc_i, t_s, t_e). This process reduced the number of tuples to 423K. The aggregated dataset contained 22,245 unique pairs of (EPC_i, loc_i) values identified at different time intervals.

In Figure 2 we plot the distribution of the number of intervals generated for every pair in the dataset. The higher this number, the more opportunities we have to further reduce the size of the data representation for the particular pair, using our lossy temporal aggregation scheme. We note that, in this dataset, about 10% of the observations are reported in a single interval, which can not be further reduced at the time domain. For the remaining combinations though, we can trade accuracy for compactness in their temporal representation.

In Figure 3 we plot the execution times of the three proposed algorithms: Linear, Greedy and OptimalDP using as input the (tag,loc) combination with the greatest number of intervals (569) from the previous step. We executed the algorithms varying the number of output intervals B requested. Each output interval corresponds to one tuple in the processed RFID data stream. As expected the execution time of Linear is the same independent of the value of B. Greedy works bottom up in constructing the requested number of intervals and its execution time decreases with B. The OptimalDP runs faster when a larger

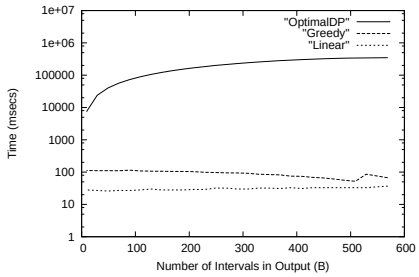

Fig. 3. Execution times

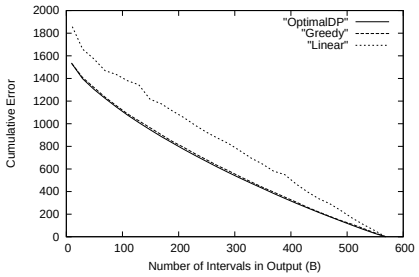

Fig. 4. Error of each algorithm

degree of aggregation (smaller value of B is requested). We note that Greedy is up to three orders of magnitude faster than OptimalDP for the same value of B.

In Figure 4 we compare the cumulative error (see Equation 1) computed for the output intervals of each algorithm, when we vary B. As expected, the error of all representations drops when more intervals are used to describe the presence of the tag. We also note, that Greedy computes in most of the cases a representation that is practically the same as optimal. However, this is achieved at a fraction of the time that the dynamic programming algorithm requires, as is evident by Figure 3. We observed this near-optimal behavior of Greedy in all other combinations of tags and locations for this dataset.

We also tested the effect of grouping RFID tags that are moved together over periods of time. We started with the stream resulting from the basic temporal aggregation, which consisted of 423K tuples. Our analysis identified 77K groups of items appearing in identical time intervals. The overall space reduction, which also accounts for the space required for the surrogate group-id descriptions, was 39%. The running time of the spatial aggregation process was 3.3secs. We note that this process did not introduce any error as the original stream can be reverse-engineered by replacing the surrogate group-ids with the corresponding EPCs. The spatial aggregation process can be combined with any of the three temporal aggregation algorithms. In Figure 5 we present the results of an experiment were we tried different methods for reducing the size of the RFID data stream. In the graph we depict (1) the size of the initial raw RFID stream, (2) the size of the stream after the basic temporal aggregation, (3) the resulting stream of the basic temporal aggregation followed by the spatial aggregation process, (4) the resulting stream after applying the Greedy algorithm in the initial dataset in order to reduce the number of tuples for each (epc,loc) combination by a factor of 3:1 (for those combinations with at least three intervals) and, (5) the resulting stream when Greedy is combined with the spatial aggregation process. The first 3 schemes are lossless, while in (4) and (5) some error is introduced because of the Greedy algorithm. Of course, one may further reduce the stream size by choosing even fewer output intervals while executing the Greedy algorithm.

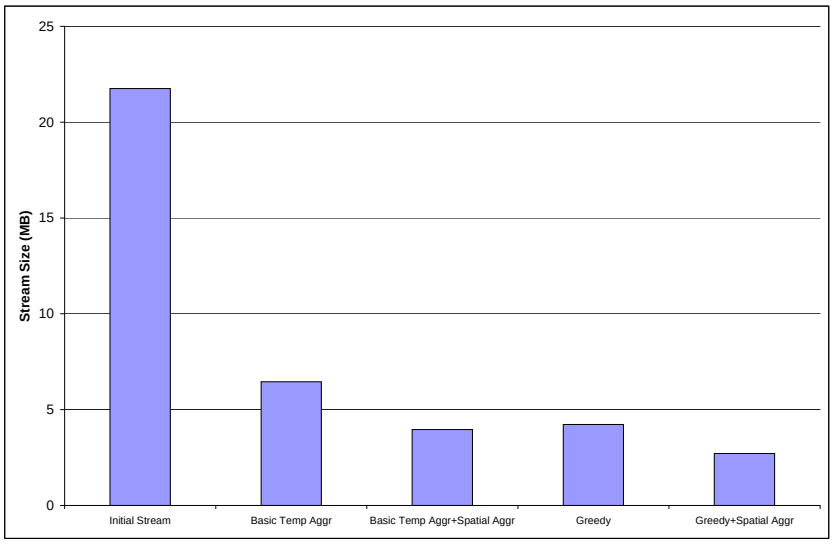

Fig. 5. Combination of Temporal and Spatial Aggregation

7 Conclusions

The increased adaptation of RFID technology promises to deliver massive datasets. These datasets need to be tamed by reducing their volumes in an application-controllable manner. In this paper we presented several algorithms for temporal and spatial aggregation of RFID data. Our algorithm can reduce the volume of their input data by exploiting correlations at the time and space (location) dimension that characterize the identification of an RFID tag. We provided an experimental study using real RFID traces and demonstrated the effectiveness of our methods.

References

1. Stockman, H.: Communication by Means of Reflected Power. In: IRE. (Oct. 1948)
2. Chawathe, S., Krishnamurthy, V., Ramachandran, S., Sarma, S.: Managing RFID Data. In: Proceedings of VLDB. (2004) 1189–1195
3. Kotidis, Y., Roussopoulos, N.: A Case for Dynamic View Management. ACM Transactions on Database Systems (TODS) **26**(4) (2001) 388–423
4. Jeffery, S., Garofalakis, M., Franklin, M.: Adaptive Cleaning for RFID Data Streams. In: Proceedings of VLDB. (2006)

5. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and Analyzing Massive RFID Data Sets. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE). (2006) 83
6. Finkenzeller, K., Waddington, R., eds.: RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. Wiley, John & Sons, Incorporated (2003)
7. Krompass, S., Aulbach, S., Kemper, A.: Data Staging for OLAP- and OLTP-Applications on RFID Data. In: BTW. (2007) 542–561
8. Park, J., Hong, B., Ban, C.: A Continuous Query Index for Processing Queries on RFID Data Stream. In: 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). (2007) 138–145
9. Rao, J., Doraiswamy, S., Thakkar, H., Colby, L.S.: A Deferred Cleansing Method for RFID Data Analytics. In: Proceedings of the 32nd international conference on Very large data bases (VLDB). (2006) 175–186
10. Sarma, S., Weis, S.A., Engels, D.W.: RFID Systems and Security and Privacy Implications. In: CHES '02. (2003) 454–469
11. Welbourne, E., Koscher, K., Soroush, E., Balazinska, M., Borriello, G.: Longitudinal Study of a Building-wide RFID Ecosystem. In: Mobisys. (2009)
12. Wang, F., Liu, P.: Temporal Management of RFID Data. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB). (2005) 1128–1139
13. Cocci, R., Tran, T., Diao, Y., Shenoy, P.J.: Efficient Data Interpretation and Compression over RFID Streams. In: Proceedings of the 24th International Conference on Data Engineering (ICDE). (2008) 1445–1447
14. Ioannidis, Y.E.: The History of Histograms (abridged). In: VLDB. (2003) 19–30
15. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Optimal and Approximate Computation of Summary Statistics for Range Aggregates. In: PODS. (2001)
16. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal Histograms with Quality Guarantees. In: Proceedings of 24rd International Conference on Very Large Data Bases (VLDB). (1998) 275–286
17. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: One-Pass Wavelet Decompositions of Data Streams. IEEE Trans. Knowl. Data Eng. **15**(3) (2003) 541–554
18. Sacharidis, D., Deligiannakis, A., Sellis, T.K.: Hierarchically Compressed Wavelet Synopses. VLDB J. **18**(1) (2009) 203–231
19. Cormode, G., Garofalakis, M.N.: Histograms and Wavelets on Probabilistic Data. In: ICDE. (2009) 293–304
20. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Dissemination of Compressed Historical Information in Sensor Networks. VLDB J. **16**(4) (2007) 439–461
21. Guitton, A., Trigoni, N., Helmer, S.: Fault-Tolerant Compression Algorithms for Delay-Sensitive Sensor Networks with Unreliable Links. In: DCOSS. (2008) 190–203
22. Gonzalez, H., Han, J., Li, X.: Flowcube: Constructuing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows. In: roceedings of the 32nd International Conference on Very Large Data Bases (VLDB). (2006) 834–845
23. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: ICDE. (1996) 152–159
24. Kotidis, Y.: Extending the Data Warehouse for Service Provisioning Data. Data Knowledge. Engineering **59**(3) (2006) 700–724