Efficient Data Interpretation and Compression over RFID Streams

Richard Cocci, **Thanh Tran**, **Yanlei Diao** and **Prashant Shenoy**
Department of Computer Science, University of Massachusetts Amherst
{rcocci, ttran, yanlei, shenoy}@cs.umass.edu

***Abstract*— Despite its promise, RFID technology presents numerous challenges, including incomplete data, lack of location and containment information, and very high volumes. In this work, we present a novel data interpretation and compression substrate over RFID streams to address these challenges in enterprise supply-chain environments. Our results show that our inference techniques provide good accuracy while retaining efficiency, and our compression algorithm yields significant reduction in data volume.**

I. Introduction

RFID is a promising electronic identification technology that enables a real-time information infrastructure to provide timely, high-value content to monitoring and tracking applications. However, RFID data—a triplet <tag id, reader id, timestamp> in its most basic form—raises new challenges since it may be insufficient, incomplete, and voluminous.

Insufficient information: Since RFID is inherently an identification technology designed to identify individual objects, a stream of RFID readings does not capture inter-object relationships such as co-location and containment.

Incomplete data: Despite technological advances, RFID readings are inherently noisy with observed read rates below 100% in actual deployments. Missed readings result in a lack of information about an object's location, significantly impairing the tasks of determining both location and containment.

High volume streams: Perhaps the key distinguishing characteristic of RFID streams are their high data volumes which can easily overwhelm a data stream system. Hence, it is imperative that data be filtered and compressed close to the hardware while preserving all useful information.

In this paper, we present SPIRE, a system that addresses the above challenges. SPIRE departs from the prior work by building an *interpretation and compression* substrate over RFID data streams which employs three key techniques: (1) a time-varying graph model that captures possible object locations and containment relationships with its stream-driven construction, (2) a probabilistic algorithm that infers the most-likely locations and containment relationships of objects, and (3) an output algorithm that transforms an input stream to a compressed, yet informative output stream. We have implemented our interpretation and compression substrate and have evaluated it using synthetic RFID streams emulating an enterprise supply-chain environment. Our results show that our inference techniques provide good accuracy while retaining efficiency, and our compression algorithm yields significant reduction in data volume.

II. Problem Statement

Before defining the problem, we present the notion of the physical world. A *physical world* covers a geographical area

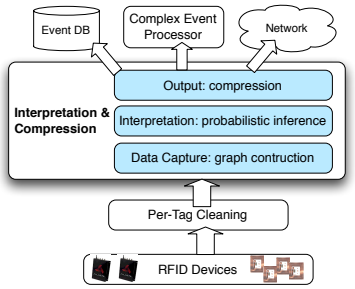

Fig. 1. Architecture of SPIRE: Our substrate consists of (i) a *data capture* module for stream-driven construction of a time-varying graph model encoding possible object locations and containments; (ii) an *interpretation* module to probabilistically infer the most likely location and containment for an object, and (iii) a *compression* module that outputs stream data in an compressed format with smoothing.

comprising a set of objects O, a set of pre-defined, fixed locations L, and an ordered discrete time domain T. The set of locations can be either pre-defined logical areas such as aisle 1 in warehouse A, or (x, y, z) coordinates generated by a positioning system. At time instant t, the *state of the world* includes a set of location relationships, where each $o_i \in O$ is present at some $l_k \in L$. Additionally, there exists containment relationships between objects o_i, $o_j \in O$ at some $l_k \in L$. Note that containment is dependent on the fact that both the container and contained object are present at the same location.

The state of the world changes whenever an object enters the world, an object exits the world through a designated channel, or an existing object changes its location or containment relationship with other objects. The set of locations L also contains a special location called *unknown*. In particular, an object is in the "unknown" location if it is not present in any pre-defined location (e.g., in transit between two locations) or if it exited the physical world improperly (e.g., was stolen).

RFID readers provide a means to observe the physical world. The readings produced at time t are collectively called an *observation of the world*. In this work, we focus on readers mounted at fixed locations—a common configuration in today's RFID deployments. For such fixed readers, a reading captures the location of the object (which is the same as the location of the reader). Such readings, however, are inadequate for capturing the containment between objects.

The **data interpretation** problem is to construct an approximate yet accurate estimate of the state of the world based on the observations thus far. For a given object, we provide probabilistic values representing its most likely location and container. Recent research on RFID data cleaning [1][2] has employed temporal and spatial smoothing to alleviate missed

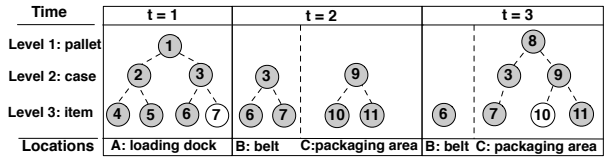

Fig. 2. A sequence of observations in a warehouse.

readings, but does not capture inter-object relationships or provide location estimates. Note that in our definition, data interpretation over streams is only concerned about the present state of the physical world and not the past or the future.

The **data compression** problem is to transform the input stream into an output stream with a reduced data volume but with no loss of information. Such compression requires the knowledge of what data is redundant and thus can be safely discarded. In this work, we use interpretation to obtain such knowledge and generate an output stream that (i) augments the input stream with additional, likely information about objects, and (ii) has a significantly reduced volume of data.

A warehouse scenario is depicted in Figure 2, where RFID readers are installed above the loading dock, the conveyor belt, and the packaging area. At time t=1, the loading dock reader reports objects 1 to 6, denoted by the shaded nodes. These nodes are arranged according to the packaging levels that the reported tag ids indicate [3]. Object 7 is also present but was missed by the reader, denoted by an unshaded node, i.e. a *missed reading*. Containment between objects, depicted by the dashed edges, is not reported by the readings and often uncertain. Examples of *ambiguous containment* are the containment between items 4, 5, 6 and cases 2, 3 at this time.

At time t=2, case 3 is scanned individually on the belt. It is possible to confirm the containment between the case and its item(s) now if the domain knowledge of the deployment reveals such *special readers* that scan containers of a particular type one at a time. Additionally, a new case 9 is read in the packaging area. At time t=3, item 6 is read at the belt again (it fell off its case at t=2 and stayed here). A new pallet 8 is assembled from cases 3 and 9 in the packaging area. Item 10 remained with its case but was not read, creating the inaccurate appearance that the item is missing.

III. Data Capture

This section describes our data capture technique to construct a time-varying graph model from the raw stream.

A Time-Varying Colored Graph Model: Our graph model $G = (V, E)$ encodes the current view of the objects in the physical world, including their reported locations and (unreported) possible containment relationships. In addition, the model incorporates statistical history about co-occurrences between objects. Example graphs for the observations in Figure 2 are shown in Figure 3.

The node set V denotes all RFID-tagged objects in the physical world. Since we are assuming a supply-chain environment, an object has a packaging level of an *item*, *case* or a *pallet*; the packaging level is encoded in the RFID tag ID [3]. Our graph is arranged into layers, with one layer for each packaging level. Each node either has a color that denotes its location or is uncolored if its location is currently unknown. The node colors are updated for the stream of readings in each epoch using the color of the location where each tag is observed. If an object is not read by any reader in a particular epoch, its node becomes uncolored. However, uncolored nodes retain memory of their most recent color and the observation time denoted by (*recent_color*, *seen_at*).

The directed edge set E encodes possible containment relationships between objects. A directed edge $o_i \rightarrow o_j$ denotes that o_i contains object o_j (e.g., case i contains item j). We allow multiple outgoing and incoming edges to and from each node, indicating an object such as a case may contain multiple items, and conversely, an item may have multiple potential cases (our probabilistic inference will subsequently chose only one of these possibilities). We allow combinations of colored and uncolored nodes, with the exception that an edge cannot connect two nodes of different colors; that is, containment is prohibited for two objects resident in two different locations.

To enable inference, the graph also encodes additional statistics. Each edge maintains a bit-vector *recent_collocations* to record recent positive and negative evidence for the collocation of the two objects. The bit is set every time the two nodes connected by the edge are assigned the same color. Further, each node maintains *past_certain_parent* statistics to remember the last confirmed parent, either revealed by a special reader or through inference with high certainty, the time of confirmation, and the number of conflicting observations obtained thus far. Among all incoming edges, only one can be a confirmed edge, denoted by the edges with double arrows in Figure 3.

We assume that time is divided into epochs and the graph is updated using stream data from each epoch. Our construction algorithm takes the graph G from the previous epoch and a set of readings R_k from each reader k ($1 \leq k \leq K$) in the current epoch, and produces a new graph G^*. An important feature of the algorithm is that it proceeds incrementally as readings arrive from each reader, and guarantees a consistent output G^* after seeing the readings from all readers in an epoch. This ensures that the algorithm works even when the various readers are coarsely synchronized in time. Given R_k of each reader, the graph update procedure entails four steps.

Step 1. *Update and color nodes:* If a new object is observed for the first time, a new node is created in the graph. For each observed object, the corresponding node is colored with the color of the reader that observed it. The colors of unobserved objects are not updated but simplify fade at a certain rate.

Step 2. *Update edges:* If two nodes in adjacent layers have the same color, an edge is added between them if one does not exist. This enumerates all possible containment relationships (e.g., a blue item can be contained in any of the blue cases that are present on a shelf). t=1 in Figure 3 demonstrates edges being constructed for the arriving pallet.

Step 3. *Prune graph:* An edge is removed from the graph if its vertices are assigned different colors, which may occur when two previously co-located objects are now reported in different locations. t=3 in Figure 3 shows the edge $v_3 \rightarrow v_6$ being pruned due to conflicting node colors. Alternatively, edges can be removed from an object when a single edge

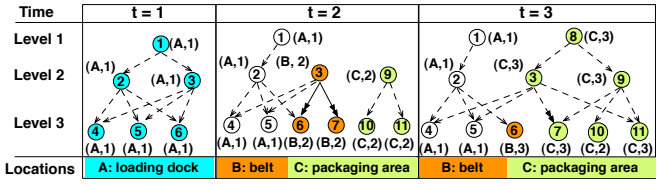

Fig. 3. Examples of the time-varying colored graph model.

is confirmed as the object's container. Special readers, such as a belt reader that read exactly one pallet at a time, allow for improvement in the accuracy of the graph model while simultaneously helping to prune unnecessary edges.

Step 4. *Update Statistics:* This step updates statistics of the edges that have at least one node colored in step 1. Given an edge e, if the two linked nodes have the same color, *recent_collocations* of e is updated by setting the most recent bit to True. Furthermore, if the reader k is able to confirm the containment denoted by e, we update the *past_certain_parent* of the child node. If one of the linked nodes is uncolored, the most recent bit of *recent_collocations* is set to False. In this case, we also check if e was set as the certain parent edge of the child node, and if so count the current observation as a conflicting observation of the confirmation. These statistics play a key role in containment inference.

IV. Data Interpretation

The graph constructed from the data capture step can result in uncolored nodes or nodes with multiple parent nodes. The data interpretation step attempts to estimate the most likely location of an unreported (uncolored) object and the most likely container (parent) of an (either reported or unreported) object using a probabilistic inference technique.

Edge Inference: Edge inference is applied to each incoming edge of a node v regardless of whether the node is colored or not. It assigns a probability p_{e_i} to each edge; the edge with the highest probability is then chosen as the most likely container of v. Past history is used to compute probability values, which includes (i) the recent history of co-locations, as represented by the bit-vector *recent_collocations* and (ii) the last confirmation of an edge either by a special reader or as a result of inference with high certainty, as captured in the data structure *past_certain_parent*. The use of past history makes edge inference less sensitive to missed readings.

Edge inference at a node consists of two steps. The first step computes a weight w_{e_i} for each incoming edge using the history of observed co-locations. The next step builds a probability distribution across all incoming edges. It computes a probability p_{e_i} for each edge by balancing the relative weight on this edge against the edge's last confirmed parent.

Node Inference: Node inference is applied to an uncolored node v—an object with an unknown location—and attempts to infer the most likely location of the object or confirm its absence from any known location. The key challenge in node inference arises from a three-way tradeoff among *object dynamics*, *continuation of past state*, and *absence with no other knowledge*. Specifically, a given object can remain in its current location, move to a new observable location, or disappear from all observable locations.

To account for these possibilities, node inference builds a probabilistic distribution over all possible colors of a node v, including (1) its most recent color, (2) the colors of its neighboring nodes that can be propagated through the edges, and (3) a special color "unknown". Among all possible colors, the one with the highest probability represents the most likely estimate of this object's location. In t=3 in Figure 3, v_{10}'s location was inferred due to color inherited from its neighbors.

V. Stream Output with Compression

The output module takes the results of inference, i.e. the most likely estimates of the location and container of each object, and transforms them into a compressed output event stream. The key idea behind compression is that only those readings that indicate a *state change* need to be included in the output stream. The state of an object is said to have changed if its location or its containment changes. In the absence of a state change, all readings merely confirm the current state of the physical world and can be safely discarded.

Our *Location compression* works on the intuition that if an object is stationary—resident at the same location for a period of time—only an initial event needs to be output to indicate its first arrival at this location and all subsequent readings in the same location can be safely discarded. Separately, *containment compression* exploits stable containment, for both stationary and mobile objects, based on the intuition that for the extent of a containment relationship it is possible to infer all of the child's events through the parent. Location and containment compression techniques have been previously proposed for RFID warehouses [4] but use expensive disk-based operations such as sorting and summarization, as opposed to our methods which are processed on the fly and modify the output stream.

VI. Summary of Performance Evaluation Results

Overall, our results show that the our framework for inference is able to correctly infer both object location and containment with a higher than 90% accuracy when the reader's read rate is above 80%, a level reached in many current RFID deployments. Our compression techniques are also capable of reducing total data volume by more than 80% when compared to the raw reader output. Furthermore, our framework is capable of scaling to simulated workloads in excess of 100,000 objects while still maintaining processing throughput above stream speed.

Acknowledgments. The work has been supported in part by the National Science Foundation under the grant CNS-052072.

References

[1] M. J. Franklin, S. R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. W. 0002, O. Cooper, A. Edakkunni, and W. Hong, "Design considerations for high fan-in systems: The HiFi approach." in *CIDR*, 2005, pp. 290–304.

[2] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for rfid data streams." in *VLDB*, 2006, pp. 163–174.

[3] "EPCglobal tag data standards version 1.3." http://www.epcglobalinc.org/, Mar 2006.

[4] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and analyzing massive RFID data sets." in *ICDE*, 2006, p. 83.