Data Architectures for RFID Transactions

Suresh Chalasani, *Senior Member, IEEE*, and Rajendra V. Boppana, *Senior Member, IEEE*

***Abstract*—We focus on the data models for storing the data generated by radio frequency identification (RFID) transactions and architectures for processing such data. We consider the supply chain comprised of the manufacturer, distributor, retailer, and the consumer. We discuss details of the data generated by RFID transactions and data models to store such data. Different organizations in the supply chain may use this data for different applications such as automatic product ordering, shelf replenishment, and product recall. We present models to anticipate the data requirements generated by RFID transactions and indicate how existing enterprise applications can be adapted to handle RFID data. The results presented in this paper will help a practitioner to 1) design and develop databases and applications for handling RFID data and 2) significantly reduce the storage requirements of RFID data. Using the data architectures, we discuss two supply chain applications—product recall and shelf replenishment—in detail. We present analytical models for the cost and time required for shelf replenishment in a retail store.**

***Index Terms*—Data models, database architectures, electronic product code (EPC), enterprise resource planning (ERP) systems, information storage, supply chain information systems.**

I. Introduction

RADIO frequency identification (RFID) uses wireless technology to identify objects—for example, products in a supply chain, wild animals that need to be tracked, and so on—from a distance, without requiring line of sight [2]. In the context of supply chain, it may be considered as a wireless variant of optical scanning of product barcodes with some important differences. Instead of optical barcodes, RFID tags containing 96- to 128-bit electronic product codes (EPCs) are attached to products. The tags, which can be battery-powered active tags or battery-less passive tags, can be read automatically by RFID *tag readers*, which can send the scanned tag information to a host computer system for processing and storage. In contrast to the optical barcode technology, scanning of product inventory can be automated with RFID technology.

The potential advantages of RFID technology in the supply chain are numerous [2], [5], [11]. RFID technology has the ability to provide up-to-the-minute information on sales of items and thus can give an accurate picture of the inventory levels. This accuracy may lead to reduction in inventory levels, thus causing a reduction in inventory costs. RFID technology at the pallet level has the potential to automate the distribution of goods in the supply chain between manufacturing plants, warehouses, and retail stores of different organizations. Reading RFID tags on a continuous basis allows companies to identify all items, thus cutting down losses from lost/misplaced inventory. Several organizations including Wal-Mart and Proctor & Gamble (P&G) are currently testing and deploying RFID technology in their supply chains. In addition, the Department of Defense has mandated its suppliers to use RFID tags at the pallet level.

In a retail store, RFID tag information is generated based on events such as a product leaving a shelf or a product being checked-out by a customer at a (perhaps automatic) checkout counter. Such events or activities generate messages for the host system (a.k.a. central transaction server). The host system, when it processes these messages, in turn may generate messages for other partners in the supply chain. The host system may send some of the RFID transaction data to the enterprise system of the retailer.

In this paper, we focus on the data models for storing the data generated by RFID transactions and architectures for processing such data. We consider the supply chain comprised of the manufacturer, retailer, distributor, and the consumer. We discuss the events that trigger RFID transactions and the corresponding data generated. We present data models to store such data. Different players in the supply chain may use this data for different applications such as automatic product ordering, shelf replenishment, and product recall.

We present a model to anticipate the data requirements generated by RFID transactions. In the past decade, many companies have invested billions of dollars in enterprise resource planning (ERP) systems. We indicate how existing ERP systems can be extended to handle RFID data and applications. We present two applications of the RFID data models: product recall and faster shelf replenishment.

II. Literature Survey

Applications of RFID technology for many industries including retailing and healthcare and industries and implementation issues in business supply chains have been discussed [2], [5], [6], [9]. Numerous studies at MIT's Auto-ID Center have demonstrated RFID applications that result in substantial gains in efficiency and effectiveness of logistics processes, but the studies have also identified situations where the technology needs to advance to provide greater benefits.

Physical Markup Language (PML) has been designed for describing the product information referred to by an RFID tag [1]. The tag data contain simply a code, and these data are translated by an object naming server (ONS) to product information, which is described using PML. Chalasani and Sounderpandian categorize different types of transactions that may arise in a retail store from RFID tag readings [3]. Mandviwalla and Asif

Manuscript received July 26, 2006; revised November 25, 2006 and June 6, 2007. The work of R. Boppana was supported in part by the U.S. National Science Foundation under Grants EIA-0117255 and CRI-0551501. Paper no. TII-06-07-0072.R2.

S. Chalasani is with the School of Business and Technology, University of Wisconsin-Parkside, Kenosha, WI 53141 USA (e-mail: chalasan@uwp.edu).

R. V. Boppana is with the Department of Computer Science, University of Texas, San Antonio, TX 78249 USA (e-mail: Boppana@cs.utsa.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2007.904147

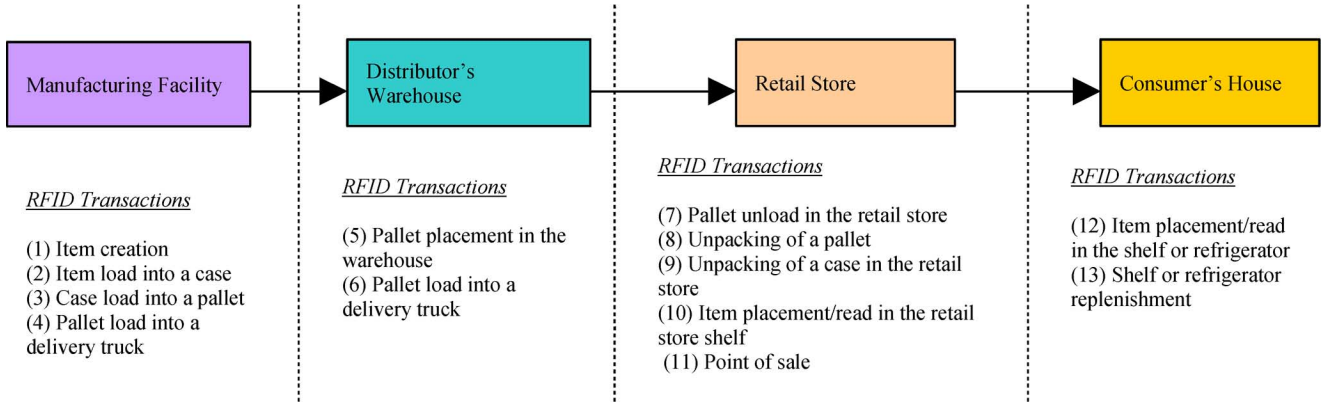

Fig. 1. Transition of an item from the manufacturer to the consumer in the supply chain and the relevant RFID transactions.

present a tutorial on RFID and identify important research problems related to the integration of RFID into the supply chain [6]. Traub envisions an enterprise architecture in which the RFID middleware layer plays a key role. The middleware layer known as application level events (ALE) interface receives tag information from RFID tag readers and forwards this information, after consolidation and pruning, to different applications [9]. Chalasani *et al.* propose building intelligence into the tag readers so that they can achieve automatic identification of misplaced items and automatic generation of shelf replenishment alerts [4]. Ohkubo *et al.* discuss privacy issues related to RFID tags and present an overview of the current solutions on RFID privacy issues [7]. Privacy concerns related to RFID tags have also been discussed in other articles. Garfinkel *et al.* describe the problems related to RFID privacy issues and give an overview of solutions [14]. Juels provides a thorough literature survey of security and privacy issues related to RFID technology [13]. Rieback *et al.* discuss how RFID tags, especially read-write tags, can propagate viruses through enterprise systems [8]. Eckfeldt discusses the benefits and risks of RFID technologies from a consumer perspective [5].

There are a few papers that discuss RFID data models and architectures. Wang and Liu present data models for RFID data and discuss a general rule-based approach to data aggregation/optimization [16]. The issues such as unreliable tag reads and managing high volume of RFID data are discussed by Sarma [17]. Traub *et al.* [18] give an architectural framework for RFID data including the specification of interfaces for the end user's components, but they do not discuss the system architecture, which is the focus of this paper. A detailed comparison of our results with those in the literature is presented in Section IX.

III. RFID Transactions

For the purposes of this paper, we assume the supply chain is comprised of the manufacturer, distributor, retailer, and the consumer. In this paper, we only assume passive tags [2], the most commonly used tags for retail items. However, we do assume that the tags are read-write tags. As an item with an RFID tag moves from one location to another location in the supply chain, it may be read at several different locations in the supply chain. We define an *RFID transaction* to be an event that corresponds to the reading of an RFID tag by an RFID reader. Each RFID transaction generates data including the RFID tag (EPC), the reader id, and other relevant information.

The transition of an item with an RFID tag from the manufacturer to the consumer is depicted in Fig. 1. We assume that the RFID tags are applied at the item, case, and pallet level. As an item is manufactured, an RFID tag is placed on the item, which generates the item creation RFID transaction at the manufacturing facility. Placing an item into a case, placing the case into a pallet, as well as loading a pallet into a delivery truck generate different RFID transactions at the manufacturing facility. At the distributor's warehouse, placing the pallet into a warehouse shelf and loading the pallet onto a delivery truck (to be delivered to the retail store) generate RFID transactions. In a retail store, events such as shelf replenishment, movement of an item from one shelf to another, and sale of an item generate RFID transactions. At the consumer's home, a futuristic model suggests that the consumer's refrigerator will be equipped with an RFID tag reader; this results in RFID transactions being generated when an item is placed in the refrigerator and when an item is taken out of the refrigerator, with these events possibly triggering a refrigerator replenishment RFID transaction.

Processing of the activities generated by the RFID tags in the supply chain information systems is shown as a series of steps in Fig. 2. After the reader reads a tag, the tag together with the ID of the RFID reader are sent to the host computer system (Step 1 of Fig. 2). The host computer system then requests an object naming server (ONS server) to translate the tag (Step 2). The ONS server is very similar to the domain name server (DNS server) in the Internet that translates web URLs into IP addresses. The ONS server accepts a tag as the input and finds out the manufacturer's information and the product information for that tag. This information is transmitted back to the host computer system (Step 3). Based on the information obtained in steps 1 through 3, the host computer system decides the type of transaction that needs to be performed (Step 4). Some RFID transactions may generate alerts for the staff at the manufacturing facility, retail location, and the distributor's warehouse (Step 5). Alerts are required, for example, when the last item of a particular type leaves the shelf and the shelf needs to be replenished. The central server decides whether the inventory has fallen below the reorder point, in which case, it may reorder the item using integrated ERP systems between the manufacturer and the retailer. For each transaction, the transaction details are

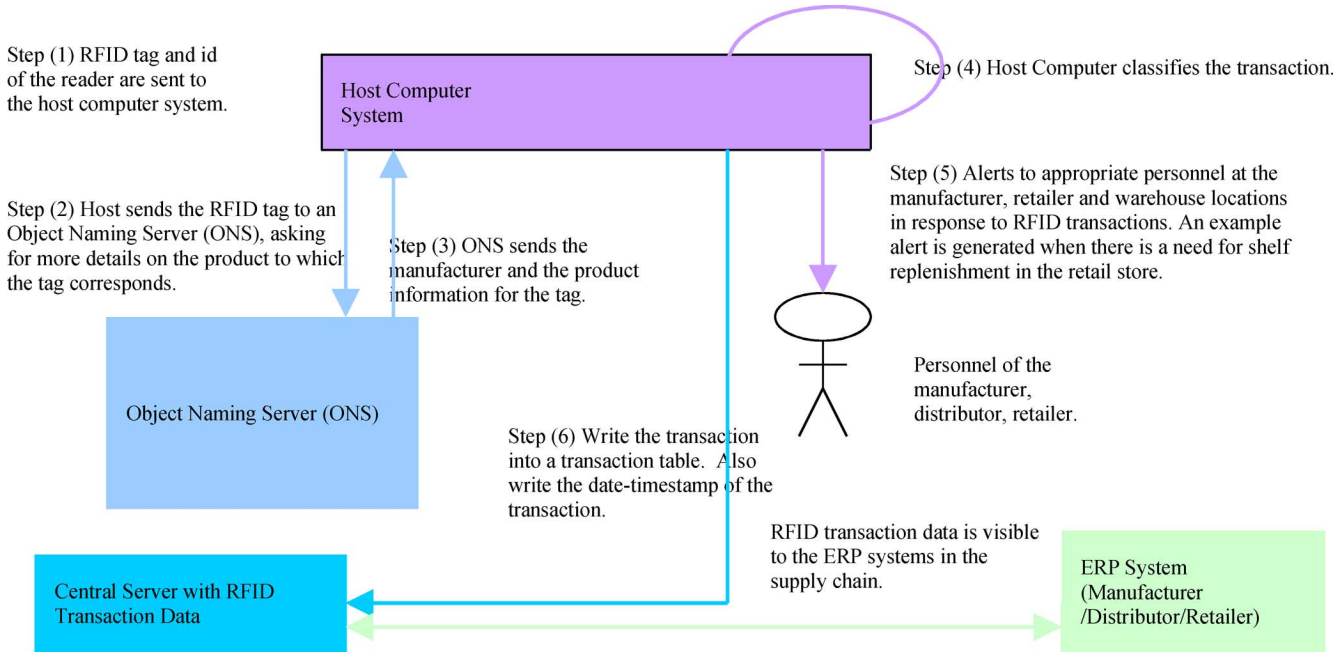

Fig. 2. Processing of RFID tag-reads by the host computer system in the supply chain.

written into a transaction database (Step 6). It is possible to update the ERP system and the Enterprise Operational Data Store (EODS) with RFID transactional data on a regular basis.

IV. Data Needed to Represent RFID Transactions

To arrive at an enterprise data model for RFID transactions, we need to consider what should be stored in response to each RFID tag read. For each transaction, date-time-stamp at which the transaction (event) takes place and the id of the reader that read the tag (Reader_Id) are stored automatically. Hence, these data items are not explicitly indicated in the following discussion. For the item-creation transaction, the item's RFID tag is stored. Each item is created under a unique batch number. These batch numbers can be programmed into the host computer system that processes these transactions.

For the item-load-into-a-case transaction, the item's RFID tag (Product_EPC) and the case's RFID tag (Case_EPC) are stored. Similarly, for the case-load-into-a-pallet transaction, the case's RFID tag (Case_EPC) as well as the pallet's RFID tag (Pallet_EPC) are stored in the database. For the pallet-load-into-a-truck transaction, in addition to the pallet's RFID tag, the distribution channel id needs to be captured. The distribution channel id is associated with the delivery truck into which the pallet is loaded. For the purpose of this paper, we assume that the RFID tag reader's id (Reader_Id) provides the unique distribution id for each truck. We assume that the tag readers are fixed for each truck and the IP address of the tag reader (Reader_Id) serves as the distribution channel id.

Loading items into cases and cases into pallets is typically accomplished in the assembly line as products (items) are manufactured [4]. There is only one case in the assembly line into which a given item can be placed and only one pallet into which any given case is placed. By the position of the tag readers in the assembly line and by processing only one case (or one pallet) at any given time, it can be ensured that the association between item tags and case tags (and between case tags and pallet tags) is error-free.

For the pallet-placement-in-the-distributors-warehouse transaction, Pallet_EPC is captured. It is impractical to have just one RFID tag reader for the entire warehouse, given the large geographical area of a warehouse. We assume that each location (such as an aisle) is uniquely monitored by an RFID tag reader.[1] Thus, the Reader_Id of the RFID tag reader uniquely provides the location where the pallet is placed.

At the retail store, unloading of a pallet and removing cases from pallets generate several RFID transactions [3]. For the pallet-unload transaction, Pallet_EPC is captured. For unpacking-a-pallet transaction, the pallet's RFID tag (Pallet_EPC) and the RFID tags of the cases (Case_EPC) in the pallet need to be stored. For the unpacking-a-case RFID transaction, case's RFID tag (Case_EPC) and the EPCs of items (Product_EPC) in each case need to be stored. In the retail store, item placement into the retail store shelf and item movement from one shelf to another generates transactions that require Product_EPC to be stored.

Point of sale (POS) transaction requires the following data items to be stored into the database: RFID tag of the item (Product_EPC) and POS transaction id. Here the POS transaction id is the id that uniquely identifies the sales transaction in the retailer's ERP system. ERP systems such as SAP readily provide this information [6]. For discount retailer stores such

[1]In some instances, however, this assumption is not valid. In a retail store, an item may be read by tag readers in adjacent isles in addition to the tag reader designated for its aisle. A resolution mechanism should be used to address this problem. For example, tag readers can keep track of signal strength, and in the event that an item is read by multiple readers, the reader with the strongest average signal will be the designated reader; or the central computer with item inventory database could designate the official tag reader. Another approach, applicable when multiple units of the same kind are stocked, is one in which the tag reader that scans most of the items of the same kind becomes the designated reader for those items. Therefore, we assume that each item can be kept track of by a unique tag reader.

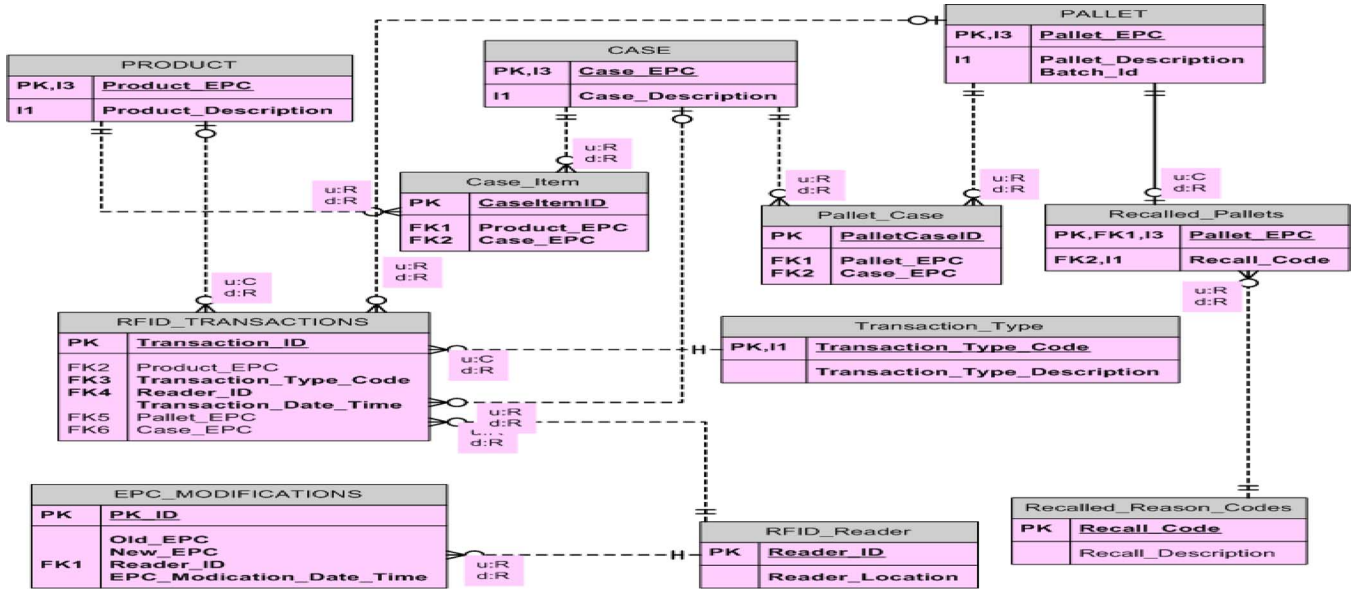

Fig. 3. Data model to hold RFID transactional data in manufacturer's enterprise operational data store (EODS).

as SAM's Club, customer id is available as the member id. However, for stores such as Wal-Mart and Target, customer id is available only through the POS transaction record based on the credit card purchases. For the item-placement-in-the-refrigerator transaction in the consumer home, Product_EPC needs to be captured. We assume that the Consumer_Id may be uniquely identified by the Reader_Id. Capturing this may raise privacy concerns. Such privacy issues may be addressed by scrambling customer information and RFID tag values, which can be unscrambled only with customer's authorization [13], [14]. We discuss how the privacy issues fit into the proposed data models in Section V.

V. Data Models for RFID Transactional Data

The transactions described above are handled by several tables in the enterprise operational data store (EODS). These tables are depicted in Fig. 3 for the manufacturer. The Reader table contains the Reader_ID for each RFID reader. This reader ID is the primary key in this table; the reader ID can be the IP address or a similar number assigned to each reader. In addition, it contains the location of the reader. Reader_Location often is a composite attribute containing the aisle and shelf and other data that precisely identifies the location of the reader.

The product table has several attributes pertaining to the product, such as the product description. Product_EPC is the electronic product code (EPC) that uniquely identifies each product and is embedded in the RFID tag. The pallet table contains information on pallets including the EPC, description of the pallet, and the unique batch number (Batch_Id) under which the pallet was created. Similarly, the Case table contains information on cases. A separate table maintains the relationship between the pallet and the cases contained in that pallet. This table, referred to as the Pallet_Case table, simply contains Pallet_EPC and Case_EPC. Case_Item table maintains the data on the items contained in each case. Recalled Pallets table stores information on recalled pallets along with the reasons for the recall.

Transaction type table is a lookup table that assigns transaction codes to each type of transaction (such as Point of Sale or Shelf Replenishment or Item Placement). Each of the tables—Reader, Product, Case, Pallet, Transaction Type—have a one-to-many relationship with the Transactions table, with the many side of the relationship ending on the Transactions table. The transactions table holds each RFID transaction at the enterprise level by recording the applicable data such as product EPC, case EPC, pallet EPC, the reader ID, and the transaction type.

Fig. 4 shows a similar data model for the retail store. The retailer's model has other attributes such as POS_Transaction_ID, which is the ID that corresponds to the point-of-sale data record for this item in the enterprise database. Maintaining POS_Transaction_ID in the transactions table is essential to identify the customer to which the product with a specific RFID tag was sold. Similar data models can be designed for the distributor. The retailer and the distributor do not have information such as Batch_Id.

The above data models allow tag modifications. For example, an RFID tag can be rewritten by a reader in a process where the old tag is replaced with a new tag. Rewriting an RFID tag allows a tag to be reused. The table EPC_MODIFICATIONS contains the old tag value and the new tag value. It also holds information on the reader that modified the tag and date-time-stamp of tag modification. This information is very useful, for example, in validating the authenticity of a tag.

The data models discussed in this section, especially the retailer's EODS model, may include information on the customer's purchases by including the POS_Transaction_Id as part of the RFID transaction table. This allows integration of the RFID data with the ERP applications such as customer sales. This raises consumer concerns on privacy of data related to consumer purchases. Several technical solutions have been proposed in the literature to mitigate such concerns [7], [13], [14]. Some of the solutions include the following: 1) using the EPC kill command, which kills the tag at the point of sale

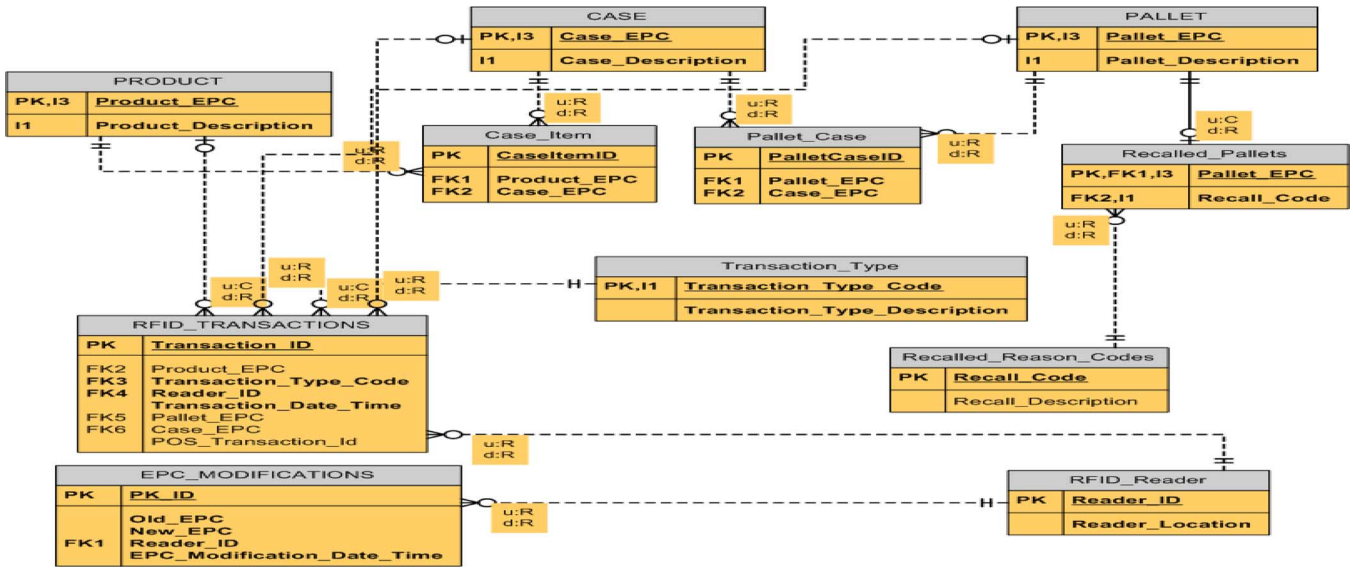

Fig. 4. Data model to hold RFID transactions in retailer's Enterprise Operational Data Store (EODS).

so that the item in customer's possession cannot be tracked further; 2) encrypting tag information using cryptographic keys and managing the keys needed for encryption and decryption; 3) giving each tag a set of pseudonyms and letting the tag cycle through these pseudonyms each time the tag is read; and 4) employing blocker tags that "spam" unauthorized readers so that they receive false information on the tag's identity. These solutions can be incorporated into the proposed data models by adding new tables and/or attributes to the data model as discussed below.

- EPC kill command solution: Add attributes to the RFID Transactions table to capture the following data: EPC_Killed, which is a Boolean variable that indicates whether the EPC tag has been killed or not. In this solution, EPC_Killed attribute will be set to true when there is a non-null value for the POS_Transaction_Id.
- Encryption solution: Add a new table that maintains the encryption and decryption keys for different types of tags. This table may be designed so that different product types have different keys.
- Tag Pseudonym solution: Add a new table EPC_Pseudonym that has at least the following attributes: EPC (this can be a pallet, case, or item EPC) and the EPC_Pseudonym (which identifies a single pseudonym for the corresponding EPC). The EPC and the EPC_Pseudonym attributes together form the primary key of the table; multiple pseudonyms can be added for a single EPC by adding multiple rows to this table.
- Blocker tags solution: This solution is not based on keeping additional information on the tag and hence does not necessitate changes to the data models.

VI. RFID Transactional Data Volume and Optimization

Let n be the total number of items, c the total number of cases, and p the total number of pallets. Also, let f_n, f_c, and f_p denote the reading frequency of items, cases, and pallets. If B denotes the number of bytes needed to store an RFID transaction and T the period during which tags are read, then the total storage needed, S, is estimated using

$$S = (n * f_n + c * f_c + p * f_p) * T * B. \quad (1)$$

The storage needed for RFID transactions can be quite large, even if we only consider the transactions in a retail store. For example, if there are 100 000 items on the shelves at a retail store location, and the items are read every 15 min, there are 400 000 transactions every hour. In addition, if each transaction requires 256 bytes of storage, the total storage requirement is 100 MB. If the store operates on average for 15 h a day and the retailer has 1000 such stores, the total storage required in the retailer's EODS is 1.5 terabytes a day, not considering the transactions at the manufacturing facility and the distributor's warehouse and the transactions involving cases and pallets.

A. Graph Model for RFID Transactions

To understand the essential transactions that need to be kept in the database, we propose a graph model to represent RFID transactions of items. In this model, each node corresponds to an RFID reader. There is a directed arc between node X (corresponding to reader X) and node Y (reader Y) with label $[\mathrm{t}, \mathrm{t}+\mathrm{T}]$ if the tag is read by reader X at time t and the same tag is read by reader Y at time $\mathrm{t} + \mathrm{T}$, where T is the duration between scans. If X and Y are different, then this activity represents the movement of an item from one shelf (monitored by reader X) to another shelf (monitored by reader Y) in a retail store. Otherwise, it represents consecutive readings by the same tag reader, which can be depicted by a self-arc in this graph model. Both possibilities are indicated in Fig. 5.

In terms of storage requirements, each arc in this graph model corresponds to two rows (entries) in the RFID_Transactions table for the same tag. One row corresponds to $\mathrm{Reader\_ID} = \mathrm{X}$ and $\mathrm{Date\_Time\_Stamp} = t$, while the other row corresponds to $\mathrm{Reader\_ID} = \mathrm{Y}$ and $\mathrm{Date\_Time\_Stamp} = t + T$.

A simple approach to reduce the amount of data storage needed is to keep track of the first and last timestamps of

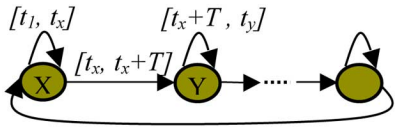

Fig. 5. Graphical representation of reading an item by different tag readers and by the same tag reader.

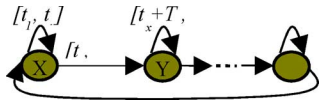

Fig. 6. Graphical representation of reading a misplaced item by different tag readers at different times.

continuous readings of an item by a tag reader. In terms of our graph notation, this can be restated as follows.

If there are multiple self-arcs on node X (corresponding to reader X) with the following labels

$$[t, t+T], [t+T, t+2T], , [t+(n-1)*T, t+n*T]$$

$(n \geq 2)$, replace these self-arcs with one self-arc labeled $[t, t+n*T]$.

For easier reference, we denote this as *Strategy 0*. The total storage requirement is given by

$$S_0 = 2*(n*g_n + c*g_c + p*g_p)*B \qquad (2)$$

where g_n, g_c, g_p are the number of tag readers through which an item, case, or pallet passes through, n, c, and p are the number of items, cases, and pallets, B is the number of bytes needed for each transaction, S_0 is expected to be significantly smaller than S. As an example, if an item on the average is read by a sequence of 200 tag readers between the time it is manufactured and sold, the total amount of storage for 100 000 items equals $2*100\,000*200*256$ bytes, which is approximately 10 GB, for the complete life-cycle of the items in the supply chain. This is in contrast to the requirement of 1.5 terabytes *a day* if no optimization technique is applied to RFID transactions.

However, when an item is moved from one location to another—for example, when the item is misplaced by a customer in a retail store—it is not obvious how multiple tag reader scans of the item over a period of time can be represented efficiently. To present storage reduction techniques for this scenario, we define a home location for each item, case, and pallet. For any given RFID tag (corresponding to an item, case, or pallet), its home location is defined as the location(s) where the item is expected to reside in before it leaves the premises. For example, in a retail store, items are placed in specific shelves and are expected to stay in those shelves until they are picked up by customers and eventually underwent point-of-sale transactions. For the manufacturing facility and the distributor's warehouse, typically there may be a specific home location for each type of pallets. A home location thus can be defined as the set of RFID tag readers corresponding to the shelves in which the item may reside before it leaves the premises (manufacturing facility, retail store, or distributor's warehouse).

To discuss the optimization technique related to transactions for misplaced items, we use a retail store scenario in the following discussion. We assume that there is a unique home RFID tag reader for each item. (If multiple tag readers are used to cover the home location of an item, then this requirement is satisfied by assigning a common prefix value in their IDs.) A misplaced item leaves its home location, perhaps by customers' actions, and is brought back to its home location by retail store personnel. While the item is away from its home location, it is read by other tag readers causing RFID transactions for the item. When the item is brought back to its home location, it corresponds to a multinode cycle in our graph model as illustrated in Fig. 6. Node X denotes the tag reader at the item's home location, and the other nodes denote the other tag readers that tracked the item during its traversal away from the home location.

An item may be misplaced more than once. The corresponding graph model for this item has more than one multinode cycle. In that case, we believe it is of interest to only keep the transactions related to the most recent traversal. Transactions related to completed cycles may be removed unless sabotage is suspected. In that case, these transactions should be moved to another database suitable for security analysis. Otherwise, the transactions of the completed cycles are not of interest.

We consider two storage strategies to keep track of an item's traversal—misplaced and has not returned to its home location.

1) Strategy 1: Store the first and last time stamps of continuous scans of the item by its home location and the current location.

This strategy is useful for low-cost items, for which it is only necessary to know their current locations so that store personnel can put them back at their respective home locations. The reason for keeping track of the last time stamp at home location is to determine the duration for which the item is misplaced and using it in analyzing the sales potential of the item. This strategy can be implemented using one table of storage by maintaining the storage records for only the home location reader and the current location reader and deleting the records for any other location in the table. The record for the home location is always maintained in this table, while the records for other locations are removed, except for the record corresponding to the most recent read.

2) Strategy 2: Store the first and last time stamps of continuous scans of the item by its home location and every location it visited.

This strategy is useful for high-cost items, for which it may be necessary to conduct security analysis to see if any foul play is involved. It is also useful in manufacturing and distribution processes, since analysis of pallet movements can facilitate evaluation and improvement of operational efficiencies.

To understand the storage implications of the two strategies, consider a retail store scenario where items reads are of primary interest. Let m be the probability that an item is misplaced and $k \geq 1$ be the average number of locations visited by a misplaced item before it is restored to its correct location. With *Strategy 1*, the transaction storage given in (2) is increased by a factor of $(1+m)$, since at any given time, only the misplaced items require additional storage. With *Strategy 2*, the storage requirements are increased by a factor of $(1+km)$, if only the latest traversal of a misplaced item is to be kept. If it is desired that all data on misplacements be preserved for post-incident

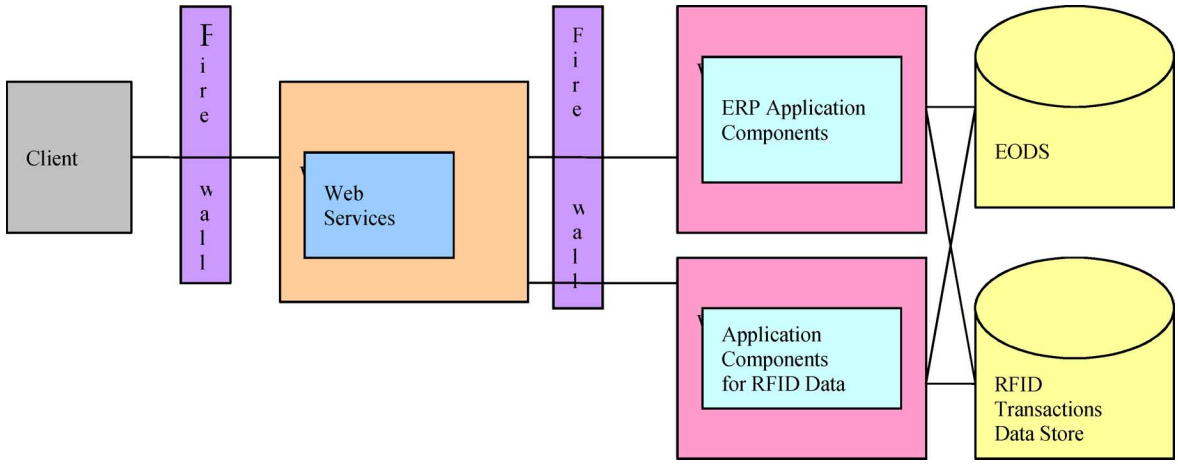

Fig. 7. Enterprise architecture to store and process RFID-related transactions.

security/efficiency analysis, then the storage requirement is increased by the number of times the item is misplaced.

To reduce the storage needed for Strategy 2, we propose a storage optimization based on our graph model. We can represent the duration of an item's stay at each location by a 3-tuple: $\langle tag\ reader\ id, t_f, t_l \rangle$, where t_f and t_l denote the first and last time stamps of continuous scans of the item by the tag reader. We will use an extended record to store the 3-tuples corresponding to each location visited by the item during its misplacement. This extended record is kept in a separate table, and its unique id is stored in the main record for the item. This reduces the number of transaction records kept in the main table and the number of times a table needs to be accessed to analyze the traversal of an item.

VII. System Architectures for RFID Transactions

Fig. 7 shows an information system architecture that incorporates processing modules for RFID transactions. This architecture model suggests an enterprise-level architecture that integrates the ERP modules and the RFID transaction processing modules. By this enterprise-level architecture, we do not mean a geographically centralized architecture. Different components of this architecture shown in Fig. 7 may reside at different locations: for example, the ERP processing module may be located in New York, while the RFID processing module may be located in Chicago. However, the individual pieces of data required for processing are not decentralized in this architecture. That means the complete data belonging to the RFID transactions table are available in one location for each player in the supply chain. That is, the RFID transactional data for the retailer are stored in the EODS of the retailer, while RFID data of the manufacturer are stored in the EODS of the manufacturer. Our proposal for an enterprise-level architecture has the following advantages.

1) Consistent, Integrated View of Data: At any given time, all the RFID data are available to any application at the enterprise level.

2) Fast Access to Data: When the data are distributed throughout the supply chain, to obtain relevant pieces of information for applications such as product recall requires a significant amount of time. On the other hand, an enterprise-level data architecture provides information on items, warehouses, and the customers who purchased a recalled item very quickly.

3) Ease of Extension From the Existing Architectures: Already companies have spent billions of dollars in implementing ERP architectures. To implement decentralized middleware devices that hold RFID and related data, the cost of the infrastructure may increase substantially. On the other hand, the architecture shown in Fig. 7 works with and extends the existing ERP architectures with nominal increases in cost.

4) Fast Updates of the EODS With RFID Data: Using the data optimization strategies discussed in Section VI, the storage requirements for RFID transactions can be significantly reduced to just a few gigabytes for each tag. The EODS can be easily updated with RFID data on a regular basis. The advantage of making the RFID data available in the EODS is that the most recent data are readily available for applications such as product recall and retail store shelf replenishment.

The architecture model shown in Fig. 7 is applicable to any entity in the supply chain. For example, the RFID data store indicated in Fig. 7 implements the data model in Fig. 3 if this architecture is used for the manufacturer.

Fig. 8 shows how these architectures can be integrated between the manufacturer, retailer, and distributor. In the model indicated in Fig. 8, each entity in the supply chain holds its own data on RFID transactions. FW refers to Firewall, WS to Web Server, and WAS to Web Application Server. For simplicity, the ERP and RFID application components are combined together and hosted under one Web Application Server. The manufacturer, distributor, and retailer allow each other access by providing secure login with a password for each partner. Fig. 8 depicts how manufacturer, distributor, and retailer can cooperate together and exchange data among themselves to accomplish a given task. For example, if a product produced by the manufacturer needs to be recalled, the enterprise systems at different locations interact together to accomplish this task.

The product recall application is discussed in more detail in Section VII-A. The nodes labeled with 2, 3, 8, and 9 correspond to the steps of the product recall application and are explained in more detail in Section VII-A.

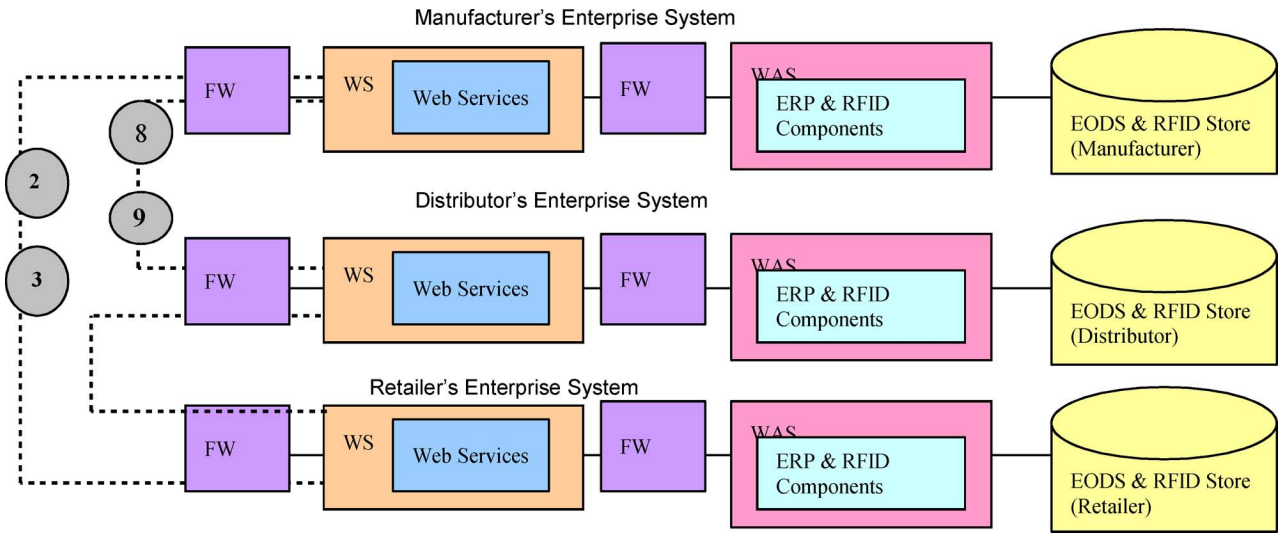

Fig. 8. Interrelation and access between the enterprise systems of the manufacturer, the retailer, and the distributor.

A. *Prototype Implementation*

We implemented a prototype based on the data models and the data architectures presented in this paper. We worked with Teska and Associates, a consulting company based in Racine, WI to implement the prototype [15]. In this prototype, we have used fictitious data so that no single company's data are revealed. Also the security layer is disabled so that the readers can experiment with the prototype. The prototype has the following four major functions.

- Show Product, Case, Pallet Details by EPC: The user can type an EPC and find the details on the product (or case or pallet) for that EPC. The detailed report for that EPC includes all the RFID transactions for the EPC including the date-time stamp and the corresponding reader information.
- Search for Products By Product Name: The user can select a product name and find all product EPCs that correspond to that product. The user can then obtain a drill-down report on a single EPC.
- Search for Transactions By Reader: The user can select a reader by its location and find out all RFID transactions generated by that reader.
- Search for Transactions by Transaction Type: The user can select a transaction type (such as item creation, pallet shipped) and find out all RFID transactions that correspond to that transaction type.

This prototype was implemented using Java Server Pages and the enterprise Java technology for the programming and presentation logic. MySQL database was used to implement the data models indicated in Figs. 3 and 4.

For the prototype implementation, we labeled the data needed by the application based on how frequently the data changes. The labels of static data and dynamic data are given according to the following principles.

- Label the data that change frequently as "dynamic" data. Examples of such data include RFID transactional data since new transactions are generated on a continuous basis.
- Label the data that change infrequently as "static" data. Examples of these data include the readers in a retail store and transaction types.

We designed the prototype such that the static data are read and stored by the application in memory cache at the beginning of the application. The dynamic data in the prototype are read by the application on a demand-basis as and when a customer needs it. Since dynamic data are related to RFID transactions, we cannot pre-cache dynamic data. To speed up the process of reading dynamic data from the data stores, our prototype implements effective strategies for database connection pooling. The following are a couple of lessons that were learned from the construction of the prototype and its marketing to regional companies.

1) Lesson 1: While the generic data models described in the previous sections are applicable to most corporations, they need some changes for implementation at any given corporation. This is because different corporations employ different software technologies and ERP technologies and integration of the RFID transactional data stores with the ERP systems differs from one corporation to another corporation. For example, many organizations have implemented the SAP software as their ERP software, while some other organizations use systems such as Baan, PeopleSoft, and other Oracle products. The data models and the system architecture models we propose can be applied to any ERP implementation, in general, with some customization as needed. The communication between the RFID data stores and the current ERP implementation can be easily facilitated using a clearly defined eXtensible Markup Language (XML) interface. XML documents based on the RFID data models discussed in this paper can be easily implemented at the database architecture level by the commercially available relational database products such as Oracle or IBM DB2.

2) Lesson 2: There is no uniform method that corporations currently use to gather and store RFID data. For example, some companies have even used spreadsheets (based on software such as Microsoft Excel) to maintain RFID tag data. No comprehensive data models are being used by supplier companies to store RFID transactional data. The data models we presented in this paper are a big enhancement to the ad-hoc methods that different corporations are using to gather and store RFID transactional data. Implementation of the RFID data models discussed in this

Process Steps for Accomplishing Product Recall by the Manufacturer
// Input to this process is the batch number(s) corresponding to the pallets that need be recalled.

Step 1: Manufacturer's PRODUCT RECALL process identifies the Pallet_EPCs of the products to be recalled. This identification takes place based on the batch numbers (see Figure 3).

Step 2: Manufacturer's PRODUCT RECALL process logs into the retailer's PRODUCT RECALL process.

Step 3: Manufacturer's PRODUCT RECALL process sends the Pallet_EPCs of the recalled pallets that were distributed to the retailer. After successfully sending this information, Manufacturer's PRODUCT RECALL process logs out.

Step 4: Retailer's PRODUCT RECALL process identifies the Cases and the Items that need to be recalled, based on the Pallet_EPCs it received from the manufacturer. Pallet_EPCs of the recalled pallets are written to the Recalled_Pallets table.

Step 5: Retailer's PRODUCT RECALL process identifies whether any of the recalled items, cases and pallets are still in the retail store. If so, a report is generated identifying the items, cases and pallets with their locations and an alert (via pager/PDA) is generated for the personnel responsible.

Step 6: Retailer's PRODUCT RECALL process identifies whether any of the recalled items have been sold to the customers. It identifies the POS_Transaction_Id of such purchases and sends a request to the ERP system to obtain information of the customers who purchased the items.

Step 7: Retailer's PRODUCT RECALL process prepares a report of the customers who purchased recalled items and their contact information. An alert (via pager/PDA) is generated for the personnel responsible.

Step 8: Manufacturer's PRODUCT RECALL process logs into the distributor's PRODUCT RECALL process.

Step 9: Manufacturer's PRODUCT RECALL process sends the Pallet_EPCs of the recalled pallets to the distributor's PRODUCT RECALL process. After successfully sending this information, Manufacturer's PRODUCT RECALL process logs out.

Step 10: Distributor's PRODUCT RECALL process writes the Pallet_EPCs of the recalled pallets into the Recalled_Pallets table.

Step 11: Distributor's PRODUCT RECALL process identifies whether any of the recalled pallets are still in the distributor's warehouse. If so, a report is generated identifying the pallets with their locations and an alert (via pager/PDA) is generated for the personnel responsible.

Fig. 9. Process steps for accomplishing product recall by the manufacturer.

paper is relatively easy, and it can be accomplished with minimal changes to the organization's ERP implementations. Most organizations, even when they completely endorse the use of RFID at pallet, case, or item level, are unsure of future applications—beyond the current tracking applications—of the RFID data they collect. In such cases, we suggest implementing the data architectures developed in this paper and gathering the RFID transactional data into an enterprise-level data warehouse for possible future data mining and business intelligence applications.

VIII. Industrial Applications of RFID Data

The RFID data architectures presented in this paper enable businesses to have a more complete view of the supply chain at any given time. Questions such as

- Where exactly is a given pallet, case, or item located?
- How fast are items being sold at retail stores?
- How long are pallets, cases, or items held at a distribution site or a retail store?

can be answered within seconds using the RFID data architectures described in this paper. Obtaining such information without using the data models and architectures presented in this paper is very cumbersome, if not impossible. We next present two applications of the data architectures presented in this paper.

A. Application: Product Recall Initiated by Manufacturer

Product recall may be generated by different events: malfunction of an assembly line, bad raw materials, and improper storage of pallets of items. Malfunction of an assembly line requires product recall of all items manufactured by the assembly line in a given time period. Improper storage of items requires recall of items in specific pallets, while the use of improper raw materials requires recall of all items that used those raw materials. Product recall may be triggered by the manufacturing facility if, for example, it is discovered that the assembly line did not function according to specification for a given time period. Governmental bodies such as the United States Food and Drug Administration (FDA) may also force a product recall.

We discuss the processes involved when the product recall is initiated by the manufacturer. For the purposes of this discussion, we assume that there is a PRODUCT RECALL application module on the manufacturer, distributor, and the retailer information systems. This PRODUCT RECALL application module is one of the components indicated in the "Application Components for RFID Data" in Fig. 7. We also assume that the manufacturer, distributor and retailer PRODUCT RECALL modules have access rights to each other. Fig. 9 shows a step-by-step procedure for accomplishing product recall initiated by the manufacturer. Manufacturer's process simply logs into the retailer's and distributor's PRODUCT RECALL processes and sends them the Pallet_EPCs of the pallets to be recalled (Steps 2, 3, 8, 9). The communication between the manufacturer's enterprise system and the retailer's enterprise system takes place in Steps 2 and 3, and this is indicated in Fig. 8 by dashed lines. Similarly, the communication between the manufacturer's enterprise system and the distributor's enterprise system takes place in Steps 8 and 9. The retailer's process generates reports of the

items to be recalled and whether any such items are sold to customers using a sequence of steps (Steps 4 through 7). Similar processing takes place in the distributor's PRODUCT RECALL application in Steps 10 and 11. Steps 6 and 7 use customer information needed for notifying customers who purchased the recalled products. Such identification may invariably lead to privacy concerns. To alleviate such privacy concerns, the retailer should consider a policy of maintaining this information only for those customers who agree to such tracking.

B. Application: Shelf Replenishment

Frequently, a retailer may have several locations in a city, but one of the locations may run out of an item that a customer needs. In such circumstances, the customer may simply choose to go to a competing retailer to obtain the same. A few other times, the customer may ask the retail store clerk to check other stores for the item. The store clerk checks the inventory and informs the customer which location has that item. However, the clerk typically does not guarantee that the item is not misplaced and that it is actually available at the designated place in that store without speaking to the personnel at that other store. Therefore, upon request, the clerk calls that location and speaks to someone at that location to hold the item for the customer. The customer then travels to that retail location and obtains the item. In many cases, the customer may prefer to have the item transferred to the store he/she is shopping at and pick it up the next day. This process requires manual intervention and is error-prone. With RFID implementations, this can be automated and the items may be transferred from location to another to improve the speed and accuracy with which the customer's request is satisfied.

We first analyze the time it takes to replenish an item, when the retail location runs out of that item. We also look at the cost of replenishment. For each case, we present a probabilistic model to estimate the benefits of RFID implementation; such an analysis can be used by both the vendors of RFID products and their potential customers.

1) Analytical Model to Estimate Shelf Replenishment Time: Let l denote the number of store locations among which items may be transferred. We denote the cost of transferring an item from one retail store location to another by c. The stores are denoted as $S_0, \ldots, S_{l-1}$. Let the number of units of a particular item at store S_i be n_i at some particular time (for example, at the beginning of the week after weekly deliveries from the warehouse/distributor). The number of customers purchasing or attempting to purchase the item may be modeled using the Poisson process [10]. (The Poisson process is known to accurately model several natural phenomena such as the number of incoming phone calls to a trunk switch and the number of job arrivals to a file server.) Without loss of generality, let us assume that store S_0 is sold-out of the item in question at time T. Then the rate at which the item is purchased by customers is $\lambda = n_0/T$.

Using this information, the number of units sold at other locations may be modeled as follows. The probability that some $k < n_i$ units sold at store S_i is the same as the probability that there are k number of customers purchasing the item. The probability that the last unit of the item is sold, however, includes the probability that n_i or more customers purchased or wanted to purchase the item. Therefore, the number of units of the same item sold at another store, say, S_i, in time interval $(0, \mathrm{T}]$ is a modified Poisson distribution with the following the probability mass function:

$$\mathrm{P}[\text{Store } S_i \text{ sells } k \text{ items}] = \begin{cases} \frac{(\lambda T)^k e^{-\lambda T}}{k!}, & 0 \le k < n_i \\ \sum_{k=n_i}^{\infty} \frac{(\lambda T)^k e^{-\lambda T}}{k!}, & k = n_i \\ 0, & k > n_i. \end{cases}$$

The first equation gives the Poisson probability of selling $k < n_i$ units, which is the same as the probability that k customers purchased the item. The second equation is the probability that there are n_i or more customers who attempted to purchase the item (but only the first n_i are successful). The third equation indicates that the probability of selling more than n_i items at the retail store location is zero.

Let p_i be the probability that a certain item is not sold-out at location S_i. We can estimate p_i as

$$p_i = 1 - \mathrm{P}[\text{Store } S_i \text{ sells all } n_i \text{ items}].$$

For simplicity, let us assume $p = p_1 = p_2 = \cdots = p_{l-1}$. Therefore, $(1-p)$ is the probability that this item is also sold-out at any other chosen retail store. Let t_r be the time to transfer an item from one location to another. Let t_d be the time required to move an item from a distributor/warehouse to a retail location.

Probability that the item is sold-out at each of the other locations is $(1-p)^{l-1}$. Hence, the probability that the item is found at one or more of the other locations is $1 - (1-p)^{l-1}$

$$\begin{aligned} T_{fill} &= \text{Average time to replenish the sold-out item} \\ &= [1 - (1-p)^{l-1}]t_r + (1-p)^{l-1} t_d. \end{aligned}$$

Fig. 10 presents a graph for T_{fill} for various values of l and t_d with $t_r = 1$ day and $p = 0.1$.

2) Analytical Model to Estimate Shelf Replenishment Cost: To estimate the cost of replenishing a sold-out item by obtaining from another store or from the distributor, we need additional notation. Let c_d be the cost of transferring the sold-out item from

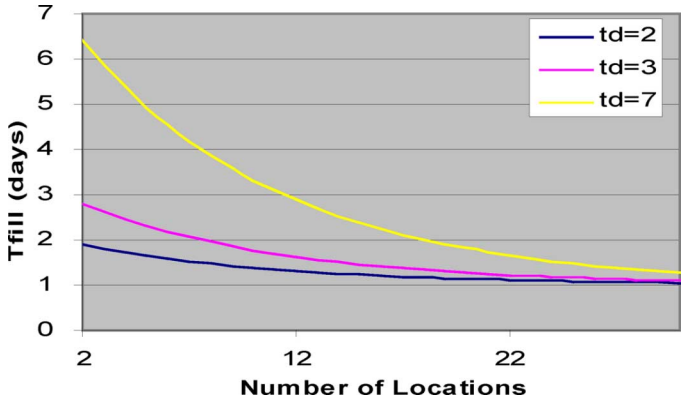

Fig. 10. Average time to replenish a sold-out item from one of the other locations or from the warehouse/distributor. The time to transfer an item from another location is 1 day, and the probability that a product is sold out is 0.1.

the warehouse/distributor to the store, c_r the cost of transferring the same from another location, and c_s the cost of empty shelf-space (in terms of lost sales). If a sold-out item is always replenished by obtaining more units from the warehouse/distributor, then the cost of replenishment is $c_d + c_s t_d$. However, to reduce the cost of an empty shelf, the item may be transferred from another location that has extra units

$$\text{P[Location } S_i, i \neq 0, \text{ has two or more units of the sold-out item]} = 1 - \text{P}[S_i \text{ sold } n_i - 1 \text{ units}] - \text{P}[S_i \text{ sold } n_i \text{ units}].$$

Let this probability be denoted as p_s. Let p_f denote the probability that one or more stores have at least two units of the item sold-out at S_0. It can be calculated as $p_f = [1 - (1 - p_s)^{l-1}]$. If l is large, then p_f is nearly 1

$$\text{The average cost of replenishment} = c_d + (1 - p_f) c_s t_d + p_f \, (c_r + c_s * t_r)$$

where t_r is the time to transfer an item from one location to another, t_d the time required to move an item from a distributor/warehouse to a retail location, and c_d, c_s, and c_r are as defined above.

The first term is the cost of transferring the item from the distributor since a sold-out item needs to be replenished at one store or another. The second term estimates the weighted cost of an empty shelf is if the item has to be sent by the distributor and none of the other stores have spare units of the item; the third term estimates the weighted cost of transferring the item from another location and the cost of an empty shelf in the mean time. It is cheaper to transfer from another location, if $t_d \gg t_r$ and c_s is high.

3) Implementation Using the Proposed Data Model: We now describe how shelf replenishment can be implemented using the data architectures described in this paper. Similar to the product recall application, we assume that there is a SHELF REPLENISHMENT application module on the manufacturer, distributor, and the retailer information systems. This SHELF REPLENISHMENT application module is one of the components indicated in the "Application Components for RFID Data" in Fig. 7. We also assume that the manufacturer, distributor, and retailer SHELF REPLENISHMENT modules have access rights to each other. By searching the Product table and the RFID_Transactions table in Fig. 4, the retailer's SHELF REPLENISHMENT application can identify the retail locations at which an item is available. RFID_Transactions table is crucial for this identification, since it allows the items to be tracked even when they are misplaced. Note that with either storage optimization strategy discussed in Section VI-A, every item's current location is always tracked; this allows the retailer's application to accurately identify the number of items available at any given location. If the item is available at one or more retail locations, the retailer's application can check which of the locations are closer to the location where the item is completely sold out. If the retailer's SHELF REPLENISHMENT application does not find any suitable retail store from which the item can be replenished, it checks with the manufacturer's SHELF REPLENISHMENT application to identify the pallets that contain the products required by the retail store. Manufacturer's SHELF REPLENISHMENT process identifies the pallets for the products by searching/joining the Pallet, Case, Product, Pallet_Case, and the Case_Item tables of the data model in Fig. 4. Retailer's SHELF REPLENISHMENT process then interacts with the distributor's SHELF REPLENISHMENT application module to identify a distributor's warehouse where the pallets indicated by the manufacturer are located. The retailer's SHELF REPLENISHMENT process then requests replenishment from a nearby warehouse.

IX. Related Work

Perhaps the previously published work closest to ours is the paper by Wang and Liu [16]. Their data models are similar to ours. However, our work differs from their paper in several aspects. For example, we address supply chain issues such as product recall and shelf replenishment and show how dynamic modifications to tags (in the case of read-write tags) can be handled. Another difference is Wang and Liu's data optimization techniques are rule-based; rules are placed to filter data based on readings of tags by the readers or to change the ending time-stamp of RFID observations. For example, in Wang and Liu's approach, duplicate readings from multiple readers within a given time period can be eliminated by placing a rule to ignore the reading by one reader. Our optimization techniques focus on applications such as tracking misplaced items. The concept of home location for an item is required for re-shelving misplaced items. Similarly, finding the history of all tag reads is important for a lost item. With a filter that discards all observations within a time period except the observation of one tag reader, as proposed by Wang and Liu, it is impossible to obtain a history of the item's traversal through the retail store. Our graph-based optimization techniques optimize RFID observation data while maintaining a history of the tag reads by multiple successive readers. The data optimization techniques that we present in this paper are based on graph models where the successive tag reads are indicated by the traversal of the graph. We indicated different strategies to optimize data volumes and discussed the differences between these strategies. We also presented models to analyze the benefits of faster product replenishment at retail stores.

The problem of managing RFID data has also been pointed out by Sarma [17]. Sarma's article highlights issues such as unreliable reads and high-volume of RFID data and briefly discusses solutions such as ignoring intermittent appearances of tag reads. In contrast, we address the issue of data volume extensively and present graph models to describe our optimization strategies. Such a theoretical model is helpful in understanding the problem at a more abstract level.

Traub *et al.* present EPCglobal architectural standards for RFID data [18]. As indicated by them, the standard gives the interfaces that the end user's components need to implement but does not define the system architecture (see [18, p. 22]). We presented system architectures in Section VII that enable supply chain applications such as product recall and automatic shelf replenishment. Our work complements the work presented in [18] and assumes that the ERP implementations are already in place for the organizations involved in the supply chain. Since organizations have already collectively spent hundreds of millions of

dollars in ERP implementations in recent years, it is of importance to suggest RFID system architectures that work with existing ERP implementations. Section VII discussed such RFID system architectures and a prototype implementation based on the ideas in this paper.

X. Conclusions

In this paper, we considered a supply chain comprised of the manufacturer, distributor, and the retailer. We first discussed different events that produce RFID transactions in the supply chain. Then we presented data models to store the data generated by these transactions. We indicated how processing of these transactions can be accomplished at the enterprise level. We discussed storage requirements for RFID transactional data, and developed strategies to reduce these storage requirements significantly. We discussed how RFID data models presented in this paper can be useful for applications such as product recall by the manufacturer and for faster shelf replenishment. Analytical models for costs and benefits corroborated by results from practical implementations will help convince organizations to adopt RFID-based technologies faster. Also business cases that discuss the return on investment (ROI) with RFID technologies may shed more light on the benefit of RFID in the supply chain. In academia, several labs are conducting experiments with RFID tags and data generated by those tags. Some of these studies include the performance of the RFID tag reader networks and the reliability of the overall RFID data architectures in the supply chain. Best practices in designing RFID tag reader networks and integrating the RFID tag readers with the enterprise architectures is another strong area for future research. Coping with faults in the RFID networks in the supply chain is yet another area for future research.

Acknowledgment

The authors would like to thank the anonymous reviewers whose suggestions helped improve the quality of this paper.

References

[1] D. Brock, "The Physical Markup Language: A Universal Language for Physical Objects," White Paper, MIT Auto-Id Center, 2001, MIT-AUTOID-WH-003.

[2] G. Borriello, "RFID: Tagging the world," *Commun. ACM* (Guest Editorial to RFID Special Issue), vol. 48, no. 9, pp. 34–37, Sep. 2005.

[3] S. Chalasani and J. Sounderpandian, "RFID for retail store information systems," in *Proc. 10th Americas Conf. Information Systems (AMCIS 2004)*, New York, Aug. 5–8, 2004.

[4] S. Chalasani, R. V. Boppana, and J. Sounderpandian, "RFID tag reader designs for retail store applications," in *Proc. 11th Americas Conf. Information Systems (AMCIS 2005)*, Omaha, NE, Aug. 11–14, 2005.

[5] B. Eckfeldt, "What does RFID do for the consumer?," *Commun. ACM*, vol. 48, no. 9, pp. 77–79, Sep. 2005.

[6] M. Mandviwalla and Z. Asif, "Integrating supply chain with RFID: A technical and business analysis," *Commun. Assoc. Inf. Syst.*, vol. 15, no. 24, Mar. 2005.

[7] M. Ohkubo, K. Suzuki, and S. Kinoshita, "RFID privacy issues and technical challenges," *Commun. ACM*, vol. 48, no. 9, pp. 66–71, Sep. 2005.

[8] M. Rieback, B. Crispo, and A. S. Tanenbaum, "Is your cat infected with a computer virus?," in *Proc. 4th Annu. Int. Conf. Pervasive Computing and Communications (PerCom)*, Pisa, Italy, Mar. 2006.

[9] K. Traub, "Radio frequency identification at enterprise scale," in *Proc. 2004 Computer Measurement Group (CMG-2004) Conf.*, Dec. 2004.

[10] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. New York: Wiley, 2002.

[11] B. Nath, F. Reynolds, and R. Want, "RFID technology and applications," *Pervasive Comput.*, pp. 22–69, Jan.-Mar. 2006.

[12] S. Chalasani and R. V. Boppana, "Software architectures for e-commerce computing systems with external hosting," *Int. J. Comput. Appl.*, vol. 27, no. 3, pp. 190–198, 2005.

[13] A. Juels, "RFID security and privacy: A research survey," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 381–394, Feb. 2006.

[14] S. Garfinkel, A. Juels, and R. Pappu, "RFID privacy: An overview of problems and proposed solutions," *IEEE Security Privacy*, pp. 34–43, May/Jun. 2005.

[15] Demonstration of a Data Architecture for RFID Implementation, Teska Associates, 2006. [Online]. Available: http://rfid.teska.net/standard.jsp.

[16] F. Wang and P. Liu, "Temporal management of RFID data," in *Proc. 31st VLDB Conf.*, Trondheim, Norway, 2005, pp. 1128–1139.

[17] S. Sarma, "Integrating RFID," *ACM Queue*, pp. 50–57, Oct. 2004.

[18] K. Traub *et al.*, The EPCglobal Architecture Framework, EPCGlobal Document, 2005. [Online]. Available: http://www.epcglobalinc.org/standards/Final-epcglobal-arch-20050701.pdf.

Suresh Chalasani (M'91–SM'05) received the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles.

He is an Associate Professor of business at the University of Wisconsin-Parkside, Kenosha, WI, where he specializes in supply chain management systems, e-commerce, healthcare management, and bioinformatics.

Dr. Chalasani was a recipient of multiple research and instructional grants from the National Science Foundation and the University of Wisconsin System.

Rajendra V. Boppana (M'91–SM'99) received the B. Tech. degree in electronics and communications engineering from Mysore University, Mysore, India, in 1983, the M. Tech. degree in computer technology from the Indian Institute of Technology, Delhi, India, in 1985, and the Ph.D. degree in computer engineering from University of Southern California, Los Angeles, in 1991.

Since 1991, he has been a faculty member in the Department of Computer Science at the University of Texas, San Antonio. His research interests are in parallel and distributed computing, performance evaluation, computer networks, and mobile computing and communications. He published extensively and served on the program committees of several conferences in these areas. His current and previous research has been supported by several grants from NSF, DOD, AIA, and other federal funding agencies.