TELKOMNIKA, Vol.11, No.3, March 2013, pp. 1348 ~ 1354
ISSN: 2302-4046

RFID Spatio-Temporal Data Management

Wang Yonghui, Xu Jingke, Wang Shoujin
School of Information and Control Engineering, Shenyang Jianzhu University
*Corresponding author, e-mail: yonghuiwang@sjzu.edu.cn

Abstract

Radio-frequency Identification (RFID) technology promises to revolutionize the way we track items in supply chain, retail store, and asset management applications. The size and different characteristics of RFID data pose many interesting challenges in the current data management systems. In this paper, we provide a brief overview of RFID technology and highlight a few of the spatio-temporal data management challenges that we believe are suitable topics for exploratory research.

Keywords: *RFID, Spatio-Temporal Data, Management*

1. Introduction

RFID technology uses radio-frequency waves to transfer data between readers and movable tagged objects without line of sight. RFID technology has gained significant momentum in the past few years, with several high-profile adoptions. RFID holds the promise of real-time identifying, locating, tracking and monitoring physical objects, and can be used for a wide range of pervasive computing applications. To achieve these goals, RFID data have to be collected, transformed and expressively modeled as their virtual counterparts in the virtual world. But RFID data can't be effectively managed by only using traditional data model because they have their own unique characteristics, such as aggregation, location, temporal and history-oriented, which have to be fully considered and integrated into the data model. Therefore, there are some significant challenges in the current data processing and management systems that must be overcome. The first of them is the enormous amount of low-level data that must be processed on the fly. RFID observations contain redundant information in form of duplicates, which have to be filtered. Moreover, RFID observations have implicit meanings, which have to be semantically transformed and aggregated. Thus, RFID delivers a granularity that needs to be effectively managed from a quality as well as a quantity perspective. This paper aims to formalize the constructive way for efficient RFID data management, filtering and aggregation.

In this paper, we discuss the fundamental characteristics of RFID data in section 2 and suggest a layered system architecture followed by a brief explanation about each layer in section 3. Section 4 mentions the research problems which are associated with each layer separately within three subsections. Section 5 concludes the paper.

2. RFID Spatio-Temporal Data

The operation of an RFID system generates a stream of data resulting from interrogation cycles occurring at recurring time intervals at each Reader. Because of the nature of RFID data, which is significantly different from traditional data warehouse approaches, characteristics of RFID data have to be fully considered in RFID data management systems. Despite of diversity of RFID applications, RFID data share common fundamental characteristics [5], which can be listed as follows:

Simplicity of data: Data generated from an RFID application can be seen as a stream of RFID tuples of the form *(EPC*, *location*, *time*) [13], where *EPC* is an Electronic Product Code which universally identifies an item [14]. *Location* is the place where the RFID reader scans the item, and *time* is the time when the reading took place. As it shows, RFID data does not carry much information. In order to transform this raw data into a form that enterprise applications can use, several levels of inferences must be done.

Received September 28, 2012; Revised January 13, 2013; Accepted January 24, 2013

Large volume of data: One of the biggest issues in RFID is to deal with large volume of data. The volume and velocity of RFID data will exceed the capacity of existing technology infrastructure. As an example, Wal-Mart is expected to generate 7 terabytes of RFID data per day [8]. Since each tag is tagged periodically and the data about tag EPC, location and reading time will be continuously produced in the system, even modest RFID deployments will generate gigabytes of rapidly changing data a day.

Inaccuracy: One of the primary factors limiting the widespread adoption of RFID technology is the inaccuracy of the data stream produced by RFID readers. The observed read rate in real world RFID deployments is often in the 60-70 % range [16], [17]. Unfortunately, such error rates render raw RFID data essentially useless for the purpose of higher level applications. Thus, we need to clean this data before feeding our system with such unreliable data. As a result, the data in RFID are generally inaccurate.

Spatial and Temporal: RFID applications dynamically generate observation (e.g. regular changing of the location of tagged items) and the data carry state changes[18]. Thus, in RFID data management, it is essential to model all such data by an expressive data model suitable for application level interaction including tracking and monitoring. Given the facts that not only RFID readers but also the tags can now be built into PDAs, Cell Phones, and moving objects, both tagged items and readers are in constant movement.

3. System Architecture

Figure 1 suggests a layered architecture for managing RFID data. The lowest level consists of RFID tags attached to different items. The next layer, called Data Capture layer, is the layer of RFID readers. The data emerging from this layer can be considered as RFID data streams of tuples of the form of (EPC, location, time).

RFID data concentrator, the core layer of the architecture, is responsible for mapping the low-level data streams from readers to a more manageable form that is suitable for application level interactions. There are three primary elements: RFID middleware, event processing, and an in-memory data cache. RFID middleware provides the interface for applications to receive events from RFID readers. RFID middleware systems typically deployed between the readers and the applications in order to correct captured readings and provide clean and meaningful data to application logic. Event processing handles high-volume, high-performance flows of events by organizing raw events into pipelines. Pipelines allow the events to be categorized, then processes those categories with a set of simple tasks. By performing a large number of operations on data in small “bursts,” overall throughput is increased, and the average speed that any individual event can be processed is increased, as well. Finally, an in-memory database, or data cache, makes the concentrator work in real time. In-memory data management techniques are crucial to accommodate the real-time nature of RFID.

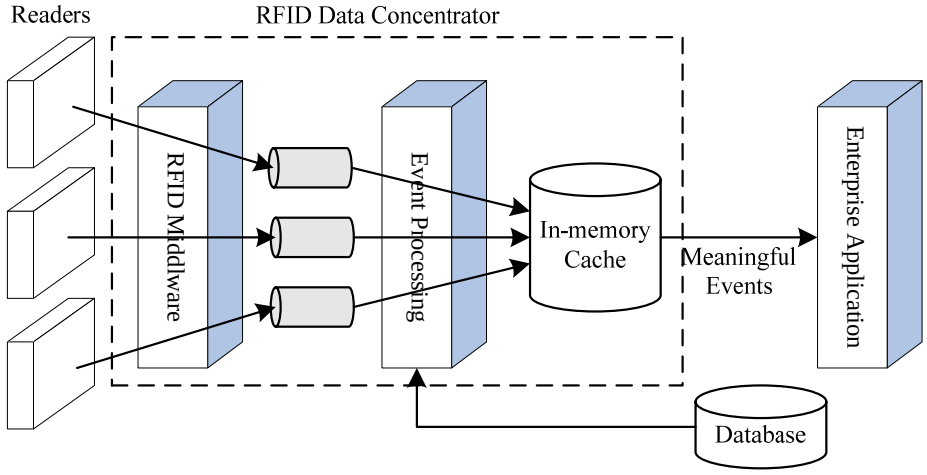

Figure 1. RFID System Architecture

Applications on the enterprise application level can interact with the RFID data concentrator layer by issuing simple queries as well as by installing standing queries that result in a stream of matching data. This level supports the business processes of enterprise applications such as Supply Chain Management (SCM) or Customer Relationship Management (CRM).

4. RFID Data Management

Based on the RFID system architecture described in section 3, we find RFID data management is basically performed by RFID data concentrator layer. Subsequently, we will discuss how every component of RFID data concentrator manages RFID data in detail.

4.1. RFID Middleware

The RFID middleware layer is responsible for coordinating multiple tagged objects and cleaning incoming data before sending to the next layer, as well as detecting some simple events and reporting them to the management systems.

As we mentioned earlier, one of the primary factors limiting the widespread adoption of RFID technology is the inaccuracy of the data stream produced by RFID readers. So as a result of that we have big load of unreliable data which is useless for the purpose of higher level processing. Therefore, we need to clean this unreliable data which we call dirty data. Data cleaning is a common problem in most of the data management systems. For instance, in data warehouses it is usually an off-line, centralized, iterative, and sometimes interactive process that focuses on a small set of well defined tasks [20]. In contrast, RFID data are time sensitive, e.g., inventory control. This demands that the data must be cleaned online before they are streamed to applications. In RFID data management, dirty data appears in three general forms: 1) missed readings 2) unreliable readings 3) data redundancy.

4.1.1. Missed and Unreliable Readings

Missed and unreliable readings problems are very common in RFID applications and often happen in situations of low-cost, low-power hardware and wireless communications, which lead to frequently dropped readings or with faulty individual readers. These problems have been already discussed in different research works [16, 21, 22].

In general, receptor-based applications are not interested in individual readings or individual devices in terms of time and space, but rather in an application-level concept of temporal granules and spatial granules [21]. The concepts of temporal granule and spatial granule have been defined in [21]. Jeffery et al introduce temporal granule as the smallest unit of time that the application operates. For instance, in a retail scenario when an application continuously monitors the count of items on each shelf, the temporal granule would be each 5 seconds. Spatial granule groups items based on some spatial categories which will be the lowest level of spatial granule at which the application operates, such as a shelf in a retail scenario. ESP (Extensible Sensor Stream Processing) is a general technique for sensor stream data cleaning [21]. ESP has five stages each of which is responsible for a different logical aspect of the data. However, ESP is a general technique for sensor data stream and we believe ESP may fail in the following two cases:

- It may fail to find the spatial granule for a supplier warehouse where the location of items are changing regularly. So it is difficult to group the items in order to define coarser spatial granule.
- It may fail when the size of the temporal granule (sliding-window) has not been chosen correctly. The window size must be large enough to smooth the lost readings (False-negative) but small enough to accurately capture tag movements (False-positive).

The first failure refers to the evolving spatial and temporal nature of RFID data when the location of items is constantly changing. For example, in a supplier warehouse where the tagged items are moving on the conveyor belt, it is hard to find the correct unit of spatial granule, which also, increases the chance of missing readings for readers. However, in general, items are not always seen in isolation, but often travel together. A common example is that of pallets and cases. Two cases on the same pallet will tend to both be detected by RFID readers at the same time. Similarly, the pallet will be detected along with the two cases. All together they form an aggregate or containment. Item aggregation provides an opportunity to enhance the

reliability of RFID information by referencing the existence of individual items in terms of the location of their container.

The second failure has already been addressed in [16]. Jeffery et al proposed SMURF(Statistical sMooting for Unreliable RFID data), the first declarative, adaptive smoothing filter for cleaning raw RFID data streams. SMURF uses a sampling-based approach to find the right temporal window size for the applications. Unlike traditional techniques, SMURF does not expose the smoothing window parameter to the application; instead, it determines the right window size automatically and continuously adapts it over the lifetime of the system based on observed readings.

4.1.2. Data Redundancy

The redundancy problem is recognized as a serious issue in RFID and sensor networks. We discuss this issue at two different levels:

Redundancy at reader level: This problem occurs when an item is in the vicinity of more than one reader which are simultaneously sending signals to the item. For example, as Figure 2 shows, readers R1, R2, R3 and R4 are redundant since the tags covered by each is covered by at least one other reader. The optimal solution requires only R2 to be active while the other readers may be turned off. However the problem of finding the optimal solution for the redundant reader elimination is NP-hard [23]. In [23], Carbunar et al proposed an algorithm called RRE which is a randomized, decentralized, and localized approximation algorithm for the redundant reader elimination problem. The RRE algorithm has three different steps. First, it detects the set of RFID tags placed in the vicinity of a reader. Second, each RFID reader attempts to write its tag count (number of covered tags) on to all its covered tags. A tag placed in the vicinity of several readers will overwrite the count stored on behalf of a reader only if the new value is larger. The reader that issued the highest count for a tag, locks the tag. In the final step of RRE, each reader sequentially queries all its covered tags to discover the ones it has locked. A reader that has not locked any of its covered tags is declared redundant.

The RRE algorithm assumes that the position of readers will be constant for a long period of time. This assumption may not be held in applications such as the supply chain where the position of readers may change in order to optimize the business process [24].

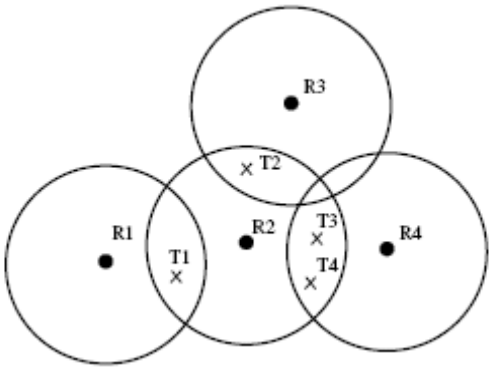

Figure 2. Redundancy reader example

***Redundancy at data level*:** RFID data are usually regarded as an example of streaming data, and as result, the redundancy on the data level has always been handled in the general way of dealing with data streams. However, we believe, RFID data has its own peculiarities which have been largely ignored. We can compare RFID data and general data streams to illustrate some special features of RFID data. First, the data volume for RFID is usually larger (some readers can perform over 100 readings each second). Second, the RFID data on average is less useful than other data streams. For example, in traffic monitoring and financial applications, every record might be useful for further analysis. On the other hand, in the case of RFID data, we should be able to identify those data that have been repeatedly read

multiple times. The less useful part of RFID data is the data that are continuously reported after the initial reading. For instance, in supply chain management, a tagged item can move to the shelf and sit the whole day on the shelf and send the data to the RFID management system constantly. But, from the management point of view, the most useful information for event detection is when the tagged item moves to the shelf and when the item is removed by a customer. Therefore, it is necessary to reduce RFID data redundancy before processing.

4.2. RFID Event Processing

Any current business application that is exposed to a network of RFID readers, each of which is capturing tag IDs from many products and containers, will be required to handle event streams. These event streams will contain some meaningful events, but also many events of little relevance to the core processing of the application. The event streams will also contain events that are reported in error or which contain errors. Managing this inbound data will require implementation of a processing tier that can handle such event streams, filter the meaningful from the irrelevant, and generate a more concise and pertinent set of events. That processing operation is one of the core principles of complex event processing. For, it is not the events in a complex event processing operation that are complex in nature. Rather it is the ability to derive meaning from an event by understanding the event – within the context of other events as well as other forms of business data – that provides the "complexity" of complex event processing.

Furthermore, while event streams are often rapid, that should not imply that they are necessarily transient. Complex event processing must also recognize that not just the current event stream determines current state or activity. Event processing engines must have access to both current event streams and a persisted history of events in order to measure, compare and contrast activity over time.

Above the standard RFID architecture of tags, readers and ALE middleware, a successful, scalable CEP architecture consists of:

(1) Event Engine
   a) Once events move through the collection phase, they can be processed, according to type. The event engine executes event pipelines that can apply the various forms of filtering, aggregation and transformation according to event type.
   b) A pipeline is a programmatic structure made up of individual components. Each component performs some form of processing (gathering, filtering, grouping, transforming) on an event and streams its results to the next component. A pipeline can consist of a single process or multiple processes each performing a single step in a complex transformation. The result is a high performance, adaptable event processing construct called a pipeline.

(2) Event Database
   A repository for events. Rather than allow events to be simple transitory entities, the event engine has the ability to record each event, either in its raw form or in a processed form, in a scalable, distributable event database. This database forms a historical repository for event processing and for external applications looking to monitor and measure event processing over time and against standard metrics.

(3) Event Server
   Provides a flexible interface to the event engine, allowing integration of different asset coding standards (GRAI vs. SGTIN vs. UCC-128), and event query tools. Through the event server, the event engine can be used to process not only RFID feeds and serve a single set of RFID-driven applications, but also feeds from standard bar code readers, database file/table-driven inputs and provide data that can be syndicated across multiple applications via a JMS subscription model.

(4) Event Caching
   The event cache provides the real-time access to a host of client applications seeking to query the event database. The cache architecture allows for many models of deployment (point-to-point via queues or topic-based publish and subscribe) depending on the required information systems integrations.

(5) Event Query Language (EQL)
   a) The EQL provided with the architecture will allow external applications to retrieve data from the event database. It must provide a standard interface for all calling applications to ensure consistent use of the event history.

b) For RFID, the typical queries must support lists of tags passing a named reader in a given time period, a list of readers through which a particular tag has passed within a given time period, and counts of events pertaining to a particular tag, reader, organization or time period.

4.3. In-Memory Cache

The RFID data cache is based on a real-time in-memory event database (RIED). The RIED manages data much like virtual memory systems in modern operating systems. Virtual memory systems support uniform memory references to data, whether that data is located in primary or secondary memory. Data is held in memory on a most recently used basis and paged out when the space is required by another program. Using this virtual memory mapping architecture (VMMA), the database moves unused data to a secondary storage area where it can be persistent or transient, depending upon the needs of the RFID application. When the application needs to access an object that is not in primary memory, the RIED transfers the page containing the referenced data automatically — possibly together with adjacent pages — to primary memory that serves as the application's cache. This cache acts as a pool of event objects that have been recently used in the virtual memory of the application's host. The database's direct mapping mechanism uses lock caching (as well as data caching) and call back locking to ensure cache coherency with no runtime overhead after the initial access.

The net effect of this design is that all data is kept in memory as much as is practical, thus providing the kind of in-memory performance that is required by RFID applications. The advantages of this approach over simplistic in-memory data management schemes include:

(1) Size of the database
A simplistic in-memory data manager is constrained by the size of addressable memory for that process. With RFID VMMA, there is no problem if the amount of event data requires more memory than is available in its process address space. The RIED storage manager transparently pages data from secondary to primary memory as soon as it is required.

(2) Resilience to system interrupts
Since a copy of all EPC data is held in secondary memory in a transactionally consistent way, the state of the application can be automatically recovered when it comes back up.

(3) Data Sharing
To avoid concurrency conflicts and reduce memory contention, simplistic in-memory data managers will keep one or more snapshots of the event data in memory so that the database can be queried without interfering with the real-time streaming. This may be appropriate for some applications, but because it reduces the size of available primary memory, it is often not ideal. The RIED provides many options for sharing in-memory data across multiple processes and machines. This allows external applications access to data without interfering with the real-time data streaming in from the readers.

(4) Data Distribution
The database's Cache-Forward Architecture provides a simple, transparent, transactionally consistent data distribution mechanism so that the application for RFID can efficiently collect data in local caches.

5. Conclusion

RFID technology promises to revolutionize the way we track items in supply-chain, retail store, and asset management applications. In this work, we have provided an overview of the history of this promising technology and highlighted the RFID spatio-temporal data management challenges that we believe are suitable topics for exploratory research. The main part of our work focused on discussing the research problems which are associated with the different layers of our suggested system architecture for RFID spatio-temporal data management.

Acknowledgment

The work is supported by the Science and Technology Funds from Liaoning Education Department (L2012212).

References

[1] L Sullivan. Target Meets With Suppliers About RFID Plans. *Information Week*. 2004.
[2] FSI Metro Group. http://www.future-store.org/.
[3] J Collins. DOD Tries Tags That Phone Home. *RFID Journal*. 2006.
[4] P Harrop. RFID in the Postal Service. *MoreRFID*. 2005.
[5] J Collins. Boeing Outlines Tagging Timetable. *RFID Journal*. 2006.
[6] C Swedberg. Hospital Uses RFID for Surgical Patients. *RFID Journal*. 2005.
[7] RB Ferguson. Logan Airport to Demonstrate Baggage. *Passenger RFID Tracking*. eWeek. 2006.
[8] II Reports. The Internet of Things. http://www.itu.int/pub/S-POLIR. IT-2005/en, 2005.
[9] C Swedberg. Survey says manufacturers drive RFID uptake. *RFID Journal*. http://www.rfidjournal.com/article/articleview/2371/, May 2006.
[10] Walmart supplier information: Radio frequency indentification usage. http://www.walmartstores.com, 2005.
[11] C Heinrich. *RFID and Beyond: Growing Your Business Through Real World Awarness*. Wiley Publishing, Inc. 2005.
[12] M Lee, F Cheng, and Y Leung. A quantitive view on how RFID will improve a supply chain. *Technical Report RC23789 (W0511-065), IBM Research Center*. 2005.
[13] H Gonzalez, J Han, X Li, and D Klabjan. *Warehousing and analyzing massive RFID data sets*. In Proc of the International Conference on Data Engineering (ICDE06). 2006: 1-10.
[14] EPC tag data standard version 1.1. Technical report. EPCGlobal Inc. 2004.
[15] M Palmer. Principles of efferctive RFID data mangment. *Enterprise Systems*. 2004.
[16] S Jeffery, M Garofalakis, and M Franklin. *Adaptive cleaning for RFID data streams*. Proceedings of the 32nd international conference on Very large data bases (VLDB). 2006: 163–174.
[17] C Floerkemeier and M Lampe. Issues with RFID usage in ubiquitous computing applications. *Pervasive Computing: Second International Conference, PERVASIVE*. 2004.
[18] F Wang and P Liu. Temporal management of RFID data. *Proceeding of the VLDB05*. 2005: 1128-1139.
[19] C Bornhovd, T Lin, S Haller, and J Schaper. *Integrating automatic data acquisition with business processes experiences with saps auto-id infrastructure*. Proceedings of the 30th VLDB Conference. 2004; 1182-1188.
[20] E Rahm and H Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull*. 2000; 23(4):3-13.
[21] S Jeffery, G Alonso, M Franklin, W Hong, and J Widom. A pipelined framework for online cleaning of sensor data streams. *International Conference on Data Engineering (ICDE 06)*. 2006; 140.
[22] S Jeffery, G Alonso, M Franklin, W Hong, and J Widom. Declarative support for sensor data cleaning. *Pervasive*. 2006.
[23] B Carbunar, M Ramanathan, M Koyuturk, C Hoffmann, and A Grama. Redundant reader elimination in RFID systems. *Sensor and Ad Hoc Communications and Networks*. 2005: 276-184.
[24] S Chawathe, V Krishnamurthy, S Ramachandran, and S Sarma. *Managing RFID data*. Proceedings of the 30th VLDB Conference. 2004; 1189-1195.
[25] Y Zhuge, H Garcia-Molina, J Hammer, and J Widom. View maintenance in warehousing environment. *ACMSIGMOD*. 1995; 316-327.
[26] Iswanjono, Budiardjo Bagio, Ramli Kalamullah. An algorithm for predicting the speed of traffic light violators. *TELKOMNIKA Indonesian Journal of Electrical Engineering*.. 2011; 9(1): 55-64.
[27] Mardiyono, Suryanita Ren, Adnan Azlan. Intelligent monitoring system on prediction of building damage index using neural-network. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(1): 155-164.