Applied Mechanics and Materials
ISSN: 1662-7482, Vols. 773-774, pp 823-827
doi:10.4028/www.scientific.net/AMM.773-774.823

Submitted: 2014-09-14
Accepted: 2014-11-14
Online: 2015-07-15

An Efficient Approach to Processing Massive RFID Data in Supply Chain

Hairulnizam Mahdin[1, a], Mohd Farhan Md Fudzee[2,b] , Azizul Azhar Ramli[3, c], Mohd Aizi Salamat[4,d] and Shahreen Kasim[5,e]

[1,2,3,4,5]Faculty of Computer Science and Information Technology, Univeristi Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia

[a]hairuln@uthm.edu.my,mail, [b]farhan@uthm.edu.my, [c]azizulr@uthm.edu.my, [d]aizi@uthm.edu.my, [e]shahreen@uthm.edu.my

Keywords: RFID, Bloom filter, data filtering, sensor, supply chain

Abstract. Radio Frequency Identification (RFID) is widely used to track and trace objects in supply chain management. However, massive uncertain data produced by RFID readers are not suitable for directly use in RFID applications. This is due to repetitive readings which are unnecessary because it contains only the same information. Thus, an approach to remove repetitive readings in supply chain is paramount to minimize massive data storage that could affect query performances. We propose a simplified approach, which is suitable for a wide range of application scenarios. Experimental evaluations show that our approach is effective and efficient in terms of the removing duplicate readings and compressing the massive data significantly.

Introduction

Radio Frequency Identification (RFID) system is item detection tools that can complete the tasks automatically compared to the traditional barcode [1]. It does not need line of sight for its item to be detected. Whenever the tags were in the reader's vicinity, it will be automatically be read. The automatic feature has enabled the verification of large quantity items shipment which was impossible before. However, RFID readers will keep sending it readings whenever the tagged item is still in its vicinity [2]. This scenario has cause RFID to produce a lot of duplicate readings. These readings were actually unnecessary because it contains only the same information [3]. In the field of supply chain, the important readings are only the first and the last time the items were detected in the reader's vicinity [4]. Otherwise, there will be a lot of duplicate readings even in a very short time because of the capability of the RFID readers. Duplicate readings can slow the system's query performance [4] because the pile of data getting higher over the time. For example a small supermarket that has 10,000 tagged items will return 10,000 tuples for each reading cycle. If there are 10 reading cycles in an hour, the number of tuples produced within the time is 100,000 lines, 800,000 lines per day and 24 millions lines per month. If we going to query how many detergent of a particular brand on particular day were available in the store, it takes the query process some time to dig for the answer from such big data. Here, duplicate readings were the most responsible for huge volume of the data and it does not contribute significantly to the system information. Therefore it is paramount to filter those duplicate readings to ensure only readings that are needed were left in the system. This paper proposes an approach to filter duplicates readings based on Bloom filter. Bloom filter is very space efficient approach in data filtering and can provide very fast searching results with some tolerance over false positive. This paper is organized as follows: section 2 describes RFID system architecture, section 3 is on related works, section 4 is the details on the algorithm and performance analysis and section 5 is the conclusion.

System Architecture

Fig. 1 shows the four main layers of RFID. The top layer consist of tags, the second layer is consists of readers, third layer is the middleware and the fourth layer is the database and enterprise applications. RFID have three types of tag which can be passive, active or semi passive [5]. Tags contain unique ID that used to identify physical object. The main different between the tags is their source of power to operate. Passive tag does not have its own power source. It harvests the power from the signal beacon by the reader. Once it gets enough power, it will send it signal back to the reader. Active tags have its own battery on board which make it more expansive than passive tag. By having its own battery it can achieve longer range to transmit the signal. However because of the battery it has limited lifetime. Semi passive tag also has battery on board, which is used only for internal processing. It still requires the signal from the reader to perform the communication.

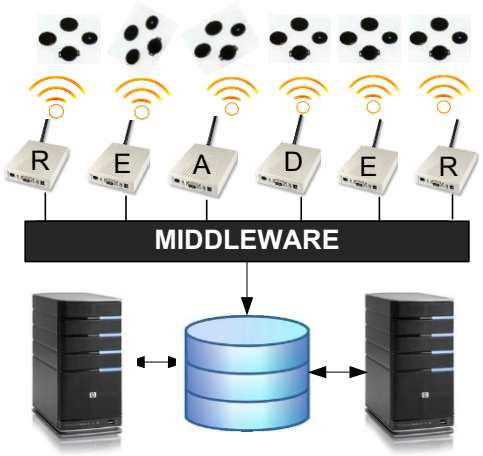

Fig. 1: RFID-enabled system architecture

At the second layer, there are two types of reader: fixed and mobile reader. Fixed reader is locate at fixed location and can have more than one antenna to cover wider vicinity. It will beacon it signal to detect the objects in its area and the process is repetitive regardless the presence of the tag. A tag can be read more than once as long it resides in the reader vicinity. A tag also can be read by more than one reader if it resides in the overlapped vicinity. Mobile reader also works like fixed reader but it uses battery to support its mobility.

After the reader collects reading from the tag, it will be send to the middleware. Middleware is the place where the filtering process will be done [6]. Not every reading is useful to the application. Usually, only the first and the last reading are important to the system to mark the arrival and departure of the item. In the literature review, we present some of the important previous work relating to the filtering RFID readings based on Bloom filter [7].

Literature Review

In this section, we will review the existing duplicate RFID data filtering approaches that are related to the topic. In our previous work [2], a single filter name Comparison Bloom Filter (CBF) to only keep single reading on a tag that have been read by more than one readers. This filter also can use to remove duplicate readings. However, this approach does not record the arrival and the departure of the tag because the limitation of Bloom filters. Bloom filter can only check whether an element has been resides in its filter only. It cannot retrieve the original form of the element inside its filter. This is because the element has been hashed and represent by bit array position in the filter. Bloom filter that achieve space efficiency by allowing a small and acceptable false positive rate (FPR). A false positive occurs in Bloom filter when a reading is detected incorrectly as a duplicate in the same window. Suppose that n distinct elements are inserted into the Bloom filter, the FPR is

$$FPR = 1 - (1 - \frac{1}{m})^{kn})^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \quad (1)$$

where k is the number of hash functions and m is the number of counters in the integer array. When m and n are given, FPR is minimized when

$$k = \frac{m}{n}(\ln 2) \approx 0.7\frac{m}{n} \quad (2)$$

Thus, we have

$$FPR \approx 2^{-k} \approx 0.6185^{\frac{m}{n}} \quad (3)$$

We want to use the efficiency of CBF to remove the duplicate readings from the RFDI data stream. To do this, we need to modify CBF where it will be able to store the arrival and the departure time for the tag. There will be only two reading stored for the particular reading. Other duplicate readings will be removed. The algorithm is explained in the next section.

```
INPUT: TID,TIME
1: IF (Elements == True) THEN
2:     BDF [] = {0}
3: ENDIF
4: FOR (each incoming TID) DO
5:        DUPSTATUS=0;
5:        FOR (i = 1 TO k) DO
6:           Pos ← Hash[i](TID)
7:           IF (BDF[Pos] == 0) THEN
8:                CounterNum [i] ← Pos
9:                  DUPSTATUS=0;
10:           ELSE
11:               CounterNum [i] ← Pos
12:                 DUPSTATUS ++;
13:          ENDIF
14:       ENDFOR
15:       FOR (i = 1 TO k) DO
16:          Pos ← CounterNum [i]
17:          BDF[Pos] ← C
18:       ENDFOR
19:       IF (DUPSTATUS==0)
20:           UPDATE LIST<-TID, TIME ARRIVAL
21:       ELSE
22:                    UPDATE  LIST<-TID,  TIME
DEPARTURE
23:       ENDIF
24: ENDFOR
```

Fig. 2: Baseline Duplicate Filtering (BDF) Algorithm

Duplicate Filtering Approach

The duplicate filtering approach proposes in this paper preserve the arrival and departure time of a tag. Therefore only two readings are saved for each tag. The readings between the two were ignored. To achieve that, recent reading on the tag is always updated with the new one. When there

is no new reading is made on a tag after certain time, the tag is declared leaving the area, and the final reading is send to the database. Algorithm Baseline Duplicate Filtering (BDF) is presented in Fig.2.

In Fig. 2, steps 1-3 implement the filtering windows for Bloom filter. At step 1, the algorithm checks if the number of readings stored in the filter is met. If it is true, all CBF counters will be reset to zero (step 2). This can be easily converted to time unit if required. This is to avoid BDF from being full. Otherwise BDF will experience what is called as 'full Bloom filter [7]. Next at step 4 to 14, each tag ID (TID) is hashed to the filter. In step 5, the duplication status (DUPSTATUS) is set to zero each time a reading is going to be hashed. At step 7 the algorithm checks whether the counter is 0 or not. If the counter is zero that's mean this is a new tag and never been recorded. The DUPSTATUS is set to 0. Otherwise, the DUPSTATUS is increase by 1. In step 19 to 23, a list is provided to record the TID and its time. If the DUPSTATUS is 0, the time that comes with the TID is its arrival time, else it was the latest time. After certain time, record from the list will be sent to the database where the latest time of the tag will be it's time of departure. By using this approach only two readings for each tag are will be recorded in the database and dismiss all other unnecessary duplicates.

Preliminary Performance Evaluation

The performance of the BDF has been compared to others several approaches in terms of number of unfiltered duplicate readings. The result obtained shows that BDF filter more duplicates than other approach which is Baseline [8]. Baseline contains more unfiltered duplicates because it uses sliding windows which have small window size. When the window size is small, the similar readings might come again into the window after a while and this left the reading unfiltered. The size of the sliding windows cannot be too big because it will increase the processing time. This is different with BDF where we use Bloom filter that can have longer reading cycle without increasing the processing time. The performance of BDF with CBF is the same because both of them were based on Bloom filter approach. However CBF does not preserved the departure time of the tag.

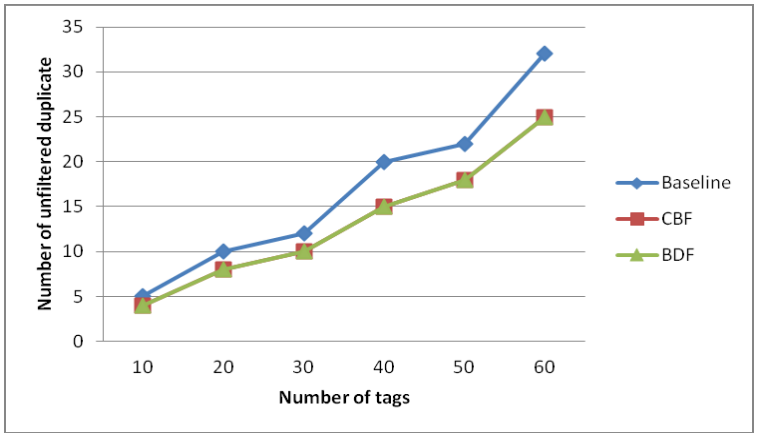

Fig. 3: Comparison of duplicate filtering performances

In our future work, we will include the importance of preserving the arrival and departure time of a tag. It can give precise description on the events occurred associated with the tag.

Conclusion

RFID will become the key technology in the modern supply chain. However, there are still issues need to be resolved with the technology to make this happens. The data management need to be very efficient as the reader is able to produce automatically a lot of readings on tags that resides in its vicinity. Only useful readings need to be preserved while the others can be filtered to ensure

there is no negligence in system performance. In this paper we proposed Baseline Duplicate Filtering Approach, which preserved the arrival and departure time of a tag in a system. By only keeping both readings, other readings between them on the same tag were automatically ignored and removed. The readings being removed were duplicate and is unnecessary to keep as the can affect the system query performance. A preliminary research shows that other approaches only preserves single reading either the first time occurred or the latest one. Both readings were needed as it can be used in many ways such as to do even predictions and recording purposes.

Acknowledgments

This work is sponsored by Ministry of Education through ERGS grant vote E054 and Gates IT Solution Sdn Bhd. We also would like to thank Junaidah Jailani and Sara Nadine for their diligent support.

References

[1] J., Worapot, S. Arch-int, and Y. Li., Business process analysis and simulation for the RFID and EPCglobal Network enabled supply chain: A proof-of-concept approach, Journal of Network and Computer Applications 34, no. 3 (2011) 949-957.

[2] M., Hairulnizam, and J. Abawajy, An approach for removing redundant data from RFID data streams, Sensors 11, no. 10 (2011) 9863-9877.

[3] L., Sih-Ying, H.-Y. Kung, C.-H. Chen, and W.-H. Lydia Hsu, An Efficient RFID Data Processing Scheme for Data Filtering and Recognition, International Journal of u-and e-Service, Science and Technology V(2012).

[4] G., Hector, J. Han, H. Cheng, X. Li, D. Klabjan, and T. Wu, Modeling massive RFID data sets: a gateway-based movement graph approach, Knowledge and Data Engineering, IEEE Transactions on 22, no. 1 (2010) 90-104.

[5] B.-M., Ricardo, J. Garcia-Hierro, L. Ruiz-Garcia, T. Jiménez-Ariza, J. I. R. Villalba, and P. Barreiro, Assessing the dynamic behavior of WSN motes and RFID semi-passive tags for temperature monitoring, Computers and Electronics in Agriculture 103 (2014): 11-16.

[6] Z. Jianglong, J. Huang, D. He, Y. Leng, and S. Xiao, The Design of RFID Middleware Data Filtering Based on Coal Mine Safety, In The Proceedings of the Second International Conference on Communications, Signal Processing, and Systems, Springer International Publishing, 2014, pp. 859-867

[7] Bloom, Burton H., Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13, no. 7 (1970) 422-426.

[8] B., Yijian, F. Wang, and P. Liu, Efficiently Filtering RFID Data Streams, In CleanDB. 2006.