An Approach to Filtering Duplicate RFID Data Streams

Hairulnizam Mahdin[1] and Jemal Abawajy[2]

[1] Faculty of Comp. Sc. & Info. Tech,
University of Tun Hussein Onn Malaysia, Johor, Malaysia
hairuln@uthm.edu.my
[2] School of Information Technology
Deakin University,
Victoria, Australia
jemal@deakin.edu.au

Abstract. In a system where distributed network of Radio Frequency Identification (RFID) readers are used to collaboratively collect data from tagged objects, a scheme that detects and eliminates redundant data streams is required. To address this problem, we propose an approach that is based on Bloom filter to detect duplicate readings and filter redundant RFID data streams. We have evaluated the performance of the proposed approach and compared it with existing approaches. The experimental results demonstrate that the proposed approach provides superior performance as compared to the baseline approaches.

1 Introduction

RFID technology enables capturing large volumes of data at high speed and can be used for identifying, locating, tracking and monitoring physical objects without line of sight. This capability makes RFID technology desirable for many applications such as supply chain management. Large-scale RFID-enabled systems are composed of multiple networked RFID readers and physically distributed tags, the middleware software that processes and manages the information gathered, and the enterprise applications that implements the enterprise business logic, and the enterprise database management systems.

In this paper, we address the problem of detecting and eliminating redundant data streams in a system where distributed network of RFID readers are used to collaboratively collect data from tagged objects. This is important as multiple readers are used in many applications for a variety of reasons such as to increase the reading reliability [1]; to read objects passing through different doors at warehouse [2]; in supply chain [3] and object's location sensing [4]. However, reading tagged objects by multiple readers could create duplicate readings when multiple readers read the same tagged object simultaneously. The problem of duplicate reading is common in RFID and an approach that efficiently detect duplicate readings is required [5].

In this paper, we propose an approach that is based on Bloom filter [6] for detecting and filtering redundant data from RFID data streams. The performance of the proposed approach is studied and the experimental results demonstrate that the proposed approach provides superior performance as compared to the baseline approaches.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 explains the proposed algorithms in detail. Section 4 presents the performance analysis of the proposed algorithm. We use the sliding windows-based approach in [7] as well as the approach proposed in [8] to compare the performance of the proposed algorithm. We also discuss the results. The conclusion is presented in Section 6.

2 Related Works

Figure 1 shows RFID-enabled system of interest. The system is assumed to contain multiple networked RFID readers (i.e., reader_1, reader_i, reader_n) deployed to collaboratively collect data from tagged objects. In this paper, we focus on passive RFID tag. These tags are widely used especially in supply chain because of its cheap prices. The RFID readers query tags to obtain data and forward the resulting information through the middleware to the backend applications or database servers. The applications then respond to these events and orchestrate corresponding actions such as ordering additional products, sending theft alerts, raising alarms regarding harmful chemicals or replacing fragile components before failure.

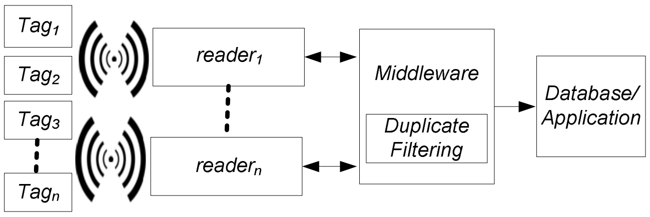


Fig. 1. RFID-enabled system architecture

In RFID-enabled systems, a reading can be defined as duplicate if it is repeated and did not give new information to the system. Redundant readings are useless and should be removed from the data stream without affecting the system. RFID data duplication problem can be discussed at data level or reader level [5]. Duplicate reading at a reader level occurs when an object is covered by more than one reader. In contrast, duplicate reading at data level occurs when RFID readers can repeat reading of the same object as long as the object is within its read range.

In this paper, we focus on reader level data duplicate filtering problem. There are many approaches proposed to filter data level duplicate reading [7][8][9][10][11]. An approach that uses a sliding window to group a number of readings and compare them with each other is discussed in [7]. Only single occurrence from a number of similar readings will be output from that particular window. Another way to remove data level duplicate readings is by using query [9][10]. Query is imposed on the database records to retain only single reading that will represent the others in the given time frame. In [11], new record is made up to represent all other similar reading by specifying the initial and last time of detection. An approach that deferred reading output is discussed in [9][10][11]. This approaches is not suitable for use with applications that demand real time processing.

There are also several approaches proposed in the literature to filter reader level RFID data duplicates. An approach based on a radio frequency absorbing material to controls the RFID propagation from dispersing to neighboring reader's area is discussed in [2]. The material is placed between readers to prevent a reader from reading neighboring tags. However, this solution is not feasible in most application because of design constraint and its high cost. Another approach based on complex computations for detecting the location of the tags from the reader is discussed in [12]. Our aim is to identify which reader should preserve the readings so every reader will only report on their own objects.

Approaches that explore the Bloom filter [6] for filtering duplicate data in RFID has recently emerged in the literature [8][13]. An approach that used the original Bloom filter to remove the duplicate is discussed in [8]. The main idea of the standard Bloom Filter (BF) is to represent an element in a form of positive counter in a bit array of size m using k number of hash functions. All bits in BF are initially set to 0 and will be replaced by 1 when it is hashed by the elements. To test whether an element is a member of a set, the element will be run through the same hash functions used to insert the elements into the array. The element is said to be the member of the set if all the bits in which the element was mapped to is positive. For each new element, the corresponding k bits in the array are set to 1.

Two approaches were proposed in [8]: eager and lazy approach that use Bloom filter to filter duplicate data. Generally, when a new reading comes at local reader, it will be inserted in the Bloom filter. The filter then will be sent to the central filter for update. Central filter is coordinating readings from all the readers under its network. In eager approach, the copy of the Bloom filter will be sent to every other reader to avoid the same reading from entering through them again. However it is too costly to update all the reader every time new reading arrives. In lazy approach, only reader that sends new reading will have new copy of the Bloom filter from the central filter. In our approach, we focus only filtering at the central filter to preserve the reading only to the authorized reader. The work in [14] filters duplicate readings at a single reader. In contrast, the work presented in this paper takes into account multiple distributed RFID readers. Thus our work can be considered as complement to these previous works.

The proposed algorithm is based on the Counting Bloom filter [15], which was introduced to allow counting and deleting operations that cannot be performed with the standard Bloom Filter [6]. In the proposed approach, if the new readings on the same object from different readers have a higher number from the previous reader, the new reading will replace the old reading in the filter. In this paper, we model the relation between the reader read rate and the distance between the reader and the tag based on the work of [16]. The reader detection range can be classified as a major detection region and a minor detection region. Tags read inside the major detection area will have about 95% read rate while the tag read in the minor detection region will have about 5% read rate [16].

3 RFID Data Stream Filtering Scheme

Double Comparison Bloom Filter (DCBF) uses CBF1 and CBF2 filters to represent the count of the readings and the reader identification (RID) respectively. Fig. 2 shows the pseudo-code of the DCBF algorithm. The input to the DCBF is the reading count for each tag (C), the tag identification (TID) and the reader identification (RID). The reading count (C) is needed to compare which reader (based on RID) have the higher reading on the tag (identify through TID).

In step 1, reading count for each tag is done at each reader and is sent with the TID and the RID to the global filter which will run this algorithm. For step 2-8, each incoming TID will be hashed and checked on its condition. If the hashed counter have 0 values or smaller than C, or RID is the same in CBF2 hashed counter, the position will be retained in an array named CounterNum. If one of the hashed counters did not satisfy the condition in step 4, the algorithm will exit (step 7) from all loop and start back to step 1 to receive new reading. If all hashed counters satisfy the condition in step 4, step 10-14 is carried out where the new value of C is stored in the CBF1 counter and RID in CBF2 counter.

Algorithm DCBF

```
INPUT: C, RID, TID
BEGIN
1:  FOR (each incoming TID)  DO
2:        FOR (i=1 TO k)  DO
3:            Pos ← Hash_i(TID)
4:            IF (CBF1 [Pos] == 0) | | (C > CBF1[Pos]) | | (RID > CBF2[Pos])  THEN
5:                 CounterNum [i] ← Pos
6:            ELSE
7:                 EXIT
8:          ENDIF
9:       ENDFOR
10:      FOR (i=1 TO k)  DO
11:          Pos ← CounterNum [i]
12:          CBF1 [Pos] ← C
13:          CBF2 [Pos] ← RID
14:      ENDFOR
15:  END FOR
END DCBF
```

Fig. 2. Double Comparison Bloom Filter Algorithm

We give example of inserting values in DCBF from the hashing process. Let say we have tag 1 from reader 1 with reading counts of 30. Tag 1 is hashed 3 times and the first hashed return 0, the second hash return 1 and the third hash return 3. The value of 30 will be inserted to these hashed counters in CBF1 and CBF2 as shown in Table 1.

Table 1. The condition of CBF after tag 1 is hashed 3 times

CBF1	30	30	0	30	0	0
CBF2	1	1	0	1	0	0
Counter positions	[0]	[1]	[2]	[3]	[4]	[5]

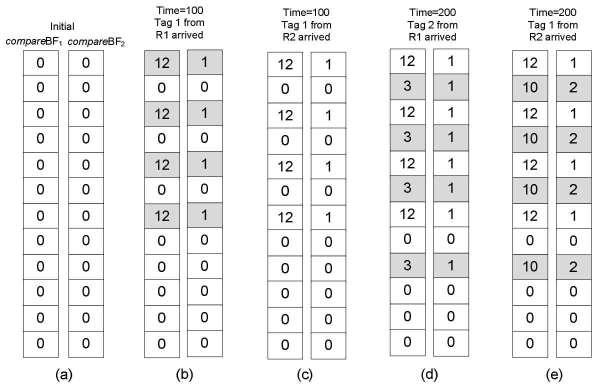
To have the complete views, here is another example how DCBF works. Initially, all the counters in both filters are set to 0. When new reading record arrives, CBF will only insert the count of the reading if the count is higher than current hashed counters values. For each time a reading is inserted into CBF, it means at that time the reader has the highest reading on that tag.

For example refer to table 2 and Fig. 3. Table 1 list the readings on tag 1 and tag 2 by two readers which are R1 and R2. Each reader will send the number of readings on their tag for every 100secs to the global filter. Initially all counters both CBF1 and CBF2 are set to 0 as shown in Fig. 3(a). At time 100, R1 sends reading on tag 1 which is 12. The tag 1 is hashed k times and will be inserted in filters since there were no other readings previously.

Table 2. Reading on tag A1 by different reader

Time	Reader ID	Tag ID	Count of readings
100	R1	1	12
100	R2	1	3
200	R1	2	3
200	R2	2	10

All the hashed counters (shaded boxes) in CBF1 will be given value of 12 to represent the number of count while hashed counters in CBF2 is given value 1 to represent the reader ID as shown in Fig. 3(b). Then reading on tag 1 by R2 arrived with number of readings 3. Tag 1 is hashed and returns the same counters like the previous reading. However this reading will be ignored by the filter because the number of readings by R2 on tag 1 is lower than the previous (Fig. 3(c)).



Initial
compareBF1 compareBF2
Time=100 Tag 1 from R1 arrived
Time=100 Tag 1 from R2 arrived
Time=200 Tag 2 from R1 arrived
Time=200 Tag 1 from R2 arrived
(a)
(b)
(c)
(d)
(e)

Fig. 3. The state of CBF1 and CBF2 based readings in Table 1

We need to keep track on how many tagged objects that each reader has. Based on table 2, at time 100, R1 has one object while R2 has none. At time 200, R1 arrived with reading on tag 2 with number of readings 3. Tag 2 will be inserted in the filters since all the hashed counters returns 0 which means this is the new reading for tag 2

(Fig. 3(d)). Now R1 has 2 objects (including tag 1). When reading from R2 arrives, it also reads tag 2 but with the higher reading than R1 did. The algorithm will insert the new number of readings on tag 2 by R2 in the filter and remove the previous reading of tag 2 by R1 (Fig. 3(e)). By storing the reader ID the filter can answer the query to which reader that a tag is belong to.

4 Performance Analysis

In this section, we present the performance analysis of the proposed algorithm. We will first discuss the experimental setup. We then discuss the results of the experiments.

4.1 Experimental Setup

As in [16] and [17], we generated the data streams using Binomial distribution. We use the sliding windows-based approach the Baseline algorithm [7] and the original Bloom filter [8] to compare the performance of the proposed algorithm.

4.2 Analysis of False Positive Rate

In this experiment, we want to analyse the false positive rate (FPR) of DCBF. A false positive will occurs in DCBF when a reading is detected incorrectly as a duplicate in the same window. We perform this experiment to find out the ratio of array size *m* to the number of readings *n* along with number of hash functions *k* that will return the lowest FPR. The result from this experiment will be used to set the parameters in the next experiments.

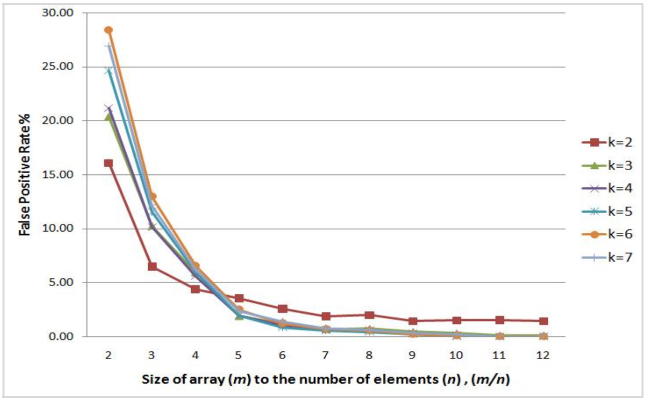


Fig. 4. False positive rate in DCBF

Result for the first experiment is shown in Fig. 4. DCBF is run under different number of hash functions with varied ratio on number of readings *n* to the array *m*. From Fig. 4 we can see that FPR rate reaching almost zero when the number of hash function is more than 4 and the size of *n* is 1/9 to the size of *m*. Based on this result we will run DCBF on next experiments with 7 hash functions and set the size of *m* 9 times bigger than number of elements *n* to get the best results.

4.3 Comparative Analysis of the Algorithms

In this experiment, we investigated the rate of incorrect filtering by DCBF, Baseline and Bloom filter. Incorrect filtering means that the tag with lower reading count failed to be filtered and have been inserted into the filter. For this experiment we generate 200 tags readings for 2 readers. The number of overlapped readings will be varied from 5% to 50% for each data sets. Tag that located in the major detection region will have 80% read rate while minor detection region will have 20% read rate. The reading cycle will be repeated for 10 times. The overlapped readings were scattered randomly in the data stream.

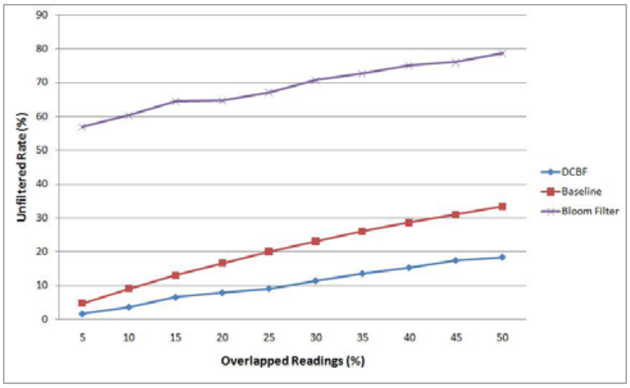


Fig. 5. Percentage of unfiltered duplicate readings

Results on Fig. 5 shows that DCBF performs better than Baseline and Bloom filter to filter the duplicate readings. DCBF has the lowest unfiltered reading rates among the other algorithms. The highest is the Bloom filter [8]. This is because Bloom filter could not stored the data on the number of readings and reader ID which is needed to perform this task correctly. For sliding windows approach they have problem to filter correctly when duplicate readings are scattered in the data stream. The RFID readings can be scattered and non-continuos due to the outside interference and signal weakness. When the readings are scattered there are readings that cannot be compared with each other while it is need to be done to filter the duplicate. This left some duplicate readings in the windows.

4.4 Execution Time

In this experiment, we measure the execution time of DCBF, Baseline and Bloom filter to perform the filtering tasks. The data set generated have different number of reading arrivals per cycle is set to 20, 40, 80, 160, 320, 640 and 1280.

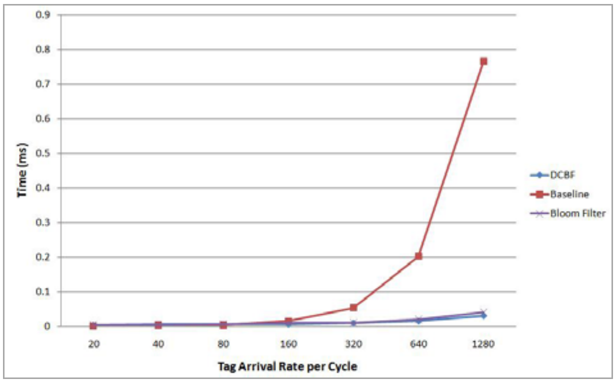


Fig. 6. Time execution comparison to filter the duplicate readings

From Fig. 6, DCBF perform better in terms of time execution than Baseline. Baseline which is based on sliding windows approach takes more time to execute especially when the reading getting high arrival rate per cycle. This is because it has to go through along the windows that become bigger with the increase of tag arrival rate for each new coming reading. This is different from DCBF where the arrival rate does not have exponential effect on its time processing. Unlike Baseline, DCBF does not have to go through along the windows to check for the duplication. Its only need to hash the tag ID to check whether it is a duplicate reading or not. For Bloom filter the performance is same like DCBF. However as the previous result shows it has very high unfiltered duplicates which make it is not suitable to perform this task.

5 Conclusions

In this paper, we studied the problem of RFID data duplicate problem and proposed a new approach that is based on Bloom filter ideas. We compared the performance of the proposed approach with several existing approaches. The results show that proposed approach has low false rate and best execution time as compared to the existing approaches.

References

1. Pupunwiwat, P., Bela, S.: Location Filtering and Duplication Elimination for RFID Data Streams. In: International Journal of Principles and Applications of Information Science and Technology, vol. 1(1) (December 2007)
2. Leong, K.S., Ng, M.L., Grasso, A.R., Cole, P.H.: Synchronization of RFID readers for dense RFID reader environments. In: Proceedings of the 2006 International Symposium on Applications and the Internet Workshops (SAINT 2006), pp. 48–51 (2006)
3. Martinez-Sala, A.S., Egea-Lopez, E., Garcia-Sanchez, F., Garcia-Haro, J.: Tracking of Returnable Packaging and Transport Units with active RFID in the grocery supply chain. Computers in Industry 60(3), 161–171 (2009)

4. Ko, C.-H.: RFID 3D location sensing algorithms. Automation in Construction. Building Information Modelling and Collaborative Working Environments 19(5), 588–595 (2010)
5. Derakhshan, R., Orlowska, M., Li, X.: RFID Data Management: Challenges and Opportunities. In: IEEE International Conference on RFID, pp. 175–182 (2007)
6. Bloom, B.: Space/time tradeoffs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
7. Bai, Y., Wang, F., Liu, P.: Efficiently Filtering RFID Data Streams. In: CleanDB Workshop, pp. 50–57 (2006)
8. Wang, X., Zhang, Q., Jia, Y.: Efficiently Filtering Duplicates over Distributed Data Streams. In: International Conference on Computer Science and Software Engineering, Wuhan, Hubei, pp. 631–634 (2008)
9. Jeffery, S.R., Alonso, G., Franklin, M.J., Hong, W., Widom, J.: A Pipelined Framework for Online Cleaning of Sensor Data Streams. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), p. 140 (2006)
10. Rao, J., Doraiswamy, S., Thakkar, H., Colby, L.S.: A Deferred Cleansing Method For RFID Data Analytics. In: Int. Conf. on Very Large DataBases (VLDB 2006), pp. 175–186 (2006)
11. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and analysing massive RFID data sets. In: Proc. of the International Conference on Data Engineering (ICDE 2006), pp. 1–10 (2006)
12. Song, J., Hass, C.T., Caldas, C.H.: A Proximity-Based Method for Locating RFID Tagged Objects. Adv. Eng. Inform 21, 367–376 (2007)
13. Shen, H., Zhang, Y.: Improved approximate detection of duplicates for data streams over sliding windows. Journal Of Computer Science And Technology 23(6), 973–987 (2008)
14. Mahdin, H., Abawajy, J.: An Approach to Filtering RFID Data Streams. In: 10th International Symposium on Pervasive Systems, Algorithms, and Networks, pp. 742–746 (2009)
15. Fan, L., Cao, P., Almeida, J.: Broder A Z. Summary cache: A Scalable Wide-Area Web Cache Sharing Protocol. IEEE/ACM Trans. Networking 8(3), 281–293 (2000)
16. Jeffery, S.R., Garofalakis, M., Franklin, M.J.: Adaptive cleaning for RFID data streams. In: Proceedings of the 32nd International Conference on Very Large Databases, pp. 163–174 (2006)
17. Chen, W.-T.: An accurate tag estimate method for improving the performance of an RFID anticollision algorithm based on dynamic frame length ALOHA. In: Chen, W.-T. (ed.) IEICE Trans. on Automatic Science and Engineering, pp. 9–15 (2009)