Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 7107914, 7 pages
http://dx.doi.org/10.1155/2016/7107914

Research Article

Filtering Redundant Data from RFID Data Streams

Hazalila Kamaludin,[1] Hairulnizam Mahdin,[1] and Jemal H. Abawajy[2]

[1]*Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400 Batu Pahat, Johor, Malaysia*
[2]*IEEE, School of Information Technology, Deakin University, Waurn Ponds, VIC 3216, Australia*

Correspondence should be addressed to Jemal H. Abawajy; jemal@deakin.edu.au

Received 23 July 2015; Accepted 23 November 2015

Academic Editor: Pietro Siciliano

Radio Frequency Identification (RFID) enabled systems are evolving in many applications that need to know the physical location of objects such as supply chain management. Naturally, RFID systems create large volumes of duplicate data. As the duplicate data wastes communication, processing, and storage resources as well as delaying decision-making, filtering duplicate data from RFID data stream is an important and challenging problem. Existing Bloom Filter-based approaches for filtering duplicate RFID data streams are complex and slow as they use multiple hash functions. In this paper, we propose an approach for filtering duplicate data from RFID data streams. The proposed approach is based on modified Bloom Filter and uses only a single hash function. We performed extensive empirical study of the proposed approach and compared it against the Bloom Filter, d-Left Time Bloom Filter, and the Count Bloom Filter approaches. The results show that the proposed approach outperforms the baseline approaches in terms of false positive rate, execution time, and true positive rate.

1. Introduction

RFID's capability of object identification without line of sight has proved to be useful in many applications that need to know the physical location of objects. For example, RFID has been used in pharmaceutical companies for purposes such as tracking of counterfeit pharmaceutical products as well as managing inventory. Similarly, RFID has been used in hospitals for tracking expensive medical devices, patient identification and location, and blood inventories [1–3]. RFID is also useful in indoor positioning [4] to acquire precise location of the monitored object like monitoring people working underground. These applications have shown that RFID has greatly improved the process of automatically identifying, locating, tracking, and monitoring the objects with less human intervention. Although RFID provides many benefits such as reducing labour cost, RFID generates a lot of duplicate data due to RFID readers' overlapped regions in dense area and also readers multiple read cycles [5]. In order to increase accuracy of the read data, RFID readers do reading of the tags periodically but a lot of duplicate data was produced [5]. A reading is considered duplicate when it is repeated and does not deliver new information to the system.

In addition to wasting communication, processing, and storage resources, duplicate data may lead to wrong interpretation and decisions [6]. For example, it may lead to miscalculation of the stocks on the store shelf [6]. This necessitates a mechanism for duplicate data removal from RFID data streams before it is transmitted for processing by the client applications. Many Bloom Filter- (BF-) based approaches for filtering duplicate RFID data streams are proposed in the literature [7–10]. However, these approaches are very complex and slow as they use multiple hash functions. Any significant reduction in the time required to perform a Bloom Filter operation translates to a significant speedup for many practical applications [11].

In this paper, we propose BF-based approach for detection and filtering out of duplicate readings from RFID data stream. Unlike the existing approaches that use multiple hash functions, the proposed approach uses a single hash function. Extensive performance analysis of the proposed approach is carried out and compared it with the approaches proposed in

[9, 10, 12]. The results of the experiments show that the proposed approach performs better than the baseline approaches in [9, 10, 12] in terms of false positive rate, execution time, and true positive rate. The experiments also demonstrate that performance speedup is directly proportional to the number of hash functions employed by the BF-based approaches.

The rest of the paper is organized as follows: Section 2 presents the background information and the related work while the proposed algorithm is described in detail in Section 3. Performance analysis of the proposed algorithm is presented in Section 4. The results presented in this section demonstrate that the proposed approach significantly outperforms other existing approaches. Conclusion and future work directions are presented in Section 5.

2. Background

RFID based systems basically consist of transponders (tags), interrogators (readers), and middleware deployed at a host computer. A tag is attached to an object that needs to be monitored or identified. Meanwhile, a reader creates radio frequency field to detect radio waves reflected back from tags and converts the radio waves into digital information for processing. The middleware collects and processes RFID data stream from readers before it is stored in the database for use by the enterprise applications.

A tag is categorised as active or semiactive or passive. Passive tags have no power while active and semiactive tags have their own battery to run the chip circuit but semiactive ones still need power from reader to communicate. The RFID reader will send request to tags to acquire data and forward it to the backend applications or database servers through the middleware. The applications then respond to the received data and generate corresponding actions such as triggering of theft alert, ordering of new stocks, or indication of replacing fragile component before failure.

Multiple networked RFID readers are commonly installed to maintain availability and readability of the tags. However, this kind of installation produced data redundancy or duplicate reading. The same problem also arises when reading is done in multiple cycles [6]. Generally, there are three types of duplicate reading problems: (i) data level, (ii) multiple read cycle, and (iii) redundant reader. Duplicate readings at data level occur when multiple tags with the same Electronic Product Code (EPC) are attached to the same object to reduce missing rate and increase reliability [13]. For static tagged objects or objects that remain in reader vicinity for a long time (in multiple reading cycles), they are read by the reader multiple times [14]. In order to cover a larger area or distance, tags that exist in overlapped areas are read by multiple readers [15].

Figure 1 depicts the problem of duplicate readings in hypermarket scenario where the items are static on each shelf and send data to the RFID system constantly. In the hypermarket, the shelves are equipped with the RFID readers with the aim of maintaining availability and readability. As the reading vicinity of reader 1 overlaps with reader 2 and reader 3, tag 1 and tag 2 can be read by all the three

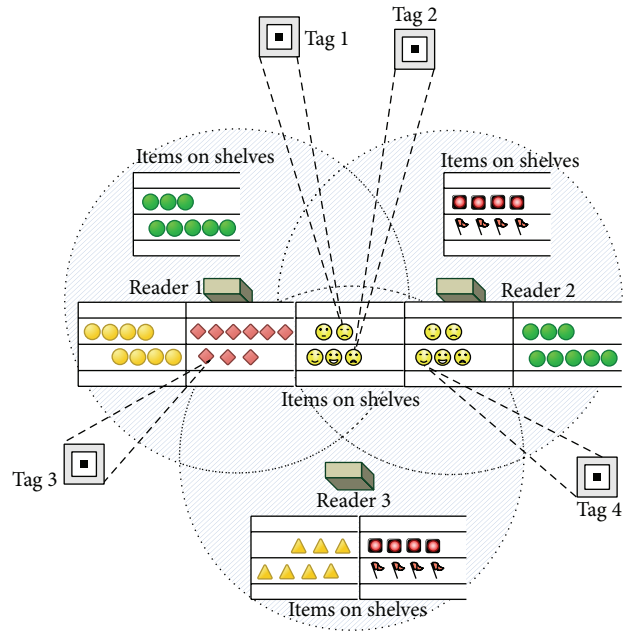

Figure 1: RFID enabled system at hypermarket: monitoring static items on shelves.

readers. These duplicate readings will flood the network bandwidth and may have adverse effect in applications such as stock management [9]. To address this problem, a variety of Bloom Filter-based [9, 10, 12, 16] and non-BF-based [14, 17] approaches have been proposed in the literature. The latter approaches include those that used sliding window [14, 17] and landmark window in filtering duplicate readings [9]. The sliding window approach is more suitable to the applications that always have new readings that have not been recorded before. In this paper, we will focus on the approaches that use Bloom Filter.

A Bloom Filter (BF) [12] data structure contains bit array of size m with k hash functions. Initially all bits in the array will be set to 0 and replaced with 1 when being hashed by an element. An element can be tested whether it is already in the array by hashing it using the same set of hash functions. An element is said to be a member of the array if all k positions are set to 1. The Count Bloom Filter (CBF) [9] approach filters duplicate RFID data stream at reader level. The Time Bloom Filters (TBF) and Time Interval Bloom Filters (TIBF) are approaches for removing RFID duplicate filtering [7]. TBF is a simple extension of the conventional Bloom Filters while TIBF need more spaces to reduce errors. However, these approaches suffer from error rates while the number of tuples increases.

The Chain-Linked Counting Bloom Filter (CCBF) [8] attempts to reduce filtering processing rate according to the hashing times. This approach combines Counting Bloom Filters with dynamic chain lists to reduce errors. But this approach is complex and consumes a lot of time. It also relates dynamic setting with the object arrival rate and not the departure rate which indicates readings that can be removed from the filter. d-Left Time Bloom Filter (DLTBF) [10] is

Table 1: Summary of existing duplicate RFID data filtering approaches.

Approaches	Weaknesses
DLTBF [10]	Needs to compute the least loaded bucket before insertion. Also, it uses a number of hash functions.
CBF [9]	Filters duplicate readings only at a reader level. It also uses a number of hash functions.
BF [12]	The filter is easily becoming full since it does not allow deletion and therefore generates a higher rate of false positive.
TBF, TIBF [7]	Not memory efficient because multiple time counters are needed to store the reading time.
Sliding windows [19]	Inefficient since it has to scan along the sliding window every time new reading is coming in which it scans almost the same data. Besides that, sliding windows can overlap but disjoint the first items.
Landmark windows [19]	Size of the window can be very big to provide more accommodation and accurate results. Besides that landmark windows can overlap in any way.
CCBF [8]	Delay process because when it gets to identify probability of duplicate, it has to search the double-linked list to check whether the intersection of all time intervals corresponding to each hash function $h1(x); h2(x); \ldots; hk(x)$ is empty so that the tag did not arrive within r time. Just related dynamic setting with the object's arrival rate and not the departure rate which indicates readings that can be removed from the filter.

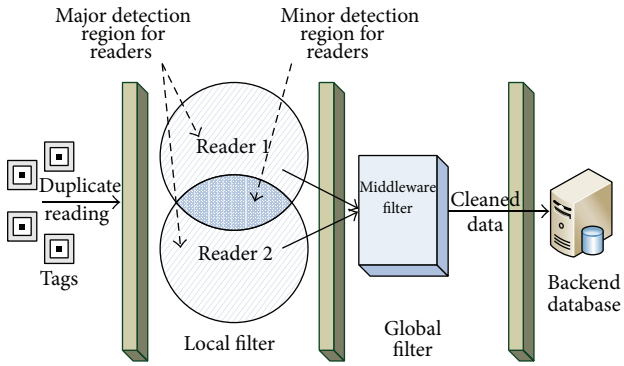

Figure 2: Multilevel duplicate RFID data stream filtering approach.

```
INPUT: TID
BEGIN
(1) IF (Time == True) THEN //initialize the arrays
(2)      array1 = array2 = {0}
(3) ENDIF
(4) FOREACH TID DO
(5)    Pos ← Hash(TID)            //get hash value for TID
(6)    IF ((array1[pos] == TID) ‖ (array2[pos] == TID))
       THEN
(7)       Duplicate read
(8)    ELSEIF (array1[Pos] == ∅) THEN
(9)       array1[Pos] ← TID    //store in array1
(10)   ELSEIF (array2[pos] == ∅) THEN
(11)      array2[pos] ← TID    //store in array2
(12)   ELSE
(13)      False Positive
(14)   ENDIF
(15) ENDFOR
END DFH
```

Algorithm 1: DFH.

an extension of d-Left Counting Bloom Filter (DLCBF) [16]. The DLTBF approach needs to compute the least loaded bucket before insertion. However, it is able to store the detected time of an element into one counter. The common denominator of the existing BF-based RFID duplicate data filtering approaches is that they use multiple hash functions. Our approach uses a single hash function and an integer array is used to keep the RFID data stream as in [7]. Table 1 provides a summary of the duplicate RFID data filtering approaches.

3. Removing Duplicate RFID Data Stream

In this section, the proposed multilevel duplicate RFID data stream filtering approach is presented. The overall filtering approach is illustrated in Figure 2. Generally, RFID reading vicinities are divided into major and minor areas [9, 18]. All tag readings will be first filtered at the reader itself (local filter) and the readings among the readers will then be filtered at the middleware before the data is sent to the database and subsequently to the application for processing.

We assume that the readers probe the tags every 5 seconds. Algorithm 1 shows the pseudocode of the proposed RFID duplicate data filtering approach which we refer to as a Duplicate Filter Hash (DFH). The algorithm maintains two arrays. As in [19], the landmark window is used in the proposed filtering approach because movement of the tagged object cannot easily be predicted and it may sometimes stay longer in the same area. Besides that, landmark window also naturally suits the Bloom Filter and it will remove all data when specific point is met.

DFH will first check the time to remove all the readings at lines 1–3. If the time is met, all the counters will be reset to zero. This is related to the storage management of large volume RFID data. Each incoming reading will be hashed with a single hash function (line 5) and the output is considered as the counter position into the two arrays. The algorithm first checks to see if the reading is duplicated and if so the reading is discarded (lines 6 and 7). Otherwise, if array1 position at the counter is empty, then the tag ID is stored there, or else, if the array2 position at that counter is

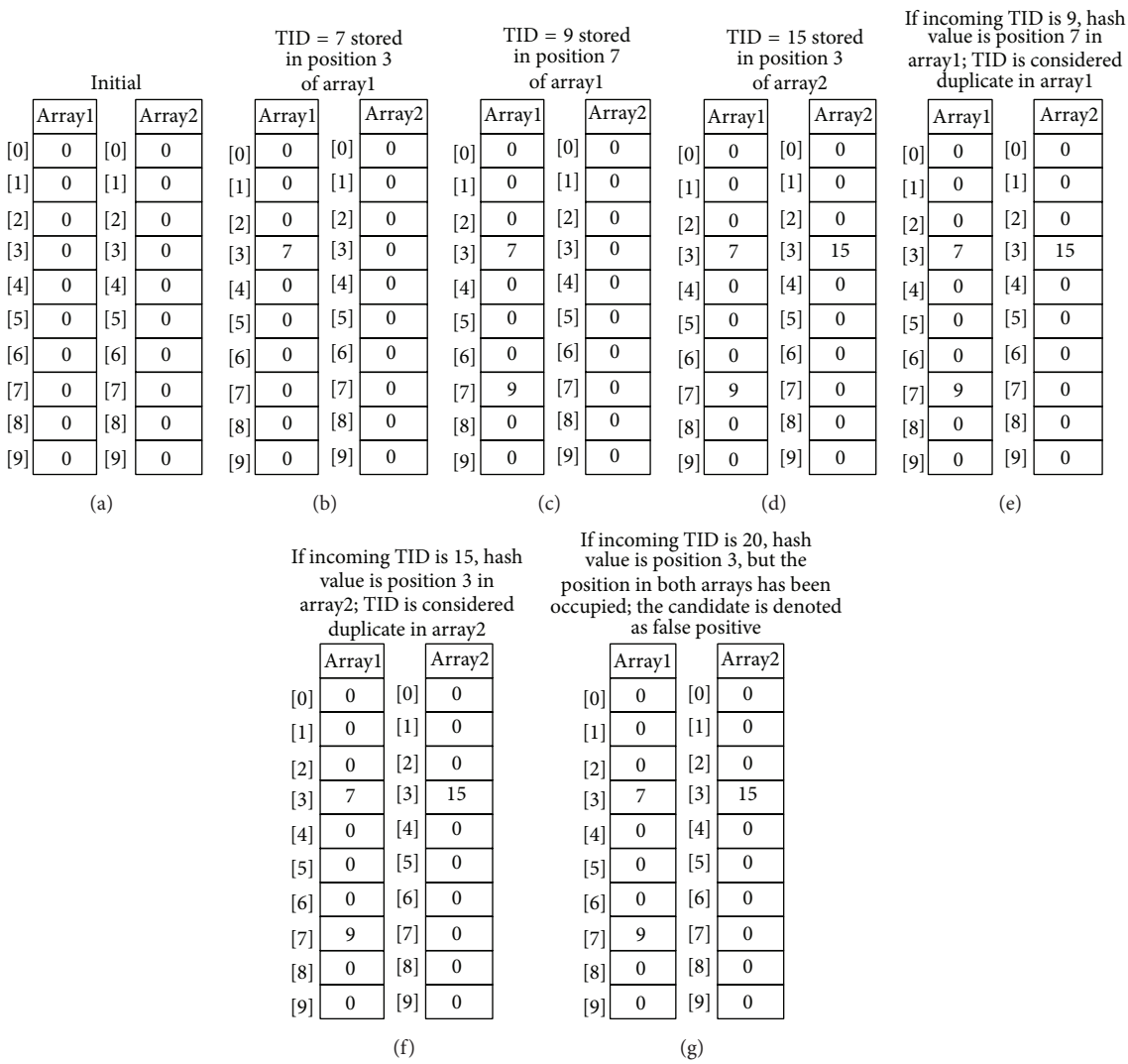

Figure 3: The state of DFH before and after insertion of x.

Table 2: Reading on tags by multiple readers.

Time	Reader ID	Tag ID
5	R1	7
5	R2	9
10	R2	15
10	R1	9
15	R1	15
15	R2	20

Table 3: Reading on tags by single reader.

Time	Tag ID
5	7
5	9
10	15
10	9
15	15
15	20

empty, then the tag ID is stored there. If not, the reading is reported as a false positive.

We now illustrate DFH using the read data shown in Tables 2 and 3. Figure 3 shows the sequences of read data mappings. Figure 3(a) shows the state of DFH before the arrival of tag reading. The first read is tag TID = 7 (see Table 3). Let us say that the hash value of TID = 7 is 3. Therefore, we store 7 at position 3 in array1 (Figure 3(b)). For the new tag reading 9, its hash value is 7. This reading is stored in array1 since it is still empty (Figure 3(c)). For tag reading 15 and hash value 3, we cannot store it in array1 as that position is already occupied. However, the counter value of array2 at this position is not occupied; thus 15 is placed there (Figure 3(d)). The next reading is 9 and the hash value is 7. This reading is considered duplicate in array1 since previously tag ID 9 has been stored. Similarly, the next for tag ID 15 with hash value 3

is considered duplicate in array2. Finally reading 20 with hash value 3 will be considered as false positive since both arrays at position 3 are not empty.

4. Performance Analysis

In this section, simulation is used to analyse the performance of the proposed duplicate reading detection and removal approach. In the experiment, we used the same RFID data streams as in [9, 10, 14, 17]. The data streams have been generated using Poisson distribution to illustrate the tag arrival in such granule time. The data is then randomized to have scattered datasets. The performance of the proposed algorithm is compared against the d-Left Time Bloom Filter (DLTBF) [10], the Count Bloom Filter (CBF) [9], and the conventional Bloom Filter (BF) [12] in terms of false positive rate (FPR), execution time, and true positive rate (TPR). Note that a slight modification is made to these algorithms to fit in with the problems being solved.

In our approach, FPR is referring to the rate of unsuccessful inserted elements in the filter for each number of readings. Therefore, the lower FPR represents the fact that many elements are successfully inserted in the current window of the filter. Furthermore, TPR in our approach is denoted by rate of successful inserted element in the filter. For that reason, higher TPR shows only small numbers of elements that fail to be filtered in the current window. Following are equations for our FPR and TPR, where m denotes number of readings:

$$\text{FPR} = \left[\frac{\text{FP}}{m}\right]\%, \qquad \text{TPR} = \left[\frac{\text{TP}}{m}\right]\%. \tag{1}$$

4.1. Results and Discussions

4.1.1. False Positive Rate. In the experiment, false positive rate of DFH is measured with a few sets of readings and counter sizes. To find out the best ratio of counter size i to the number of readings m that will return the lowest false positive, we performed extensive experiment.

Figure 4 shows the FPR for DFH using different number of counter sizes $i = 5000$, $i = 10000$, and $i = 15000$. The number of readings varied from 500 to 5000 with increment of 500 for each sample. For all i values, FPR is at the lowest when the number of readings is 500. The result shows that FPR reach almost zero percentage when size of the sample is less than 2000. Based on this result, to get lowest FPR for DFH, the counter size i must be 6 times bigger than the number of readings. This measurement is used as the baseline in the next experiments to get the best results.

4.1.2. Comparative Analysis of False Positive Rate. In this section, we compare the false positive rates of the four algorithms. Figure 5 shows the FPR for each of the filtering approaches. On average, both BF and CBF have higher FPR as compared to DFH and DLTBF. This is because both approaches apply more hash functions which makes the filter "full" earlier since many counters were occupied for the element.

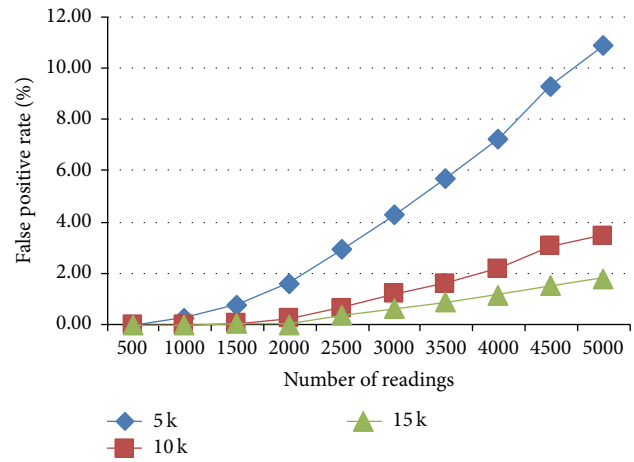

Figure 4: False positive rate of DFH with counter sizes $i = 5000$, $i = 10000$, and $i = 15000$.

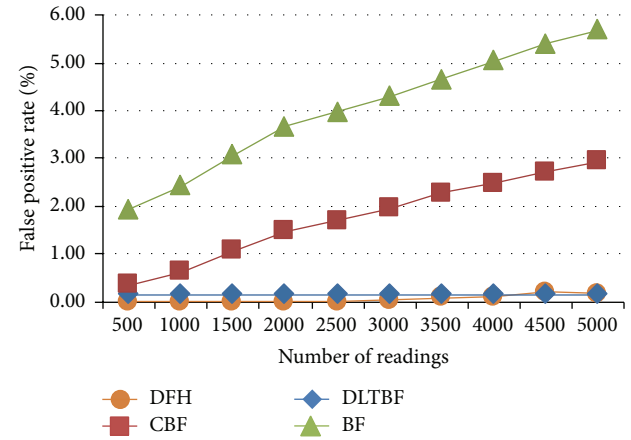

Figure 5: Comparison of FPR between Bloom Filter approaches.

DFH approach is having better FPR as compared to the other approaches since it does not take too much space for a single element inserted in the filter. Therefore, more elements can reside in the filter even though it has less randomness. FPR for DLTBF remains constant with the increasing number of readings. This is because DLTBF considers the nonduplicate element to be as duplicated by constant probability within time r.

4.1.3. Comparative Analysis of Execution Time. Figure 6 shows the execution time of the approaches to filter duplicate readings as a function of increasing number of readings from 10^3 to 50^3 with increment of 10000 for each sample. The window size in this experiment is set to 50000, the highest number of readings.

From Figure 6, we observe that on average BF and CBF took longer to filter the readings as compared to DLTBF and

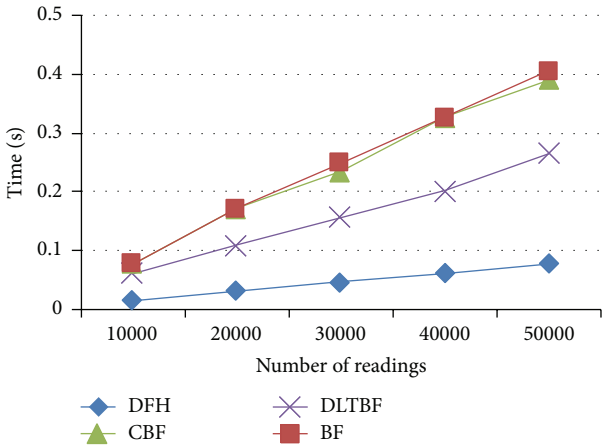

Figure 6: Comparison of execution times for filtering duplicate readings.

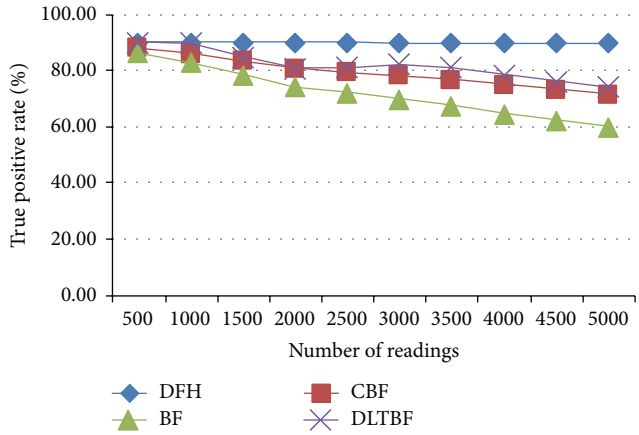

Figure 7: True positive rate comparison for filtering duplicate readings.

DFH. Both BF and CBF apply more hash functions and therefore they took longer to perform filtering. This contradicts with observations in [11] which asserted that reducing the number of hash functions will produce equivalent reduction in the time taken to perform filtering. Since DLTBF applies a reduced amount of hashing compared to BF and CBF, it took moderate execution time for processing. However, DFH substantially outperforms the three filtering algorithms. This is because it applies a single hash function. This result verifies that, by reducing randomness (the number of hash functions), DFH required less time for filtering operation while maintaining its reliability with great results.

4.1.4. Comparative Analysis of True Positive Rate. In this section, DFH is compared with DLTBF, CBF, and BF in terms of true positive rate. The rule of TPR reveals that the higher the true positive value, the more accurate the solution. In this measurement, 10% duplicate is applied to each data sample and TPR is referred to as the number of elements successfully placed in the filter.

As depicted in Figure 7, BF and CBF acquire lower TPR because both of these filtering algorithms get "full" more quickly than DFH and DLTBF. DFH however shows better TPR than DLTBF. This better performance is because DFH occupies less space for inserting a single element and therefore it suffers less when the number of readings increases.

5. Conclusions and Future Directions

In this paper, the problem of RFID data redundancy has been studied and a new approach based on Bloom Filter is proposed. Performance of the proposed approach is compared with several existing approaches. The results show that the proposed approach performs much better than the other approaches in terms of false positive rate and true positive rate. Other than that, this study also has demonstrated that, by reducing the number of hash functions, the Bloom Filter-based approach can perform faster than the others in the experiments efficiency.

For future work, we plan to investigate how the size of the landmark window can be set dynamically based on object departure rate from the reader vicinity. This will help in reducing the probability of the false positive rate. Unlike during the arrival, object does not inform the reader directly that it has left the reading vicinity. Object departure rate is more appropriate in order to determine the window size since it indicates which candidates can be removed from the filter rather than clearing the entire filter when specific times are encountered. Besides that, reading time can be included in the proposed approach as an additional parameter to determine whether to store or not such duplicate reading. If the new duplicate reading is within a time frame specified, it can be dropped and if it is out of the time frame, it can be considered as another new reading and positioned in the filter.

Conflict of Interests

The authors declare no conflict of interests.

Acknowledgments

This work is sponsored by Universiti Tun Hussein Onn Malaysia under Exploratory Research Grant E054 and the Ministry of Education through SLAB scholarship. This research is also supported by GATES IT Solution Sdn. Bhd. under its publication scheme.

References

[1] Z. Zhao, "A secure RFID authentication protocol for healthcare environments using elliptic curve cryptosystem," *Journal of Medical Systems*, vol. 38, no. 5, pp. 1–7, 2014.

[2] C. Jin, C. Xu, X. Zhang, and J. Zhao, "A secure RFID mutual authentication protocol for healthcare environments using elliptic curve cryptography," *Journal of Medical Systems*, vol. 39, no. 3, pp. 1–8, 2015.

[3] Z. Zhang and Q. Qi, "An efficient RFID authentication protocol to enhance patient medication safety using elliptic curve cryptography," *Journal of Medical Systems*, vol. 38, no. 5, pp. 1–7, 2014.

[4] S. L. Ting, S. K. Kwok, A. H. C. Tsang, and G. T. S. Ho, "The study on using passive RFID tags for indoor positioning," *International Journal of Engineering Business Management*, vol. 3, no. 1, pp. 9–15, 2011.

[5] A. K. Bashir, M.-S. Park, S.-I. Lee, J. Park, W. Lee, and S. C. Shah, "In-network RFID data filtering scheme in RFID-WSN for RFID applications," in *Intelligent Robotics and Applications*, pp. 454–465, Springer, 2013.

[6] R. Derakhshan, M. E. Orlowska, and X. Li, "RFID data management: challenges and opportunities," in *Proceedings of the IEEE International Conference on RFID*, pp. 175–182, Grapevine, Tex, USA, March 2007.

[7] C.-H. Lee and C.-W. Chung, "An approximate duplicate elimination in RFID data streams," *Data and Knowledge Engineering*, vol. 70, no. 12, pp. 1070–1087, 2011.

[8] H. A. O. Yongsheng and G. E. Zhijun, "Redundancy removal approach for integrated rfid readers with counting bloom filter," *Journal of Computer Information Systems*, vol. 9, no. 5, pp. 1917–1924, 2013.

[9] H. Mahdin and J. Abawajy, "An approach for removing redundant data from RFID data streams," *Sensors*, vol. 11, no. 10, pp. 9863–9877, 2011.

[10] X. Wang, Y. Ji, and B. Zhao, "An approximate duplicate-elimination in RFID data streams based on d-left time bloom filter," in *Web Technologies and Applications*, vol. 8709 of *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 413–424, Springer, Basel, Switzerland, 2014.

[11] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: building a better bloom filter," in *Algorithms—ESA 2006*, vol. 4168 of *Lecture Notes in Computer Science*, pp. 456–467, Springer, New York, NY, USA, 2006.

[12] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[13] Y. Bai, F. Wang, and P. Liu, "Efficiently filtering RFID data streams," in *Proceedings of the 1st International VLDB Workshop on Clean Databases (CleanDB '06)*, Seoul, Republic of Korea, September 2006.

[14] S. R. Jeffery, M. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)*, pp. 163–174, September 2006.

[15] A. K. Bashir, S.-J. Lim, C. S. Hussain, and M.-S. Park, "Energy efficient in-network RFID data filtering scheme in wireless sensor networks," *Sensors*, vol. 11, no. 7, pp. 7004–7021, 2011.

[16] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "An improved construction for counting Bloom filters," in *Algorithms—ESA 2006*, vol. 4168 of *Lecture Notes in Computer Science*, pp. 684–695, Springer, Berlin, Germany, 2006.

[17] L. V. Massawe, J. D. M. Kinyua, and H. Vermaak, "Reducing false negative reads in RFID data streams using an adaptive sliding-window approach," *Sensors*, vol. 12, no. 4, pp. 4187–4212, 2012.

[18] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 1015–1020, IEEE, May 2004.

[19] I. Botan, D. Kossmann, P. M. Fischer, T. Kraska, D. Florescu, and R. Tamosevicius, "Extending XQuery with window functions," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 75–86, Vienna, Austria, September 2007.