

# A Review on Bloom Filter Based Approaches for RFID Data Cleaning

Hairulnizam Mahdin

FSKTM, Universiti Tun Hussein Onn Malaysia,  
Batu Pahat, Johor, Malaysia  
hairuln@uthm.edu.my

**Abstract.** This paper provides comprehensive review about data cleaning for RFID data streams. It serves the purpose to understand the current undertakings to ensure data quality in RFID. It focused on three major RFID data issues which are noise readings, duplicate readings and missed readings. It includes in-depth analysis on existing approaches specifying on Bloom filter based approaches. This literature can be used by researcher to understand the background of RFID data filtering, the challenges and expectation in the future.

**Keywords:** RFID, data filtering, noise reading, missed reading, duplicate reading, Bloom filter

## 1 Introduction

RFID identification works by reader reading ID on tag and send it to the middleware for processing. Before readings can be transformed into information, it needs to be filtered. The RFID data stream contains unreliable data such as noise reading and duplicates. Noise reads will produce incorrect information such as incorrect stock quantity. Duplicate readings need to be removed because it over-represents the data and does not contain new information for the system. It needs to be removed to avoid unnecessary processing being performed.

The RFID data stream is different from the common relational and data warehousing because of (i) large size of data (ii) simple tuple structure (iii) inaccuracy and (iv) temporal and spatial information that make it require new data management approach. To illustrate the RFID data size, consider small store that have 10,000 items. 10,000 readings generated. If the readings are repeated for every 10 minutes, in eight hours there will be already 480,000 tuples. This is why it needs really efficient data structure based approach to cater the large size of data. The filtering process must be carried out to ensure only correct readings are used to produce information. This paper presents and analyzes the existing filtering approaches which are based on Bloom filter.

## 1.1 RFID System Architecture

A typical RFID system consists of a transponder (i.e., tag), which is attached to the objects to be identified, an interrogator (i.e., reader) that creates an RF field for detecting radio waves, and a backend database system for maintaining expanded information on the objects and other associated objects. Figure 1 shows RFID-enabled a generic system of interest. Generally RFID system architecture is made of four layers: tags, readers, middleware and database and enterprise applications.

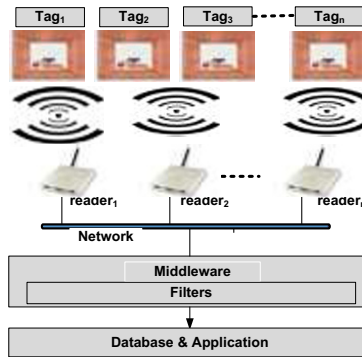


Fig. 1. An RFID-enabled system architecture

At the first layer, objects to be tracked and monitored are attached with an RFID tag. A tag contains memory to store the identifying information of the object to which it is attached and an antenna that communicates the information via radio waves. Tags can be classified based on their power sources: passive, active and semi-active tags. Active and semi-active tags have their own battery on-boards while passive tags not. Passive tag is the most commonly used tags in the market and has an indefinite operational life compare to other tags. Relative to both active and semi-active tags, passive tags are very cheap and they are widely used in very large quantities in many applications such as supply chain management.

Generally, passive tags are most used tags in many applications because of its cheap price than the others [1]. They were use massively mostly in the supply chain application. However, due to the low-powered hardware and the massive number of the tags, it raises many issues in data management and security. Most of the researches discussed in this thesis are based on the passive tags.

At the second layer, the RFID system is assumed to contain multiple networked RFID readers deployed to collaboratively collect data from tagged objects in their vicinity. The reading distance ranges from a few centimeters to more than 300 feet depending on factors such as interference from other RF devices [2]. The RFID readers query tags to obtain data and forward the resulting readings to the middleware. The middleware processes these data and then send it 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.

## 1.2 RFID Reading Classes

The type of readings generate by reader could be generally classified into four classes: true positive, false positive, false negative and duplicate readings. Only true positive reading is acceptable in the RFID system. The other types of reading are the three major anomalies that need to be removed or smoothed from the data stream [3]. Each anomaly has different impact to the RFID system. The unreliable data are being listed as one of the major hindrance in achieving widespread adoption of RFID technology [4].

### 1.2.1 True positive

True positive is correct readings made by the reader. It returns the actual ID on the tag to the reader. The structure of tag ID according to the EPC Tag Data Standard consists of four parts: Header, EPC manager, Object Class and Serial Number. The example of tag ID is *01.0000E78.00019A.000198CFE*. Header identifies the length, type and structure of EPC, EPC Manager identifies the manufacturer, Object class identifies the type of product and Serial Number is the unique identifier for the item.

### 1.2.2 False positive

The second reading class in RFID readings is false positive, also known as noise reading. The reading returns tag ID that does not exist in the system. The corrupted reading can be caused by: (i) low power signal, (ii) signal interference, (iii) signal collision, and (iv) infrastructure obstacle [5]. Low power signals occur when the tag is located at the end of reader's vicinity or the reader trying to read too many tags at the same time [6]. The tag will not be having sufficient power to reply the signal back to the reader successfully. The radio signal also can be weakened by interference from metal and water [6].

The third cause of noise reading is the signal collision which is common in RFID. Signal collision occurs when two or more tags responding to the reader at the same time [7]. It also occurs when the tag is being read by more than one reader [8]. The signal collision can change the content of the signal which creates new ID that did not exist in the system. Another source of noise readings is the infrastructure obstacle such as the orientation of the objects and obstacles from the surrounding environment. When numbers of pallets are coming together, some of the tags can be buried deep in the pallet arrangement, which made them hard to be read. The amount of power coming to them is not enough for the tag to responds to the reader correctly, which results in noise readings.

The effect of false positive or noise readings is it reports an existence of an object that is not exists, causing the system to take wrong decision opposing the reality. For example a noise read at the security check-out in a shopping mall will trigger alarm

indicating that a customer did not pay for an item. The noise read also can indicate inaccurate number of items available in the store which causing delays in ordering new stock. The noise readings need to be removed from the data stream to avoid such confusion in those examples. A pro-active step that can be taken to reduce the noise readings problem is to ensure that the right tag is used for the application. For example it is better to use high frequency (HF) tag to read object that is build up from metal compare to ultra high frequency (UHF) tag because HF tag have good performance with metal.

### 1.2.3 False negative

The third type of reading is the false negative or missed reading. False negative is a reading that supposedly performed by the reader but is being left from entering the data stream. The source of problem can be the same as noise reading. In this case the signal did not reach the reader at all to transmit the data. False negative also can occur due to the filtering process itself. Some of the filtering process put a threshold for a reading to be counted as correct reading within a specific time period. When the object only resides in the vicinity for a short time period, the number of readings made on it is less than the threshold. Therefore it is being removed from the data stream and left undetected. The effect of false negative to the system is opposed to false positive errors. While false positive add the quantity ups from the reality, the false negative reduces the real quantity. There problem with the incorrect quantity is like business loss because some of the items shipped to customer is not being detected. Simple ways to increase the chances of the tag being read is by increasing the number of read cycle or use more than one reader to read the same area [9].

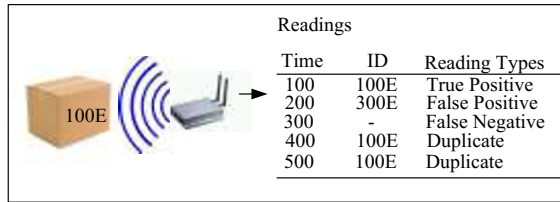
### 1.2.4 Duplicate reading

The next reading class that considered as anomaly is duplicate reading. Duplicate reading is common problem in RFID. It is exists because RFID reader has the ability to read the same tag number of times as long the tag's is still the reader's vicinity. Duplicate reading need to be removed because it over represent the data, unnecessarily occupying the memory space and require unnecessary processing. We are only interested on the data that indicate the occurrence of events. For example on smart shelf application, the readings that are most important are when the item is put on the shelf and when it is being pickup by customer. That can indicate the current number of items that are currently available on the shelf. Another source of duplicate readings is because more than one reader is covering the same vicinity to increase the reliability of the readings. In some cases, the reader's vicinity was overlapped with each other that causing the redundancy.

## 2 Basic of RFID Data Filtering

Fig. 2 depict the basic pattern in RFID data stream that need to be filtered. It contains true positive, noise, missed and duplicates reading. At time 100 the reader read the tag correctly which it generate the ID 100E. However at time 200 it produced noise

reading where it generate tag ID 300E, which is not exists in the reader interrogation area. At time 300 it misses to read the tag but at time 400 and 500 it read the tag correctly which create duplicate readings in the data stream.



Readings		
Time	ID	Reading Types
100	100E	True Positive
200	300E	False Positive
300	-	False Negative
400	100E	Duplicate
500	100E	Duplicate

Fig. 2. Types of reading possibly generated when reader reads tag 100E

Basically, the occurrence of true positive is higher than false positive [9]. The false positive can be filtered by removing readings that have low readings. The problem now is how to set the threshold and how to perform the filtering efficiently.

The second anomaly, the false negative can be solved by adding more reading cycles, so that tag have more chances to be read. By filtering the duplicate reading too, the missed reading can be recovered. For example, it has gone missing at time 300, but at time 400 and 500 it is being read again. That’s mean the tag exist in the reader interrogation area from time 100 to 500. Based on this, the problem of false negative has been solved. The problem now when we increase the reading cycle, there will be higher probability on false positive and duplicate readings occurrence. Our research will be focusing on filtering these anomalies which in the same time recover the missed readings.

### 3 Filtering Approaches

In this section we discuss on RFID data filtering especially on noise and duplicate reading based on Bloom filter.

#### 3.1 Bloom-Filter based Approach

Bloom filter [10] can represent a data in its bit array of size  $m$  that is done through a number of hash functions. Each hash function that runs on the data will return a number which referring to the counter position in Bloom filter. The counter’s value is turn to one from zero when it is being hashed. To test whether a data has been stored in the filter, the data need to be run through the same hash functions. If the value of all counters from the hashing process is positive, that’s mean the data already have been inserted in the filter. If there is more than one counter that is zero, its mean that the data is new and have not been inserted in the filter. Bloom filter achieve efficiency by having constant operation in checking the existence of the data through the hashing process by allowing some false positive. The operation of Bloom filter is shown in Fig. 3. In Fig. 3(a) the word “Apple” is inserted into Bloom filter using 3 hash

functions. In Fig. 3(b) the “Orange” is checked whether it has been inserted to the filter using the same number of hash functions. The first and third hash return counter that is zero which means “Orange” has never been inserted in the filter.

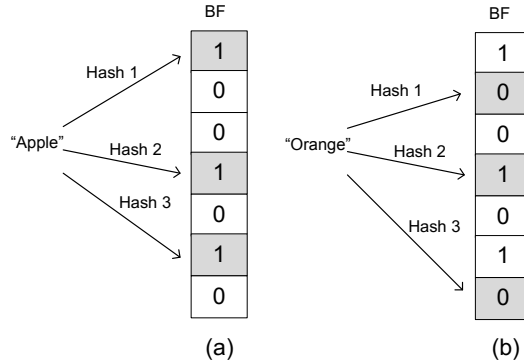


Fig. 3. (a) Insertion of “Apple” in the filter and (b) checking whether “Orange” already has been in the filter

Bloom filter has been used to filter duplicate data in its original and modified form such as in [11-13]. In [11], Bloom filter is used to filter duplicate data from being sent to the coordinator site. The readings that come to remote site will be inserted in the Bloom filter. Then the copy of the Bloom filter is sent to the coordinator sites. The coordinator site then updates its filter with the new reading and sent it to the other remote site. If the same reading being produced again at any of the remote sites, it will be ignored because it has been inserted in the filter. However, this approach is not feasible to be implemented because the coordinator needs to send the updates to each remote site whenever there is an update to its Bloom filter. To reduce this problem, they proposed the Lazy approach where the coordinator needs to send the update only to the sites that send new readings. By this the overhead processing can be reduced significantly. However this approach is still does not suitable for RFID application because RFID generate too many readings. Even if the reading cycle is set with bigger interval, when the reader performs readings, they can read all the objects in their vicinity repetitively in a short time. There will be a latency to update the coordinator and remote sites each times new reading coming in.

One weakness of Bloom filter is that it does not allow deletion. This is because a single counter in Bloom filter can be hashed number of times by different data. Turning the counter to zero will affect other data that is not involved in the deletion. To solve this problem, Counting Bloom filter (CBF) [14] introduces the use of integer array instead of bit array. It allows the counters in the filter to store 8 bits data rather than 1 bit as in Bloom filter. When a CBF counter is hashed, the counter value will be increased by 1. When deletion is made, the respective counter value will be decreased by 1. Fig. 4 shows the deletion process in CBF. Fig. 4(a) shows that word “Apple” is hashed and inserted into CBF at counter 0, 3 and 5. In Fig. 4(b) word “Orange” is hashed and inserted at counter 1, 3 and 6. Counter 3 which also has been hashed by

“Apple” is increase to 2 in CBF. At Fig. 4(c), the word “Apple” is deleted from CBF, where all the respective counters are decrease by 1.

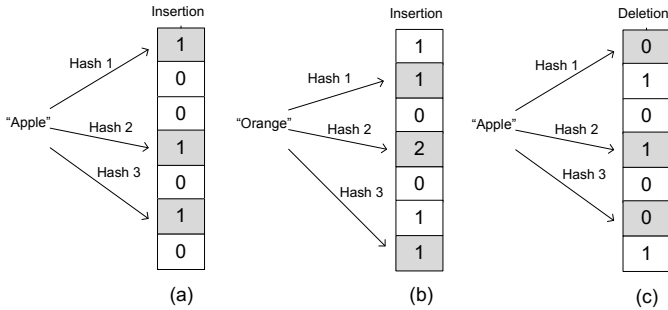


Fig. 4. (a) Insertion of “Apple” in CBF (b) Insertion of “Orange” in CBF (c) Deletion of “Apple” in CBF

From Fig. 4, we can see that to delete element from CBF, the element must be known, otherwise CBF does not have method to identify the original element in its filter. In RFID application, reading that is no longer coming in, can indicate that the object has left the reader vicinity. Therefore the reading can be deleted from memory. However, if we use CBF to filter RFID reading, it cannot remove the old readings by itself, because it cannot identify which counters represent this reading. In [13], they introduce Decaying Bloom filter (DBF) to solve this problem. DBF can delete old readings from its filter automatically by incorporate sliding windows movement in its filter.

The counters in DBF will start with the sliding windows size, and its value will be decreased automatically by one when new reading is coming. If the same reading occurs, all the respective counters will be set back to the sliding window size. If there is no new reading on the same tags, the reading will be evicted from the filter gracefully resulting from decrement process.

The purpose of Bloom filter is to achieve space-efficient with some allowable false positive. The false positive occurs when all the hashed counters is positive for an element that have not been inserted in the filter. At one level, all the hashed elements will be identified as false positive when the Bloom filter has become ‘full’. It is when almost or all the counters have been used to represent the hashed elements. To avoid this, the size of Bloom filter must be big enough to cover the number of elements that going to be inserted in the filter. Old elements must be removed whenever is possible to reduce the probability of false positive.

#### 4 Conclusion

RFID technology is the key to modern supply chain. It offers automation to object identification which surely minimizes businesses cost. However, due to its nature radio frequency can be distorted by external materials and thus create noise and missed readings. This faulty readings need to be filtered to ensured system provide

correct information to the businesses. RFID also has too many duplicate data because of the repeated readings from its reading. In order to filter those readings, one of the methods that can be used is Bloom filter which consume very minimum space with efficient process. It can be used to filter noise and duplicate readings in large RFID data stream. However the challenge is the RFID readings are unpredictable thus it is hard to determine when to clear old readings in the filter. This can lead to the condition where the filter can become ‘full’ and produce false positive results. In future research we planned to overcome this problem to exploit the efficiency of Bloom filter in filtering massive data successfully.

**Acknowledgments.** This work is sponsored by Ministry of Education Malaysia and Universiti Tun Hussein Onn Malaysia.

## References

1. Chawla, V., Ha, D.S.: An overview of passive RFID. *IEEE Communications Magazine*, vol. 45, pp.11-17 (2007) [1]
2. Clappitt, H.G.: *RFID Certification Textbook*. 3rd edition, American RFID Solution, LLC, (2007)
3. Darcy, P., Stantic, B., Sattar, A.: A fusion of data analysis and non-monotonic reasoning to restore missed RFID readings. In *Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp.313-318, Melbourne, Victoria, (2009)
4. Nogee, A.: RFID tags and chips: Opportunities in the second generation. In *Stat/MDR Reports* (2005)
5. Derakhshan, R., Orłowska, M., Li, X.: RFID data management: challenges and opportunities. In *Proceedings of the IEEE International Conference on RFID*, pp. 175-182, Grapevine, Texas (2007)
6. 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*, p. 140, Atlanta, Georgia, (2006)
7. 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 International Symposium on Applications and the Internet Workshops*, pp. 48-51, Phoenix, Arizona (2006)
8. Azambuja, M.C.d., Jung, C.F., Caten, C.S., Hessel, F.P.: RFID-Env: Methods and software simulation for RFID environments. *Business Process Management Journal*, 16, 1014 – 1038 (2010)
9. Bai, Y., Wang, F., Liu, P.: Efficiently filtering RFID data streams. In *Proceedings of the CleanDB Workshop*, pp. 50-57, Seoul, South Korea (2006)
10. Bloom, B.: Space/Time tradeoffs in hash coding with allowable errors. *Commun. ACM*, 13, 422–426 (1970)
11. Dddd
12. Wang, X., Zhang, Q., Jia, Y.: Efficiently filtering duplicates over distributed data streams. In *Proceedings of the International Conference on Computer Science and Software Engineering*, Wuhan, Hubei, 12-14 Dec. 2008; pp. 631 – 634.
13. Mahdin, H., Abawajy, J.: An Approach for Removing Redundant Data from RFID Data Streams. *Sensors*, 11, 9863-9877 (2011).
14. Shen H., Zhang Y.: Improved approximate detection of duplicates for data streams over sliding windows. *Journal of Computer Science and Technology*, 23, 973–987 (2008).
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, 281-293 (2000)