ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20150314114355

APPROXIMATE FILTERING OF REDUNDANT RFID DATA STREAMS IN MOBILE ENVIRONMENT

Guoqiong Liao, Rui Wu, Guoqiang Di, Zhen Shen, Changxuan Wan

Original scientific paper

Recently, RFID technology has been widely used in many applications such as object monitoring and tracing due to the unique features such as non-contact, automatic, fast and multi-target identification simultaneously. However, because of the interference of environmental factors and the requirement of real-time detection, the data collected by the RFID readers are often full of redundancy, which may reduce the processing efficiency of RFID application servers, even lead to making false decisions. Therefore, it is of definite necessity to filter the redundant data in RFID systems before transmitting them to the upper applications. In order to support approximate filtering of RFID data streams in mobile environment, this paper intends to study effective redundant filtering mechanism in the sliding window model. Firstly, we introduce the application background of RFID data streams and the RFID system architecture based on middleware. Then, we propose a temporal-spatial Bloom filter based on sliding windows, which extends the one-dimension array in the standard bloom filter to a two-dimension array, storing both reader IDs and the observed timestamps of original observation items. Meanwhile, in order to guarantee the false positive rate does not increase due to the reason that the space of the filter becomes full, we suggest a random decay strategy for deleting the expired elements. The error rates of the suggested filter, including false positives and false negatives, are analysed in theory. Experimental results show that the suggested filter can filter time redundant data effectively and has a good performance to deal with location movement of RFID objects.

Keywords: *bloom filter; data stream; redundant data filtering; RFID*

Aproksimativno filtriranje redundantnih RFID nizova podataka u mobilnom okruženju

Izvorni znanstveni članak

U zadnje vrijeme RFID tehnologija (Radio Frequency Identification Technology) se naveliko rabi u mnogim aplikacijama kao što su nadgledanje i praćenje objekta, zahvaljujući jedinstvenim značajkama kao što su beskontaktna, brza i simultana identifikacija više ciljeva. Međutim, zbog interferencije faktora okoline i potrebe za detekcijom u realnom vremenu, podaci koje su RFID čitači prikupili često su puni redundancije, a to može smanjiti učinkovitost obrade RFID aplikacijskih servera, pa čak rezultirati i donošenjem krivih zaključaka. Stoga je neophodno potrebno filtrirati redundantne podatke u RFID sustavima prije nego se prenesu do naprednijih aplikacija. U svrhu podržavanja aproksimativnog filtriranja RFID nizova podataka u mobilnom okruženju, u radu se pokušava analizirati mehanizam za učinkovito redundantno filtriranje modelom kliznog prozora. Najprije se daje razvoj aplikacije RFID nizova podataka i arhitektura RFID sustava utemeljeni na međusoftveru. Zatim se predlaže vremensko-prostorni Bloom filtar utemeljen na kliznim prozorima koji proširuje niz podataka s jednom dimenzijom u standardnom Bloom filtru na filtar s dvije dimenzije, pohranjujući i čitača IDs-a i promatrane vremenske oznake originalnih promatranih stavki. U međuvremenu, kako bi se osiguralo da se lažno pozitivna brzina ne poveća zbog toga što se popunio prostor filtra, predlažemo strategiju slučajnog nestajanja za brisanje zastarjelih elemenata. Relativno učestale pogreške predloženog filtra, uključujući lažno pozitivne i lažno negativne, teorijski se analiziraju. Eksperimentalni rezultati pokazuju da predloženi filtar može učinkovito filtrirati vremenski redundantne podatke te uspješno locirati RFID objekte.

Ključne riječi: *RFID; niz podataka; redundantno filtriranje podataka; bloom filtar*

1 Introduction

Radio frequency identification (RFID) technologies allow readers, from a distance without line of sight, to identify the objects associated with unique identifier codes automatically. Because of the unique features such as non-contact, automatic, fast and multi-target identification simultaneously, the technologies are very helpful to monitor, track and trace a large amount of objects in a cost effective way, and have been applied in many domains in recent years, such as monitoring the tagged objects in supermarkets and libraries, tracing drug in supply chains and tracking airline luggage, etc.

However, the adoption of RFID can also bring new issues to the RFID-based applications. For instance, basically RFID readers usually keep constantly sending data about the tagged objects within their detection range, to the application servers according to specified detection cycles, as a result of which plenty of redundant data will be generated. The redundant data, if not filtered, may on the one hand reduce processing efficiency of the application servers, and on the other hand lead to making false decisions. For example, when we check the inventory of a supermarket using RFID technologies, if without filtering the redundant data, the statistics of the items might be wrong since some items may be counted more than once. Likewise, when a car with an RFID tag passes through a highway toll station, if without filtering the redundant data, the car will be deducted more than once. Therefore, it is of definite necessity to filter the redundant data in RFID systems before transmitting them to the upper applications.

The basic idea of traditional strategies in terms of data filtering is sorting and merging, that is, sorting the data stored in the databases first, then detecting the duplicate records by way of comparing adjacent records. This kind of methods include the basic string matching method [1], the recursive matching method [1], the Smith-Waterman algorithm [2], the edition distance method [3], the Cosine similarity function [4], the priority queue algorithm [5], the sorted-neighbourhood method [6], and so on. Whereas, all the methods require the data should be stored in the databases in advance before filtering. Thus, they become invalid for filtering the redundant data in real-time data stream environment.

Currently, hash-based Bloom filters are mainly adopted for approximate filtering of data streams. There are some algorithms suggested, such as the click-stream copy filter [7], the stable Bloom filter [8], the spectral Bloom filter [9], the dynamic count Filters [10], the decaying Bloom filter [11], and so forth. Nonetheless, all these algorithms are only applicable to the environment of the traditional data streams and do not consider the characteristics of RFID data streams.

As for the case of filtering the data in RFID streams, a Redundant Reader Elimination (RRE) algorithm is proposed in [12]. In the algorithm, the readers detect the tagged objects within their ranges and write on the total detection number and the reader's ID for each object. Considering RRE algorithm may fail in some cases, Hsu at al. [13] proposed an algorithm of Layered Elimination Optimization (LEO), which can assure that the first detection can be written on the tags, and can reduce the times of write. Since the detection order about the tags in LEO algorithm is random, its reliability is relatively low. Both of the two algorithms mentioned above advocate that data filtering should be done at the reader side, which needs expensive rewritable tags and the cost is high, hence neither is applied in the RFID system commonly. Mahdin and Abawajy [14] use a counting Bloom filter to remove the redundant RFID data. It also considers that redundancy elimination will be processed at the reader side, so the algorithm will fail if the RFID readers have no independent computing capability. Moreover, the algorithm may produce both false-positive and false-negative errors.

In recent years, filtering redundant data in RFID middleware has obtained more and more attention. RFID Cube [15] is designed to remove redundant data based on data warehouse model. It is designated on the assumption that the set of tagged objects must be moved simultaneously, which therefore is not applicable to the case of a single object moving freely. Moreover, the method is just fit for the data stored in the data warehouse for decayed filtering, so that it cannot be applied to real-time filtering of the RFID data streams. Lee and Chung [16] proposed two kinds of time Bloom filters specifically for RFID data, the Time Bloom Filter (TBF) and its modified version – the Time Interval Bloom Filter (TIBF). Both filters store the detection time or arrival intervals of detected tags in the cells of the filter and then update their values so as to ensure the error rates will not increase because of the "full" of the Bloom filters. However, both are merely applicable to the case of static tags and in the application background where multiple readers are deployed in the same physical region for high reliable detection. Moreover, they do not consider the concept of data stream windows, so they cannot be used for filtering RFID data streams and mobile RFID objects effectively. Mahdin and Abawajy [17] proposed a Comparison Bloom Filter (CBF) based on the landmark window model, which stores the detection times in the filter cells and identifies the reader that the object belongs to by way of comparing the times. However, the algorithm is prone to leading to accidental deletion of valid data and cannot be used for the sliding window model.

In order to support approximate filtering of RFID data streams in mobile environment, this paper intends to study effective redundant filtering mechanism based on the sliding window model. The rest of the paper is arranged as follows. We introduce RFID application background and system architecture in Section 2. In Section 3, we discuss the principle and algorithm of the suggested temporal-spatial Bloom filter. We analyse the error rates of the suggested filter in theory in section 4. In Section 5, we verify and evaluate the performance of the proposed filter in an RFID-based supermarket. We conclude the paper in the end.

2 Background and system architecture

2.1 Background

The data generated by the RFID readers is typically stream data. In some applications, the locations of the tagged objects may be changed with the change of time. For example, in a smart supermarket, a tagged object may be taken from one shelf to another. Therefore, when filtering RFID data streams, it not only considers the observation time but also the observation locations.

As shown in Fig. 1, there are three readers R_1, R_2 and R_3 deployed on appointed shelves S_1, S_2 and S_3, respectively, which detect the tagged objects at a regular detection cycle. It is assumed the detection ranges of different readers do not overlap. Each detection record generated by the readers includes a tag ID, a reader ID, and an observation time. T_1, T_2 and T_3 represent 3 tagged objects. From time 1 to 10, T_2 and T_3 always stay on shelves S_2 and S_3, while the location of T_1 has been changed, namely S_1 (time 2)$\rightarrow S_2$ (time 8)$\rightarrow S_3$ (time 10).

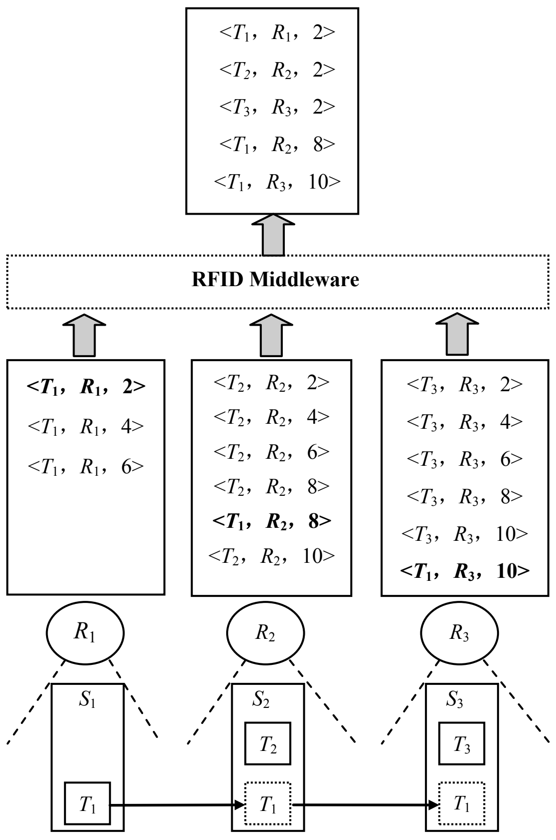

Figure 1 An example of redundant data in RFID data streams

It is assumed in the scenario of Fig. 1, the readers only send one observation record of each tagged object to the application servers in each time window and all the left are considered as the redundant data. It can be seen that from the window between 0 and 10, T_1 is detected 3 times by R_1, 2 of which are redundant. Similarly, since T_2

and T_3 are detected 5 times respectively by R_2 and R_3, 4 of which are redundant. However, T_1's location is changed twice in this window, i.e., it has also been detected once by R_2 and R_3, which cannot be considered as the redundant data. Hence, there should be 5 valid records in Fig. 1, namely <T_1, R_1, 2>, <T_2, R_2, 2>, <T_3, R_3, 2>, <T_1, R_2, 8> and <T_1, R_3, 10>, of which, there are 3 records related to T_1.

Therefore, in the scenario like the smart supermarkets, identifying the redundant data cannot only rely on the detection time but also on the location information, that is, the filtering algorithm should take both time and location information into consideration.

2.2 Redundancy definition

The sliding window model for data streams can be categorized into count-based sliding windows and time-based sliding windows. The former saves the number of the latest arriving k records (k means the size of windows), while the latter saves the data records arriving most recently in the window, which requires the arriving time of the records should be within the range of current time window <t-w-1, t>, where t is the present time and w is the size of the window. This paper will discuss redundant data filtering algorithm based on the time-based sliding window model.

Definition 1. An RFID data stream *DS* is composed of n elements S_1, S_2, …, S_n, among which S_i is a detection triplet <*tid*, *rid*, *ts*>, wherein *tid* represents the unique identification of the tag, *rid* represents the identification of the detection reader and *ts* represents the observation timestamp.

As stated in [18], the redundancy problem is recognized as a serious issue in RFID and sensor networks. Redundancy can happen at two different levels, i.e., reader level and data level.

- Redundancy at reader level occurred when there are more than one reader deployed to cover a specific location.
- Redundancy at data level occurred as data streams. The RFID data can be captured very fast several times in RFID data streams, it is necessary to identify and eliminate those data before their use.

In this paper, we only consider the redundant data in the latter case, i.e., in data stream environment.

Definition 2 (Data redundancy). Supposing W is a time window, *DS* is an RFID data stream. If there are two data $x \in DS$, $y \in DS$ in W, and $y.tid=x.tid$, $y.rid=x.rid$ and $y.ts<x.ts$, then x is regarded as a redundancy data.

According to Definition 2, if the location of an object is changed in a time window, i.e., the object is detected by different logical readers in two adjacent records, the latter will not be identified as a redundant data. Hence, the definition is applicable to filter the RFID objects in mobile environment.

2.3 System architecture

RFID belongs to a kind of automatic identification technologies, which automatically identify objects, collect data about them, and enter those data directly into computer systems with no human intervention.

RFID technologies utilize radio waves to accomplish the function. In general, a basic RFID system consists of three components: RFID tags, RFID readers, and antennas. The RFID systems work like this: the RFID tags contain an integrated circuit and an antenna, which are used to transmit data to the RFID readers; the readers then convert the radio waves to a more usable form of data. Information collected from the tags is then transferred through a communications interface to a host computer system, where the data can be stored in a stream or a database and provide useful information for the predefined applications.

Fig. 2 is the RFID-based application system architecture, which includes three levels:

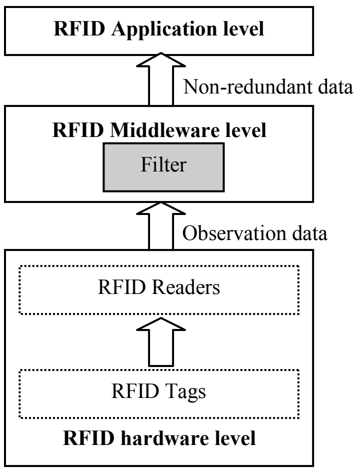

Figure 2 The System Architecture of RFID systems

- *The RFID hardware level*. This level includes RFID tags and RFID readers. The RFID readers receive and transmit data from the RFID tags using the connected antennas through radio waves. The functions of the RFID readers include powering the tags, reading data from or writing data to the tags.
- The *RFID middleware level*. The RFID middleware manages the readers, as well as filters and formats the RFID raw tag data, so that they can be accessed by the various interested enterprise applications. Hence, the middleware is a key component for managing the flow of information between tag readers and enterprise applications.
- *The RFID application level*. Based on the middleware, the RFID system provides various services for end users through this level.

Therefore, before an observation RFID data is sent to the application level, the system will judge whether the data is a redundant data in the RFID middleware. In the structure, the principle of filtering redundant data in RFID streams can be described as follows.

1) When an RFID reader detects an RFID tag, it generates a data item x, and sends it to the RFID middleware.
2) The middleware first extracts the *tid*, *rid* and *ts* in x.
3) Then, the middleware determines whether there is a previous item y in the stream which satisfies the condition: $y.tid=x.tid$, $y.rid=x.rid$ and $y.ts<x.ts$.
4) If the condition is held, x is determined as a redundant data, otherwise, x is not a redundant data.

3 Temporal-spatial Bloom filter

A Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. For it is of no necessity for the Bloom filter to store object identifications, it has been widely applied to the data stream filtering with the advantages such as less query time, less storage space and independence of data amount. This section first gives a brief introduction of the standard Bloom filter, and then designs a temporal-spatial Bloom Filter (TSBF) for filtering redundancy data in RFID data streams.

3.1 Preliminary: the Standard Bloom filter

A standard Bloom filter is traditionally implemented by a single array of M bits, where M is the filter size. On filter creation all bits are reset to zeroes. A filter is also parameterized by a constant k that defines the number of hash functions used to activate and test bits on the filter. Each hash function should output one index in M. When inserting an element on the filter, the bits in the k indexes $h_1(e), h_2(e), \dots, h_k(e)$ are set.

In order to query a Bloom filter, say for element x, it suffices to verify if all bits in indexes $h_1(x)$, $h_2(x)$, … , $h_k(x)$ are set. If one or more of these bits is not set, then the queried element is definitely not present on the filter. Otherwise, if all these bits are set, then the element is considered to be on the filter. Given this procedure, an error probability exists for positive matches, since the tested indexes might have been set by the insertion of other elements.

3.2 The design of TSBF

In order to filter the redundant data defined in Definition 2 effectively, we extend the single array in the standard bloom filter to a two-dimensional array.

Fig. 3 presents the structure of TSBF, from which it can be seen that there are in total m storage cells ($0 \div m-1$) and each one is a two-dimensional array, where one dimension stores reader IDs and the other stores the observation timestamp of the detection record. Hence, the i^{th} cell in the filter can be represented as $M_i[rid][ts]$.

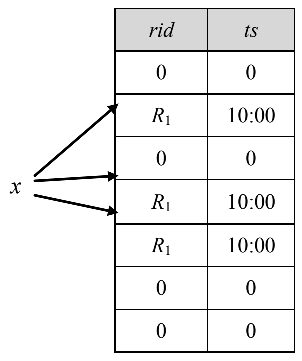

Figure 3 TSBF Structure

First of all, the filter will be initialized and the dimensions both of reader IDs and time of all the cells are set as zero. When a new data arrives, whether it is a redundant data or not is determined by the reader ID and timestamp information stored in the k cells obtained by adopting k hash functions h_1, h_2, …, h_k independently, using *tid* as the index key.

Let w be the size of the sliding windows, the identifying process of the redundant data is as the following:

1) As a data record x enters into the filter, it will be mapped to k storage cells in TSBF by k hash functions, $h_1, h_2, \dots, h_k$, according to $x.tid$.
2) If there is any $i \in \{1, 2, \dots, k\}$ and $M_i[ts] = 0$, it means that the object with $x.tid$ is a new arriving object in current window, and it is identified as a non-redundant data. In the meantime, the timestamp and location information of all the k cells are updated, namely, $M_i\,[ts] = x.ts$ and $M_i[rid] = x.rid$.
3) If all the time dimensions of the k cells are not equal to 0, but there is any $i \in \{1, \dots, k\}$, $x.rid \neq M_i[rid]$, it implies the location of the object has been changed, so it is also identified as a non-redundant data. Meanwhile, the timestamp and location information of all the k cells are updated.
4) When the *rid* of all the k cells are equal to $M_i[rid]$, if there is any $i \in \{1, 2, \dots, k\}$, $x.ts - M_i[ts] > w$, it reveals that the object is the newly arriving one in the present window, so it is also identified as a non-redundant data. Afterwards, the time and location information of all the k cells are updated.
5) Otherwise, the item is identified as a redundant data, which will be abandoned after the time and location information of all the k cells are updated.

3.3 Deletion of expired data

In the sliding time window, the data in the streams will be expired with the change of time, but the information of these expired data still occupies the storage cells if they are not deleted in time. In this way, the storage cells of the filter will become "full" gradually with the change of time, which will lead to reducing accuracy ratios. Hence, the "expired" data should be removed in time from the filter. Whereas, in order to save storage space, all the Bloom filters do not save the values of the index keys, which means we cannot find out in which cells the data is expired.

According to the characteristics of the RFID data streams, if a tagged object is constantly detected by the same reader, its time values saved in the corresponding cells in the filter will be relatively large. On the contrary, if an object is not detected by a reader for a long time, the time values of the cells related with the object keeps the early one.

Therefore, we adopt a random decay strategy for the deletion of expired data. That is, once finishing redundancy identification of a newly arrived record, we select randomly P cells in the filter to minus their time values by 1. The reason is, if the time value of a cell has not been updated for a long time, it will finally become 0 after multiple minus. This procedure equals to the process of deleting the object related to this cell in the filter, so as to avoid the increase of error rates since the filter becomes "full".

3.4 The filtering algorithm of TSBF

Consider a set $A = \{a_1, a_2, \dots, a_n\}$ of n elements. Bloom filters describe membership information of A using a bit vector V of length m. For this, k hash functions, $h_1, h_2, \dots, h_k$ with $h_i: X \rightarrow \{1 \dots m\}$. Bloom filters can be built incrementally: as new elements are added to a set the corresponding positions are computed through the hash functions and bits are set in the filter. The details of the TSBS are shown in Algorithm 1.

For each new arrived data, k hash functions are applied based on *tid* (Line 3÷4), to get k cells. For each cell, it needs to be identified whether its time value is 0. If there is a cell whose time value is 0, the filter should be updated and the data is identified as a non-redundant data (Line 6÷9). Or else, the reader IDs stored in the cells are the same as the reader ID of the newly data. If different, the filter should also be updated and the data is also identified as a non-redundant data (Line 11÷13). Or else, it will judge whether the value that is generated by cutting the time value of the cell from the time value of the data is higher than the size of the window w. If higher, the filter is updated and the data should also be identified as a non-redundant data (Line 15÷17). If none of the above conditions holds, the data is identified as a redundant data and is abandoned (Line 18÷19). Having finished the process of the new arrived data, P cells are selected randomly. If the time values of the selected cell is no less than 1, cut them by 1 (Line 21÷24). If the time value becomes 0, the RID of the cell is reset as 0 (Line 25÷26).

Algorithm 1. ***TSBF-Maintain*( )**

Input: x.tid, x.rid, x.ts

Output: flag: 0 - non-redundant; 1 - redundant

```
1.  begin
2.  //Insertion of new data
3.  for($i$=1; $i \le k$; $i$++)
4.    $p[i]=H_i(x.tid)$;//Hash Functions based on tid
5.  for($i$=1; $i \le k$; $i$++)
6.    if$M_{p[i]}[ts]$=0)
7.      //if the time value of the location of $p[i]$is 0
8.      update-TSBF(x.rid, x.ts); //update the filter
9.      flag=0// non-redundant data
10. else
11.   if $M_{p[i]}[rid] \ne x.[rid]$ //if the rid is different
12.     update-TSBF(x.rid, x.ts);
13.     flag=0; //non-redundant data
14.   else
15.     if $x.ts - M_{p[i]}[ts] > w$
16.       update-TSBF(x.rid, x.ts);
17.       flag=0;
18.     else
19.       flag=1; //redundant data
20. //expired data deleted
21. for ($i$=1; $i \le p$; $i$++)
22.   q=RanSelect-cell; //cells selcted randomly
23.   if$M_q[ts] \ge 1$
24.     $M_q[ts]=M_q[ts]-1$;
25.   if $M_q[ts]$=0
26.     $M_q[rid]$=0;
27. end
```

3.5 The example of TSBF

Fig. 4 shows an example of filtering RFID data stream, in which there are some observation data of T_1 and T_2 on two shelves.

The data stream includes 6 records arriving in sequence : <T_1, R_1, 2 >, <T_1, R_1, 4>, < T_2, R_2, 4>, < T_1, R_1, 12>, < T_2, R_2, 16> and< T_1, R_2, 16>. The size of the window is 10.

1	2	1	2	1	1	1	1	1	12	2	16
0	0	0	0	2	4	2	3	2	16	2	16
1	2	1	2	1	1	1	1	1	12	2	16
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	4	2	3	2	16	2	16
1	2	1	2	1	1	1	1	1	12	2	16
0	0	0	0	2	4	2	4	2	16	2	16
0	0	0	0	0	0	0	0	0	0	0	0

<T_1, R_1, 2> <T_1, R_1, 4> <T_2, R_2, 4> <T_1, R_1, 12> <T_2, R_2, 16> <T_1, R_2, 16>

(a) (b) (c) (d) (e) (f)

Figure 4 An Example of TSBF

1) When it starts，<T_1, R_1, 2> arrives at the filter. Assumed the data is mapped to the cells of 0, 2 and 5 by using the hash functions. For the data is a newly arrived data, the domains both of the reader and the time of the three cells are updated, *rid* = 1, *ts* = 2, which is shown in Fig. 4a;
2) When <T_1, R_1, 4> arrives, its *tid* is the same as that of the former record, so they are mapped to the same locations in the filter. Moreover, since the reader IDs of the three cells are the same and the time distance is $4-2 < 10$, this data is identified as a redundant one with no need of updating the filter as is shown in Fig. 4b;
3) When <T_2, R_2, 4> arrives，it is mapped to the cells of 1, 4 and 6. The timestamps of all the three cells are 0, so it is identified as the newly arrived data and the filter is updated, which is shown in Fig. 4c;
4) When <T_1, R_1, 12> arrives, since the corresponding *rid* of the cells is the same as that of the new data record, and the time distance is $12-1 > 10$, the filter should be updated as shown in Fig. 4d;
5) When<T_2, R_2, 16> arrives, the corresponding cells of T_2 are updated, which is shown in Fig. 4e;
6) When <T_1, R_2, 16> arrives, for the *rid* of the data record is different from the *rid* of the cells in the filter, the object location is changed. Thus, the filter is updated as shown in Fig. 4f.

It should be noted that Fig 4 only considers how to manage the newly arrival data in the data streams, but does not consider the deletion of the expired data in the streams.

4 An analysis of TSBF's error rates

One prominent feature of Bloom filters is that there is a clear trade-off between the size of the filter and the rate of false positives.

Just like the traditional Bloom filters, TSBF also generates false positives, that is, identifies a non-redundant item as a redundant one. On the other side, since TSBF will delete the expired elements through the decay strategy, TSBF also generates false negatives, that is, identifies the redundant item as a non-redundant one. This section will analyse the error rates of TSBF including false positives and false negatives.

Theorem 1. The false positive rate of TSBF is:

$$FPR=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}, \tag{1}$$

wherein k represents the amount of hash functions, m represents the amount of cells in the filter, and n represents the amount of the tagged objects inserted in the filter.

Proof: Supposing the hash functions in the TSBF Filter is simple and random, each element can be mapped to m cells in an equal probability with no relation to which other elements are mapped to. In this case, for a particular cell i, its probability without being selected under a specific hash function when inserting a new element is:

$$p_i=1-\frac{1}{m}. \tag{2}$$

Thus, its probability of not being mapped by any one in k hash functions is:

$$p_k=\left(1-\frac{1}{m}\right)^{k}. \tag{3}$$

If n objects are inserted, none of them sets the cell is:

$$pk_n=\left(1-\frac{1}{m}\right)^{kn}. \tag{4}$$

Hence, the probability that the cell is set is

$$pz_i=1-\left(1-\frac{1}{m}\right)^{kn}. \tag{5}$$

In the query stage, if all k cells corresponding to an inquired object are set, it can be identified a redundant data. Thereby, the probability of wrong judgment is:

$$FPR=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}. \tag{6}$$

Theorem 2. The false negative rate of TSBF is:

$$FNR=1-\prod_{i=1}^{h}\left(1-p_i\right), \tag{7}$$

wherein,

$$p_i=C_k^x\left(\frac{n}{m}\right)^{x}\left(1-\frac{n}{m}\right)^{k-x}\left[\sum_{i=1}^{x}\frac{h}{m}\left(1-\frac{h}{m}\right)^{i-1}\right], (x>0) \tag{8}$$

Among which, m represents the size of the filter, n represents the amount of cells selected randomly subtracted by 1, k represents the times subtracted in total in the filter, x represents the times subtracted for a certain cell, h represents the amount of the hash functions.

Proof: The false negative of TSBF happens due to the reason that the timestamp of the cells is subtracted by 1 randomly. Firstly, the probability of a certain cell subtracted by 1 is:

$$f_n=C_k^x\left(\frac{n}{m}\right)^{x}\left(1-\frac{n}{m}\right)^{k-x}. \tag{9}$$

There exists a time interval for the new data records that the false negative is generated.

Supposing the time value of the present data in the filter is t and the size of the sliding windows is w. According to Definition 2, when the time value of the other data with the same TID is more than $t + w$, the present data is definitely non-redundant data. However, since a false negative occurs in this cell, that is, if the time value is subtracted by 1 for x times, the time value of this cell should become into $t - x$. This means when the time value of the other data with the same *tid* is less than $t - x + w$, the present data is also definitely identified as redundant data. Therefore, the time interval generating the false negatives is between $t + w$ and $t - x + w$.

Therefore, the probability of the sequent data with the same TID arriving in this time interval and subtracted by 1 after applying hash functions is as follows:

$$p_l=\sum_{i=1}^{x}\frac{h}{m}\left(1-\frac{h}{m}\right)^{i-1}. \tag{10}$$

Hence, the probability of generating false negative of a certain cell in the filter is:

$$p=f_n\cdot p_l=C_k^x\left(\frac{n}{m}\right)^{x}\left(1-\frac{n}{m}\right)^{k-x}\left[\sum_{i=1}^{x}\frac{h}{m}\left(1-\frac{h}{m}\right)^{i-1}\right]. \tag{11}$$

The false negative error is one of the k hashed cells of the data is wrong, so the false negative rate of TSBF is:

$$FNR=1-\prod_{i=1}^{h}\left(1-p_i\right). \tag{12}$$

It can be concluded that Eq. (7) is verified.

5 Performance experiment and analysis

5.1 Experiment preparation

We built a real RFID-based smart supermarket with 10 readers and 200 tags, and collected the data within 1 hour. Based on the data distribution large amount of simulation data are also generated. The number of the readers extends from 20 to 50, and the number of the tagged objects extends from 400 to 1200. Meanwhile, object location movement among different shelves is simulated as well.

The performance measures of the experiments are the false positive rate (FPR) and the false negative rate (FNR) by comparison with TIBF Algorithm in [16]. Since there is no false negative in TIBF Algorithm, the comparison is only applied to false positive rate.

The hardware environment is Intel Core i3-2330M CPU with 2GB memory; the software environment is Microsoft Windows 7 and Microsoft Visual C++ 6.0.

5.2 Experimental results and analysis

1) The influence of the amount of the hash functions on the error rates.

Supposed the size of the filter is 1 × 104, the amount of the tagged objects is 1 × 105, the size of the sliding window is 10.

It can be seen from Figs. 5 and 6, when the amount of hash functions is between 2÷4, both FPR and FNR of the TSBF method are relatively small, especially when the amount of hash functions is 2, the FPR and FNR are the smallest. The reason is that there is a high probability for different data stored into the same cell if only using 1 hash function, leading to higher FPR and FNR.

When there are 2÷4 hash functions, the probability of storing different data into the same cell is relatively low, so FPR and FNR are low correspondingly. Whereas, after that, when the amount of hash functions increases, the filter cells saving the same data will increase as well, which means the load of the filter becomes heavy and the FPR and FNR increase instead.

It can be seen from Fig. 7, FPR of TSBF becomes smaller as the size of the filter enlarges, which is in accordance with the theory analysis result in Section IV. In this figure, the reason for the slight increase of FPR at the 4 × 104 owes to the selection of hash functions. In this experiment, two separate hash functions are selected, both of which might produce collision, leading to the increase of FPR.

Additionally, Fig. 7 also shows that FPR of TSBF Algorithm is much smaller than that of TSIF. The reason lies in the judgment on the reader IDs in TSBF. When the reader ID of a data with the same TID changes, namely the location of the tagged object is changed, TSBF can identify this kind of changes accurately. Yet there is no identification on the reader IDs in TIBF. If the move time of an object is long, TIBF can identify the location movement through the change of time intervals. But if an object’s locations are changed frequently, TIBF may make wrong decisions, resulting in relatively large FPR.

Fig. 8 presents the change of FNR of TSBF, in which the rate is kept at a very small range. The reason of this occasion depends on the deletion algorithm that reduces the randomly-selected cells by 1, after which the probability that the next arrival object is hashed to the same cell is very small. Accordingly, FNR is small as well.

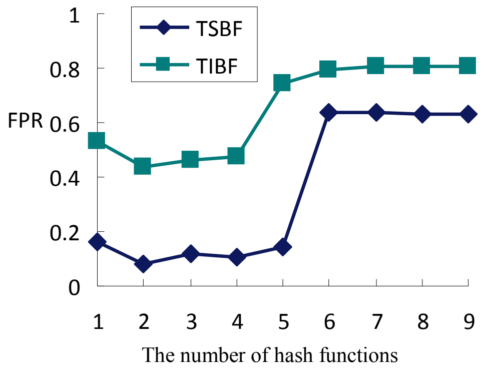

Figure 5 The influence of the amount of hash functions on FPR

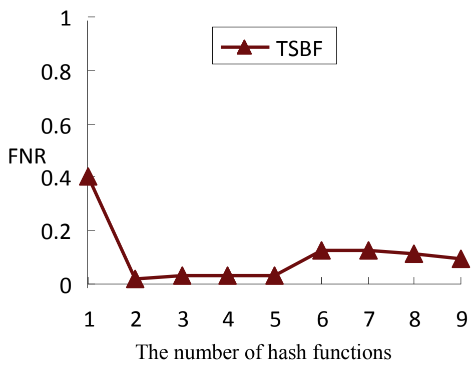

Figure 6 The influence of the amount of hash functions on FNR

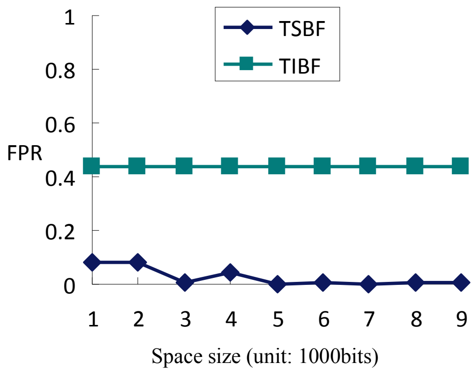

Figure 7 The influence of the size of the filter on FPR

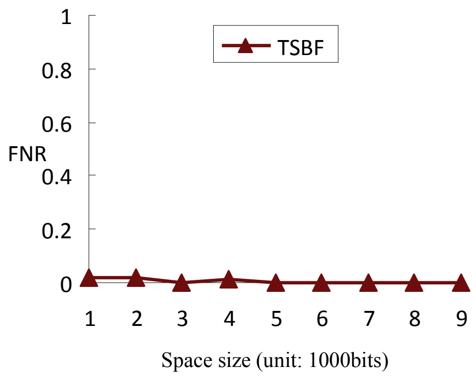

Figure 8 The influence of the size of the filter on FNR

3) The influence of the amount of data streams on the error rates

Supposing the size of the filter is 1×104, the size of the sliding window is 10, and the amount of the hash functions is 2. Fig. 9 and Fig. 10 show the changes of FPR and PNR under the change of the amount of data streams.

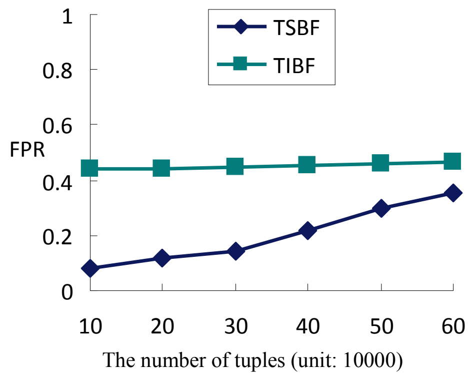

Figure 9 The influence of the amount of data streams on FPR

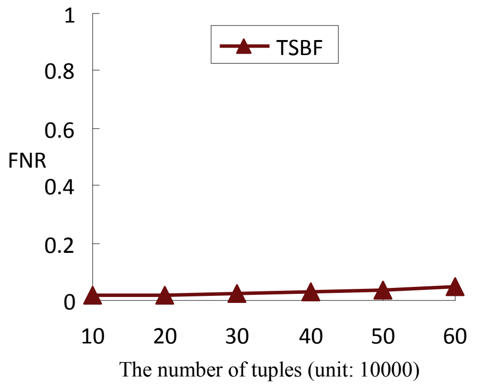

Figure 10 The influence of the amount of data streams on FNR

We can see from Fig. 9, when the amount of data is smaller than 3 × 105, the increase of FPR of TSBF is relatively slow while it is faster when the amount of data is more than 4 × 105. The reason lies in the deletion algorithm of expired elements in TSBF, which can assure that error rates will not multiply because of increasing data streams. However, the deletion algorithm is unable to delete expired data when data stream multiplies to some extent, so FPR after this period increases fast. In this respect, the FPR of TIBF does not change with the change of the amount of data stream generally. On the whole, TSBF is more effective than TIBF considering performance.

Fig. 10 shows the change of FNR of TSBF. It is apparent that FNR of TSBF keeps at a very small level because of periodical deletion on the expired element.

4) The influence of the sliding window size on the error rates

Supposing the size of the filter is 1 × 104, the amount of RFID data is 1 × 105, the amount of the hash functions is 2. Fig. 11 and Fig. 12 show the change of FPR and FNR with change of the size of the sliding windows.

Fig. 11 reveals that FPRs of both TSBF Algorithm and TIBF Algorithm increases as the size of the sliding windows enlarges. Yet this occasion is different from the ones previously in that FPR increases though data generating false positives do not increase. The reason is that the total amount of data decreases after filtered with the enlargement of the window size.

As for Fig. 12, it can be seen that FNR of TSBF changes hardly with the change of the size of the window, which it owes to the amount of data both arriving at the filter and generating false negatives decreases.

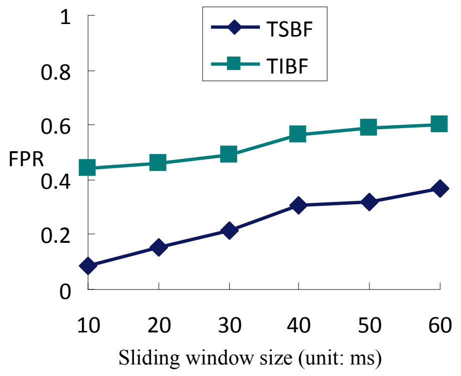

Figure 11 The influence of the sliding window size on FPR

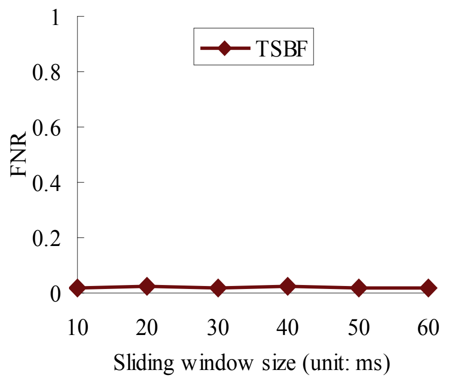

Figure 12 The influence of the sliding window size on FNR

6 Conclusions

Focusing on eliminating redundant data in RFID data stream, this paper studies an approximate filtering strategy, i.e., the temporal-spatial bloom filter, for RFID data streams in mobile environment. In the filter, each cell is presented as a two-dimensional array for storing reader's IDs and observation time, which can support location movement of the RFID objects. The error rates of the filter, including false positives and false negatives, are analysed in theory. Through experiment comparison and analysis, the suggested filter is verified as feasible and effective.

In the future, we will study the redundant RFID data filtering for uncertain RFID data streams.

Acknowledgements

This work is supported partly by NSF of China (No. 61262009, 61262033), NSF of Jiangxi, China (20122BAB201032), Science Foundation of Jiangxi Provincial Department of Education, China (No. GJJ10694, GJJ12259), Building Project of Superior Innovation Team of Science and Technology of Jiangxi, China (No. 20113BCB24008).

7 References

[1] Li X.; Liu B.; Xu, R. Research of String Matching Techniques. // Computer Engineering. 30, 22(2004), pp. 24-26.

[2] Monge, A. E.; Elkan, C. The field matching problem: algorithms and applications. // Proceedings of the 2nd International Conference of Knowledge Discovery and Data Mining / Portland, 1996, pp. 267-270.

[3] Masek, W. J.; Paterson, M. S. A faster algorithm computing string edit distances. // Journal of Computer and System Sciences. 20, 1(1980), pp. 18-31. DOI: 10.1016/0022-0000(80)90002-1

[4] Sahon, G.; McGill, M. J. Introduction to modem information retrieval. New York, McGraw Hill, 1983.

[5] Monge, A. E.; Elkan, C. P. Efficient domain-independent detection of approximately duplicate database records. // Proceedings of the 2nd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery /Tucson, US. 1997, pp. 23-29.

[6] Hernández, M. A.; Stolfo S. J. Real-world data is dirty: data cleansing and the merge/purge problem. // Data Mining and Knowledge Discovery. 2, 1(1998), pp. 9-37. DOI: 10.1023/A:1009761603038

[7] Metwally, A.; Agrawal, D.; Abbadi. A. E.Duplicate detection in click streams. // Proceedings of the 14th International Conference on World Wide Web / Chiba, Japan, 2005, pp. 12-21. DOI: 10.1145/1060745.1060753

[8] Deng, F.; Rafiei, D. Approximately detecting duplicates for streaming data using stable bloom filters. // Proceedings of the 25th ACM SIGMOD International Conference on Management of Data / Chicago, 2006, pp. 25-36. DOI: 10.1145/1142473.1142477

[9] Cohen, S.; Matias, Y. Spectral bloom filters. // Proceedings of the 34th ACM SIGMOD International Conference on Management of Data /San Diego. 2003, pp. 241-252. DOI: 10.1145/872757.872787

[10] Aguilar-Saborit, J.; Trancoso,P.; Muntes-Mulero, V. et al. Dynamic count filters. ACM SIGMOD Record, 35, 1(2006), pp. 26-32. DOI: 10.1145/1121995.1122000

[11] Shen, H.; Zhang, Y. Improved approximate detection of duplicates for data streams over sliding windows. // Journal of Computer Science and Technology. 23, 6(2008), pp. 973-987. DOI: 10.1007/s11390-008-9192-1

[12] Carbunar, B.; Ramanathan, M K.; Koyuturk, M, et al. Redundant-reader elimination in RFID systems. // Proceedings of the 2nd Communications Society Conference on Sensor and Ad Hoc Communications and Networks / Santa Clara, 2005, pp. 176-184. DOI: 10.1109/sahcn.2005.1557073

[13] Hsu, C. H.; Chen, Y. M.; Kang, H. J. Performance-effective and low-complexity redundant reader detection in wireless RFID networks. // EURASIP Journal on Wireless Communications and Networking. 22, 3(2008), pp. 138-145. DOI: 10.1155/2008/604747

[14] Mahdin, H.; Abawajy, J. An approach to filtering duplicate RFID data streams. // Proceedings of the 2nd International Conference on U- and E-service, Science and Technology / Jeju Island, 2010, pp. 125-133. DOI: 10.1007/978-3-642-17644-9_14

[15] Gonzalez, H.; Han, J.; Li, X. et al. Warehousing and analyzing massive RFID data sets. // Proceedings of the 22nd International Conference on Data Engineering / Atlanta, 2006, pp. 1-10. DOI: 10.1109/icde.2006.171

[16] Lee, C. H.; Chung, C. W. An approximate duplicate elimination in RFID data streams. // Data & Knowledge Engineering. 70, 12(2011), pp. 1070-1087. DOI: 10.1016/j.datak.2011.07.007

[17] Mahdin, H.; Abawajy, J. An approach for removing redundant data from RFID data streams. // Sensors. 11, 10(2011), pp. 9863-9877. DOI: 10.3390/s111009863

[18] Derakhshan, R.; Orlowska, M. E.; Li, X. RFID data management: challenges and opportunities. // Proceedings of 2007 IEEE International Conference on RFID / Texas, 2007, pp. 175-182. DOI: 10.1109/RFID.2007.346166

Authors' addresses

Guoqiong Liao
School of Information Technology, Jiangxi University of Finance and Economics,
Nanchang 330013, China
E-mail: liaoguoqiong@163.com

Rui Wu
School of Information Technology, Jiangxi University of Finance and Economics
Nanchang 330013, China
E-mail: wurui114@qq.com

Guoqiang Di
School of Information Technology, Jiangxi University of Finance and Economics
Nanchang 330013, China
E-mail: dylydgq@21cn.com

Zhen Shen
School of Information Technology, Jiangxi University of Finance and Economics
Nanchang 330013, China
E-mail: vipqqxl@163.com

Changxuan Wan
School of Information Technology, Jiangxi University of Finance and Economics
Nanchang 330013, China
E-mail: wanchangxuan@263.net