I. J. Communications, Network and System Sciences, **2009, 1, 1-89**
Published Online February 2009 in SciRes (http://www.SciRP.org/journal/ijcns/).

Synchronous Dynamic Adjusting: An Anti-Collision Algorithm for an RF-UCard System

Jichang CAO, Lin SHU, Zhengding LU
School of Computer Science & Technology
Huazhong University of Science & Technology, Wuhan, China
Email: shulin3760@163.com
Received September 27, 2008; revised December 6, 2008; accepted December 8, 2008

Abstract

An RF-UCard system is a contactless smartcard system with multiple chip operating systems and multiple applications. A multi-card collision occurs when more than one card within the reader's read field and thus lowers the efficiency of the system. This paper presents a novel and enhanced algorithm to solve the multi-card collision problems in an RF-UCard system. The algorithm was originally inspired from framed ALOHA-based anti-collision algorithms applied in RFID systems. To maximize the system efficiency, a synchronous dynamic adjusting (SDA) scheme that adjusts both the frame size in the reader and the response probability in cards is developed and evaluated. Based on some mathematical results derived from the Poisson process and the occupancy problem, the algorithm takes the estimated card quantity and the new arriving cards in the current read cycle into consideration to adjust the frame size for the next read cycle. Also it changes the card response probability according to the request commands sent from the reader. Simulation results show that SDA outperforms other ALOHA-based anti-collision algorithms applied in RFID systems.

Keywords: RF-UCard, Anti-collision Algorithm, Synchronous Dynamic Adjusting, RFID, ALOHA, DFSA, BBEI

1. Introduction

Identification is a central concept in user-oriented and ubiquitous computing. Radio Frequency Identification (RFID) is one of the key technologies for identifying physical objects permits remote, non-line-of-sight, and automatic reading. There is a wide variety of RFID products and applications available; the book [1] provides a good overview. A contactless smartcard promises to be a typical instance of the RFID technology, e.g. close-coupling cards (ISO/IEC 10536), proximity cards (ISO/IEC 14443), and vicinity cards (ISO/IEC 15693) [2]. Contactless smartcards often show more powerful processing ability and sufficient storage capacity than RFID tags, which benefits from the card architecture with a microcontroller unit and writeable memories. A Radio Frequency Universal Smart Card (RF-UCard) is a novel contactless smartcard platform with multiple chip operating systems (COS) and multiple applications environment [3]. Multiple COSes from different vendors can coexist on a single card, and additional COSes can be loaded after card issuing. In addition, multiple applications can be hosted by a single COS, and the application can be dynamic downloaded onto or unloaded from the card.

An RF-UCard system is often composed of three main components as shown in Figure 1.

- One or more RF-UCards, held by the users to identify. RF-UCards consist of three layers, the application, the operating system and the physical layers. The application and the operating system layers host multiple applications and COSes respectively. The physical layer includes a microcontroller unit, memories and the coiled antenna. RF-UCards could be either active or passive. Active cards are partly or fully battery powered, have the capability to communicate with other cards, and can initiate a request to the reader. Passive cards, on the other hand, do not have any internal power source but are powered up by the reader.
- One or more readers, made up of a control unit and an RF module. Its main functions are to activate the cards, initiate the communication with the cards, collect the card responses, and transfer data between the back-end server and a card. The reader is usually equipped with a single COS, and could be either mono-functional or multi- functional. The mono-functional reader merely supports the single application, and no the third party is involved in the communication. The multi-functional reader contains several independent applications, but

only one application can be activated by the user at the beginning of the communication.

- A back-end server, which contains various information about RF-UCards and applications.

The reader and RF-UCards communicate over a shared wireless channel. A read process is initiated by the reader that uses radio to broadcast periodically a request command to the RF-UCards. Each valid card[1] within the reader's read field sends its ID to the reader while it receives a request. If only one card responds, the reader can successful receive the card's ID. When more than one card responds simultaneously, messages from cards will collide and cancel each other out at the reader. This problem is referred to as the "multi-card collision", which is very similar to the "multi-tag collision" in RFID systems. Collisions can defer the transmission delay and lower the identification efficiency and cards often lose their usefulness. Hence, an anti-collision algorithm needs to be devised between the reader and the cards to minimize collisions.

The well known algorithms devised to resolve the multi-tag collision problem in RFID systems can be grouped into two broad types, namely deterministic algorithms and stochastic algorithms [1]. Deterministic algorithms resolve collisions by splitting a set of colliding tags into two subsets and attempt to recognize the subsets one by one. The typical instances of deterministic algorithms are the binary tree algorithm [4–6] and the query tree algorithm [7,8]. Stochastic algorithms are usually based on an ALOHA-like protocol in which the tags send their data at a random time period. The ALOHA-based algorithms include pure ALOHA [9], slotted ALOHA [10], static framed ALOHA [11], and dynamic framed ALOHA [12,13].

An RF-UCard system is much different from an RFID system in identification though they both communicate over a radio channel. In an RFID system, all tags within the read field will send back their responses. However, due to the fact that cards are in general equipped with multiple COSes and applications, whereas the reader always hosts the single COS, only the valid cards will send back their responses in an RF-UCard system. All cards will perform a validity check after receiving a request command. In addition, tags and cads are somewhat different in the arrival mode. Tags are usually attached to the objects and arrive at the read field in a batch mode (e.g. in a supply chain), whereas cards are often held by users and arrive in a single mode. Furthermore,

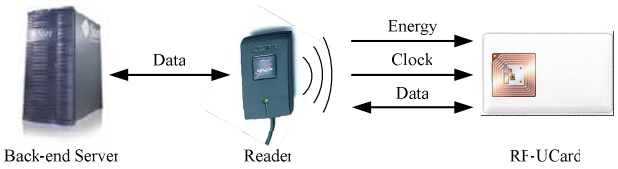


Figure 1. RF-UCard system architecture.

[1]The valid card is the card that the reader's COS has been proper loaded onto it.

most ALOHA-based algorithms applied in RFID systems assume the scenario for tag identification is static, i.e. a set of tags enter the read field and stay there until all tags are identified. No new tag arrives during the identification process. Unfortunately, this scenario is not suitable for RF-UCard systems that the card quantity is dynamic changed since the card arrival occurs randomly. Finally, anti-collision algorithms applied in RFID systems mainly focus on the tag identification but does not care what further to do after the tags have been identified. However, in an RF-UCard system, the anti-collision algorithm needs not only to identify cards, but to process cards. Thus the read time in an RF-UCard system can be divided into two parts: the identification time (i.e. the time needs to identify a card) and the processing time (i.e. the time needs to execute a specific application). Hence, in order to apply the idea involved in ALOHA based algorithms to RF-UCard multiple accesses, the algorithms need to be revised according to the characteristics of an RF-UCard system.

We propose a combinatory anti-collision algorithm, called *synchronous dynamic adjusting algorithm* (*SDA*) for multi-card collision resolution in RF-UCard systems. SDA employs a two-sided synchronous adjusting scheme that can synchronize to adjust the frame size in the reader side and the response probability in the card sides. We focus our attention on adjusting the frame size and the card response probability by exploiting information obtained from the last read cycle. The estimated card quantity and the new arriving cards in the current read cycle are both taken into consideration to adjust the frame size for the next read cycle. The card response probability changes according to the request commands sent from the reader. These adjusting schemes reduce the collisions and as a result can facilitate card identification with shorter delay and better efficiency. Simulation results show that SDA suppresses the occurrence of collisions and shortens the total read time and delay time while preserving better identification efficiency.

The rest of the paper is organized as follows. Section 2 reviews existing ALOHA based anti-collision algorithms and Section 3 reviews some mathematical tools used for the SDA design. Section 4 gives a detail description to the synchronous dynamic adjusting scheme and the procedure of multi-card collision resolution using SDA. The extensive simulations are conducted in Section 5 to show the performance of SDA versus different parameters, and to further compare SDA with two ALOHA based algorithms. Finally, the conclusions of our analysis are presented in Section 6.

2. ALOHA Based Anti-Collision Algorithms

ALOHA based anti-collision algorithms reduce the occurrence probability of tag collisions since tags transmit at the distinct time. In pure ALOHA, tags randomly select their transmission time and, in slotted ALOHA, tags is limited to transmit only at the beginning

of a time slot with a certain time period. In framed ALOHA, the reader sends the frame size and a tag randomly selects a slot number in the frame for the data transmission. Static framed ALOHA uses a fixed frame size and does not change the size during the tag identification process. On the other hand, dynamic framed ALOHA improves the identification efficiency by dynamically changing the frame size according to the amount of tag responses in the previous read cycle. We here give detailed descriptions about two typical ALOHA based anti-collision algorithms applied in RFID systems.

2.1. Dynamic Framed Slotted ALOHA Algorithm

The dynamic framed slotted ALOHA (DFSA) has been studied extensively and shows the best performance of ALOHA based algorithms. In DFSA, the reader always broadcasts a request command at the beginning of a frame to the tags within the read field, and then waits a certain amount of time slots for tag responses. Tags randomly select a time slot in the frame to send back their IDs. Within a read cycle, the reader can collect the information about the number of the empty slots, the slots occupied by one tag, and the slots occupied by more than one tag. The slot that occupied by one tag means a tag has been successful identified, and the slot that occupied by more than one tag means a collision occurs. One more read cycles needed if collisions occur. For each read cycle, the reader dynamically adjusts the frame size according to the amount of tag responses in the previous cycle. The analysis of DFSA algorithms mainly pays attentions to two primary issues [14]. The first one is how to estimate the tag quantity within the read field according to the responses in the past. The other one is how to determine an optimal frame size for the next read cycle to achieve maximum efficiency. Recently, many researchers focus on these key issues to improve the overall performance of DFSA. As a result, some valuable methods to estimate the tag quantity and to adjust the frame size are proposed [13−17]. Results revealed that the efficiency of DFSA is very dependent on the initial frame size and the maximum efficiency occurs when the frame size equals the number of tags.

2.2. Bi-directional Binary Exponential Index Algorithm

The Bi-directional Binary Exponential Index (BBEI) algorithm [18] is originally inspired from the binary exponential backoff algorithm, which is commonly used to schedule retransmissions after collisions in Ethernet networks [19]. BBEI is based on a slotted ALOHA. Unlike DFSA, it assumes that all tags within the read field respond with a certain probability $p(0 < p \le 1)$ [2]. The reader broadcasts one of three request commands (e.g. EMPTY, SUCCESS, and FAIL) at the beginning of each time slot according to the response results in the previous slot (if the slot is idle, the command is EMPTY; if the slot is occupied by one tag, the command is SUCCESS; and if the slot is occupied by more than one tag, the command is FAIL). While a tag receives the request command, it first adjusts its response probability according to the command type and then sends back its response with the newly probability. The available values for the response probability are the inverses of the binary exponential values, such as {1,1/2,1/4,1/8,1/16, 1/32,1/64,1/128,1/256,1/512,1/1024}. If the received command is EMPTY, the tags multiply its probability by two; if the command is FAIL, the tags divide its probability by two; others, the probability keeps no change.

[2]Each tag has an initial response probability when it enters the reader's read field.

3. Mathematical Basis

This section reviews some mathematical tools we will use in subsequent sections. The read time is divided into discrete intervals (slots) with fixed length. The duration of a slot is sufficient for a card to send back its response. The number of time slots that the reader needs to wait after broadcasting a request command to cards is called "frame size" and will be denoted by *N*. The number of cards is usually denoted by *n*.

3.1. Poisson Process

The Poisson process is a continuous-time counting process which is memoryless and orderly. It applies to many cases where a certain event occurs at different points in time. Consider the card arrivals in a real-world RF-UCard system also occur at different points in time, and are often independent of each other. Thus it is reasonable to assume that the sequence of card arrivals in an RF-UCard system is a Poisson process with the arrival rate λ. Let t_i be the *i*th arrival time and $\tau_i = t_i - t_{i-1}$ be the *i*th interarrival time. Owing to the properties of the Poisson process, $\{\tau_i, i \ge 1\}$ is a sequence of independent exponentially distributed random variables with the same distribution $F(t) = 1 - e^{-\lambda t}, t \ge 0$ and the probability density function (pdf) $f(t) = \lambda e^{-\lambda t}, t \ge 0$. As a result, the mean interarrival time can be given by

$$E(\tau) = \frac{1}{\lambda} \quad \textbf{(1)}$$

Let $\{n(t), t \ge 0\}$ be the number of cards that arrived within the interval $(0,t)$, the probability distribution of $n(t)$ depends only on the length of the interval, and can be expressed as

$$P\{(N(t+s) - N(s)) = k\} = P\{N(t) = k\} = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, t \ge 0 \quad \textbf{(2)}$$

Furthermore, we can obtain the expected number of card arrivals within a given interval based on the probability distribution of $n(t)$, that is

$$E(N(t)) = \sum_{k=0}^{\infty} k \cdot P\{N(t)=k\} = \sum_{k=0}^{\infty} k \cdot \frac{(\lambda t)^k e^{-\lambda t}}{k!} = \lambda t \quad \textbf{(3)}$$

3.2. Occupancy Problem

The allocation of cards to time slots within a frame is equivalent to the well known occupancy problem [20] that deals with the random allocation of balls to a number of bins where one is, e.g., interested in the number of filled bins. In the following, we will substitute "balls" and "bins" with "cards" and "slots".

Given N slots and n cards, the number k of cards in one slot is binomially distributed with parameters n and $\frac{1}{N}$:

$$B_{n,\frac{1}{N}}(k) = \binom{n}{k} (\frac{1}{N})^k (1-\frac{1}{N})^{n-k} \quad \textbf{(4)}$$

The number k of cards in a particular slot is called the occupancy number of the slot. The distribution (4) can apply to all N slots, thus the expected value of the number of slots with occupancy number k is given by $a_k^{N,n}$

$$a_k^{N,n} = NB_{n,\frac{1}{N}}(k) = N\binom{n}{k} (\frac{1}{N})^k (1-\frac{1}{N})^{n-k} \quad \textbf{(5)}$$

This is a crucial equation because we will use it to estimate the card quantity in SDA and DFSA.

4. Synchronous Dynamic Adjusting Algorithm

The DFSA algorithm only changes the frame size according to the estimated tag quantity to improve the tag identification efficiency. However, as the number of tags becomes much larger than the frame size, the occurrence of tag collisions increases rapidly. This is mainly due to the fact that all tags within the read field send back their responses with the probability 1. On the other hand, the BBEI algorithm merely resorts to changing the tag response probability to reduce collisions. Unfortunately, due to its single-slot property, the total identification time will increases sharply as the number of tags increases. SDA is developed by integrating the ideas involved in DFSA and BBEI that adjusts the frame size and the response probability synchronously. This combinatory scheme will improve the system efficiency by reducing the card collisions. In this section, we give a detail description to the employed adjusting schemes and the procedure of collision resolution using SDA.

4.1. Programming Interface

The programming interface of SDA is both provided by the reader and the RF-UCards. It comprises some communication commands, functions, and local variables, described below.

- *Req-COS*(*CID*, *N*, f_T): the command is sent by the reader at the beginning of a frame to initiate the communication with cards within the read field. Three parameters are included that *CID*, *N*, and f_T , denote the unique identifier of the reader's COS, the current frame size, and the traffic intensity of the system in the current read cycle, respectively.
- *Req-app* (*UID*, *AID*): the command is sent by the reader to the identified card to execute the specific application. The parameters, *UID* and *AID*, denote the card's identifier and the unique identifier of the application respectively.
- *Res*(*UID*): the command is sent by a valid card to the reader with its unique identifier (*UID*).
- *Valid-check*(*CID*): the function is performed by each card in the read field to validate itself to the reader with reader's *CID*.
- *Slot-select* (*N*): the function is performed by a card to random select a slot from *N* slots in the frame.
- < c_0, c_1, c_x >: a triple of numbers that quantify the slots in different states. For a given slot, there are only three possible states: empty (occupied by no card), success (occupied by one card), and collision (occupied by more than one card). c_0, c_1 and c_x denote the number of empty slots, success slots, and collision slots in a frame respectively.
- S_{UID}: a set of card *UID*s that have just been identified in the current frame. The set is held temporarily by the reader to send the *Req-app* (*UID*, *AID*) command.
- *ResProValues*[]: an array over which a card response probability p can range ($0 < p \leqslant 1$). Let k be the index of *ResProValues* so that p= *ResProValues*[k].
- f_T : a flag representing the traffic intensity of the system in a read cycle: loose ($f_T = -1$), moderate ($f_T = 0$), and crowded ($f_T = 1$).
- f_C: a flag representing one of three possible states a card may be in during a read cycle: invalid ($f_C = -1$), sleep ($f_C = 0$), and active ($f_C = 1$).

4.2. Estimation of Card Quantity

A read process of an RF-UCard system consists of multiple continuous read cycles. A read cycle starts at the time that the reader broadcasts a request command, and ends up when the last identified card in the current frame has been processed. The length of a cycle is equal to the current frame size plus the processing times. In order to pick the appropriate frame size N for the (a priori unknown) number of cards n in the read field, we have to estimate n in each read cycle. The estimation of card quantity is a key issue involved in dynamic framed ALOHA algorithms. In SDA, we employ the estimation scheme that studied extensively in most literatures and originally proposed by H. Vogt [13]. The estimation proceeds as following steps.

Step1: Based on the mathematical basis discussed previously (recall (4) and (5)), we can compute the

expected value $<c_0, c_1, c_x>$ with already known N and n. In a read cycle with the frame size N, the expected number of empty slots (with occupancy number 0) can be obtained by

$$a_0^{N,n} = NB_{n,\frac{1}{N}}(0) = N(1-\frac{1}{N})^n \quad \textbf{(6)}$$

Also, the expected number of success slots (with occupancy number 1) can be obtained by

$$a_1^{N,n} = NB_{n,\frac{1}{N}}(1) = N\binom{n}{1}(\frac{1}{N})(1-\frac{1}{N})^{n-1} = n(1-\frac{1}{N})^{n-1} \quad \textbf{(7)}$$

Thus, the expected number of collision slots (with occupancy number >1) is $N - a_0^{N,n} - a_1^{N,n}$. Figure 2 shows a function definition in Java to obtain the expected value of $<c_0, c_1, c_x>$ and Table 1 shows some useful expected values that derived from the function *getSlotCount* with given certain N and n. For an extensive experiment, we will compute more expected values of $< c_0, c_1, c_x >$ by more possible N and n. Then make an expected value table similar to Table 1.

```
void getSlotCount(int N, int n, double c0, double c1,
double cx)
{
    c0= N * Math.pow((1-(1.0/N)), n);
    c1 = n * Math.pow((1-(1.0/N)), (n-1));
    cx = N - c0 - c1;
}
```

Figure 2. The function to compute the expected values of $< c_0, c_1, c_x >$.

Step2: In each read cycle, the reader will get a read value of $<c_0, c_1, c_x>$. The Chebyshev's inequality tells us that the outcome of a random experiment involving a random variable X is most likely somewhere near the expected value of X. Thus we use the distance between the read value and the expected value of $<c_0, c_1, c_x>$ to estimate the number of cards n for which the distance becomes minimal. The estimation function denoted by ε_{vd} is defined as

$$\varepsilon_{vd}(N, c0, c1, cx) = \min_n \left\| \begin{pmatrix} a_0^{N,n} \\ a_1^{N,n} \\ a_{\geq 2}^{N,n} \end{pmatrix} - \begin{pmatrix} c0 \\ c1 \\ cx \end{pmatrix} \right\| \quad \textbf{(8)}$$

Because the current frame size N is always known, after getting a read value of $<c_0, c_1, c_x>$, we can compare the read value with the expected value table. According to (8), we can obtain the estimated card quantity n finally.

4.3. Adjusting the Frame Size

The variation of the frame size takes large impacts on the performance of dynamic framed ALOHA algorithms [14, 17]. Results revealed that the maximum performance occurs when the frame size equals the number of tags. However, this result is not suitable again to the collision resolution in an RF-UCard system. Recall that the card arrivals are a Poisson process, the card quantity within the read field is dynamically changed. The estimation scheme proposed in Section 4.2 just reflects the card quantity within the read field at the beginning of a read cycle while not including the new cards arrived in the current read cycle. Thus in SDA, we will employ a novel adjusting scheme in terms of the estimated card quantity and the new arriving cards to choose an optimal frame size for the next read cycle. Based on the card quantity estimation scheme and (3), the new frame size N^* can be obtained by

$$N^* = (n - c_1) + p \cdot \lambda N \quad \textbf{(9)}$$

where c_1 and p denote the number of identified cards in the current cycle and the initial response probability of cards respectively. Again, λN denotes the expected number of arriving cards in the current cycle (see(3)), and then $p \cdot \lambda N$ denotes the new arriving cards that really respond in the next cycle. In other words, (9) shows the idea that letting the new frame size N^* to be the number of the cards that will respond in the next cycle.

4.4. Adjusting the Response Probability

In BBEI, the tag response probability is changed within a range of the inverses of the binary exponential values. As we all know that a variable with an exponential increment shows the sharp deviations, especially when the variable becomes very large. Thus we limits the card response probability in SDA to be within a range of values with the linear increment, that is *ResProValues* ={1.0,7.0/8, 6.0/8,5.0/8,4.0/8,3.0/8,2.0/8,1.0/8}. We also assume that all cards have the same initial response probability p= *ResProValues* [k], e.g. if k =4, p=0.5, and this initial value is also held by the reader.

The adjusting scheme for the card response probability in SDA is based on the previous read results (i.e. the traffic intensity f_T of the system in the previous read cycle) and can be defined as a Java function shown in Figure 3.

```
double getResPro(int k, int fT)
{
    if(fT ==-1)
        k--;
    else if(fT ==1)
        k++;
    if(k < 0)
        k = 0;
    else if(k >= ResProValues.length)
        k = ResProValues.length - 1;
    return ResProValues[k];
}
```

Figure 3. The function to compute the card response probability.

4.5. Procedure of SDA Algorithm

Table 1. Some typical expected values of $<c_0, c_1, c_x>$ by given certain N and n.

Frame size	5 cards			10 cards			15 cards		
	*c*0	*c*1	*cx*	*c*0	*c*1	*cx*	*c*0	*c*1	*cx*
6	2.4113	2.4113	1.1774	0.969	1.9381	3.0929	0.3894	1.1683	4.4423
7	3.2387	2.6989	1.0624	1.4984	2.4973	3.0043	0.6933	1.7332	4.5735
8	4.1033	2.9309	0.9658	2.1046	3.0066	2.8888	1.0795	2.3132	4.6073
9	4.9944	3.1215	0.8841	2.7715	3.4644	2.7641	1.538	2.8837	4.5783
10	5.9049	3.2805	0.8146	3.4868	3.8742	2.639	2.0589	3.4315	4.5096
Frame size	20 cards			25 cards			30 cards		
	*c*0	*c*1	*cx*	*c*0	*c*1	*cx*	*c*0	*c*1	*cx*
6	0.1565	0.626	5.2175	0.0629	0.3145	5.6226	0.0253	0.1517	5.823
7	0.3207	1.0692	5.6101	0.1484	0.6183	6.2333	0.0687	0.3433	6.588
8	0.5537	1.5819	5.8644	0.284	1.0142	6.7018	0.1457	0.6242	7.2301
9	0.8535	2.1337	6.0128	0.4736	1.4801	7.0463	0.2628	0.9856	7.7516
10	1.2158	2.7017	6.0825	0.7179	1.9942	7.2879	0.4239	1.413	8.1631

A read cycle in SDA proceeds as following five steps.

Step1: The reader initiates a read cycle by broadcasting *Req-COS*(*CID*, *N*, f_T) to all cards within the reader's read field. Because a read process begins with the reader broadcasts the first request command (the time is denoted by 0), no card has arrived at that time and the first read cycle is always wasted. Thus it is reasonable to set the initial frame size to 1 in SDA. Moreover, the initial value of f_T is often set to 0.

Step2: After receiving *Req-COS* (*CID*, *N*, f_T), all cards within the read field perform *Valid-check* (*CID*) to validate itself to the reader with the reader *CID* derived from *Req-COS* (*CID*, *N*, f_T). If a card is invalid, it will set itself to the state *invalid* (f_C =−1) and then exits the following cycles permanently. Otherwise the cycle proceeds to the next step.

Step3: The valid cards first perform *Slot-select* (*N*) to generate a random number *s* uniformly distributed within the range from 0 to *N*-1. Then all valid cards adjust their response probability *p* by performing *getResPro* (*k*, f_T) based on the value of f_T received from the reader. Finally, the cards send back *Res* (*UID*) at the *s*th slot with the newly probability *p*.

Step4: The reader checks the slot states in the current frame in sequence. If a slot is successful, a card is identified and its *UID* is appended to S_{UID} for the later processes. After state checking, the reader can observe c_0 empty slots, c_1 successful slots, and c_x collision slots, where $c_0+c_1+c_x = N$.

Step5: The reader sequential takes the *UID* from S_{UID} as a parameter to send *Req-app* (*UID, AID*) to execute the specific application with the identified card. This step is also called the card processing step. After finishing the process, the card sets itself to the state *sleep* (f_C = 0) so that it cannot respond in the following cycles. The read cycle ends up with the finish of processing of all identified cards.

After finishing a read cycle, SDA will perform a synchronous adjusting scheme to optimize the frame size and the card response probability for the next read cycle. The adjusting proceeds as following two steps.

Step1: The reader first performs the card quantity estimation proposed in Section 4.2 to obtain an estimated card number *n*. Then, the adjusting scheme proposed in Section 4.3 is performed to choose an optimal frame size for the next read cycle.

Step2: The reader also adjusts the traffic intensity flag f_T according to the relationship between c_0 and c_x. Note that a collision slot means it is occupied by at least two cards, thus we develop an adjusting scheme for f_T as: if $c_0 > 2\cdot c_x$, then f_T=−1; if $c_0<2\cdot c_x$, then f_T = 1; else f_T=0. This step aims to refresh the value f_T to inform the card to change its response probability in the next cycle.

An example provided in Figure 4 illustrates the read process of SDA. In this example, we set the card arrival rate to 2 and the card response probability to 0.5. We

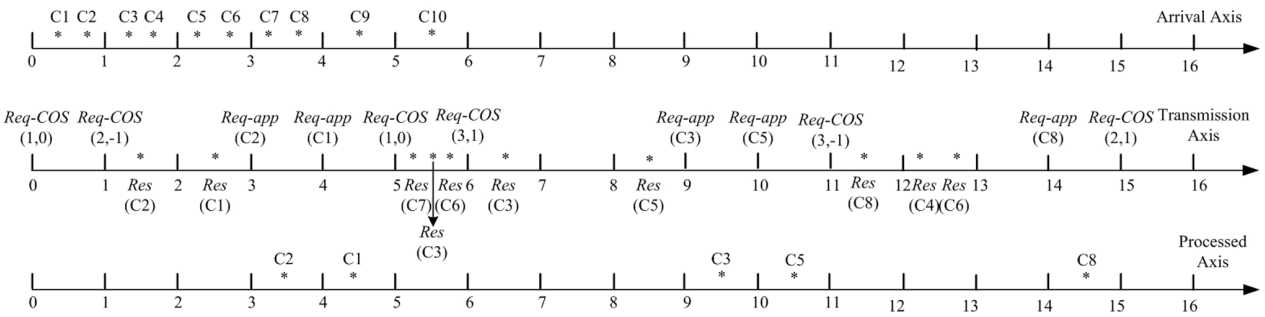


Figure 4. An example of the SDA algorithm.

assume that all arriving cards are valid and execute the same application. Thus the parameters *CID* and *AID* can be omitted in *Req-COS*() and *Req-app*() respectively. There are three time axes: the arrival axis, the transmission axis, and the processed axis. Each time axis is divided into equal slots with fixed length. The arrival axis shows the arrival time of card C1, C2, …, C10 denoted by the symbol "*". A sequence of *Req-COS*(N, f_T) and *Req-app* (*UID*) are shown above the transmission axis (note that all commands are sent at the beginning of a slot), while a sequence of *Res* (*UID*) are shown below the transmission axis (denoted by the symbol "*"). The successful processed cards are shown above the processed axis (also denoted by the symbol "*"). At t = 0, the read process begins and the reader broadcasts *Req-COS*(1, 0). However, no card arrives at this time and the reader obtains a triple, <1, 0, 0>. According to the adjusting schemes for the frame size and the response probability in SDA, the newly parameters N=2, f_T=-1 and p =0.625 are derived. The reader initiates the 2nd cycle at t =1 with *Req-COS* (2, −1). At this time, C1 and C2 have arrived and respond in slot 3 and slot 2 respectively. Thus C2 and C1 are identified and then processed in slot 4 and slot 5 sequentially. Again the reader obtains a triple, <0, 2, 0 > and the newly parameters N =1, f_T =0, and p =0.625 for the 3rd cycle. In the meanwhile of the 2nd cycle, there are other 7 new arriving cards. The read process will continue in this way and the read data derived from the former 5 cycles are shown in Table 2.

5. Simulation and Evaluation

We develop a Java program based on the Eclipse platform to simulate the process of SDA. Our simulations are based on the following scenarios.

- All arriving cards are valid, which is the worst case may be occurred in a real-world RF-UCard system.
- The card arrival follows a Poisson process, and the arrival rate varies from 0.1 to 0.9 with a step of 0.1 and from 1 to 10 with a step of 1.
- The card quantity is finite that enables the simulation to be finished normally and the simulated card set varies from 10 to 200 cards with a step of 10.
- All experiments are simulated 100 times in order to ensure the convergence of simulation results.

5.1. Performance Measures

To evaluate the performance of SDA and other ALOHA-based anti-collision algorithms, we mainly consider the following measures.

- Total read time: this metric is the total time required to identify and process all the cards inside the reader's read field. We measure the time by the timeslot because each of three mentioned algorithms (SDA, DFSA and BBEI) has a time period for carrying both the reader-to-card signals and the card-to-reader signals.
- Card delay time: this metric is the average number of timeslots waited by a card in the entire read process. It also reflects the mean sojourn time of each card in the system before being processed.
- Identification efficiency: this metric is the mean number of cards being identified in a timeslot. It equals to the ratio of the total identified cards to the sum of the frame size.
- The number of collisions: this metric is the total number of collision timeslots between the card-to-reader responses. Collisions increase the read time and thus lower the identification efficiency.
- The number of empty timeslots: this metric is equals to the sum of the empty timeslots in each read cycle. More empty timeslots waste more read time and thus also lower the identification efficiency.

5.2. Card Arrival Simulation

We first simulate the card arrivals which follow a Poisson process while varying the arrival rate λ. The theoretical values of the interarrival times for each λ can be obtained from (1). We take 100 cards into considerations to compute the simulated values of the interarrival times. Let T_λ and S_λ denote the theoretical value and the simulated value of the interarrival times respectively. The relative error between S_λ and T_λ denoted by ε_λ can be derived from

$$\varepsilon_\lambda = \left(S_\lambda - T_\lambda\right) / T_\lambda \quad \textbf{(10)}$$

Table 3 shows the simulated results when λ varying from 0.2 to 6. The results present that the relative errors are rather little for all λ values, in other words, the theoretical values and the simulated values are in good agreement.

5.3. Performance Evaluation

We then evaluate the impact of the system parameters on the performance of SDA. Three parameters, i.e. the card arrival rate, the card initial response probability and the card quantity, are considered. Although the variation of each of these three parameters will influence the performance, we mainly focus on the impact of the single parameter. Thus we conduct the independent experiments for each parameter under the different scenarios.

Table 2. The read data derived from the example.

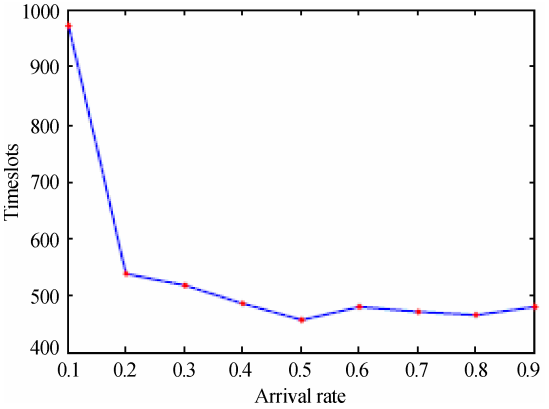
Cycle	1	2	3	4	5
N	1	2	1	3	3
f_T	0	-1	0	1	-1
p	0.5	0.625	0.625	0.5	0.625
c_0	1	0	0	1	1
c_1	0	2	0	2	1
c_x	0	0	1	0	1
S_{UID}	{}	{C2,C1}	{}	{C3, C5}	{C8}

- For the card arrival rate, we set the card quantity n to 100 and the initial response probability p to 0.5.
- For the card initial response probability, we set the card quantity n to 100 and the arrival rate λ to 1.
- For the card quantity, we set the arrival rate λ to 1 and the initial response probability p to 0.5.

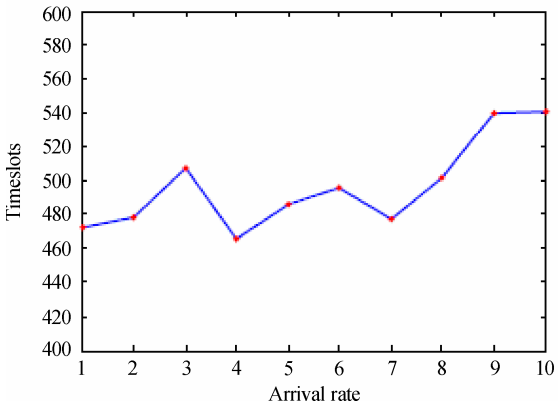
Figure 5 shows the simulation results about the impact of the card arrival rate λ. From Figure 5(a) and (b), we can see that fewer read timeslots required by SDA to complete a read process when λ varies within the range from 0.5 to 2, and the minimal value occurs at λ=0.5. When λ is less than 0.5, more timeslots required,

Table 3. The mean interarrival times of card arrivals (n =100).

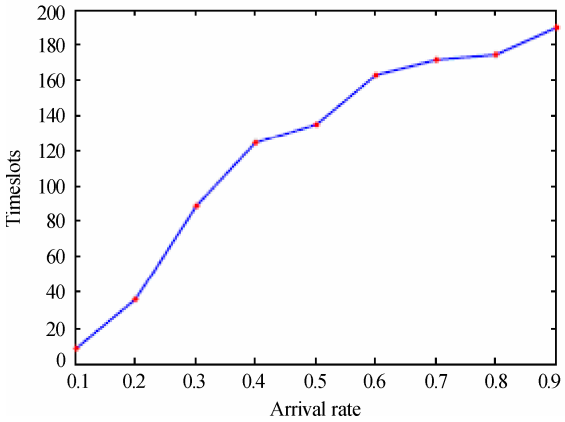
λ	T_λ	S_λ	ε_λ
0.2	5	5.0436	0.0087
0.4	2.5	2.5052	0.0021
0.6	1.6667	1.6843	0.0106
0.8	1.25	1.2614	0.0091
1	1	0.9878	-0.0122
2	0.5	0.4984	-0.0032
3	0.3333	0.3343	0.003
4	0.25	0.2534	0.0136
5	0.2	0.1974	-0.013
6	0.1667	0.1641	-0.0156



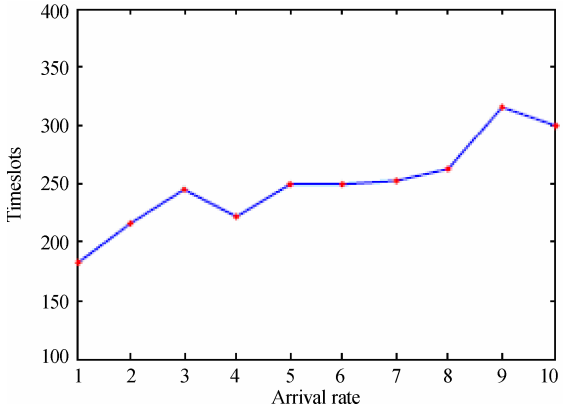
(a) Total read time (λ < 1, p = 0.5, n = 100)



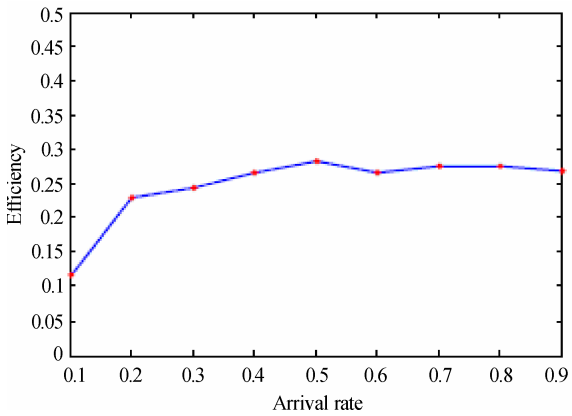
(b) Total read time (λ ≥ 1, p = 0.5, n = 100)



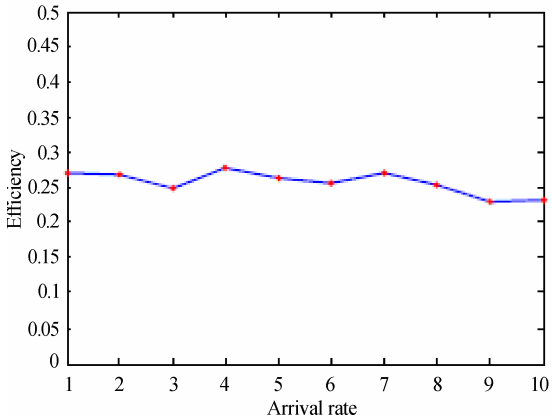
(c) Card delay time (λ < 1, p = 0.5, n = 100)



(d) Card delay time (λ ≥ 1, p = 0.5, n = 100)



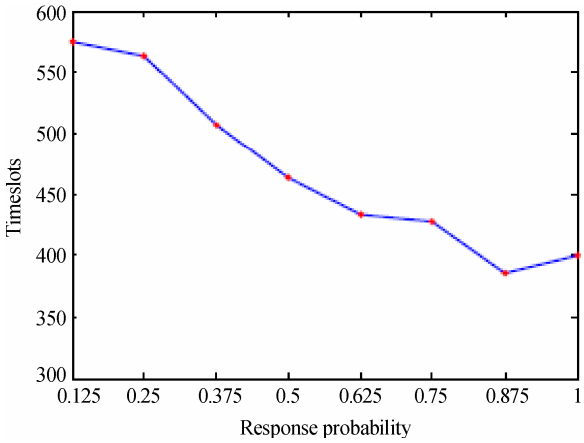
(e) Identification efficiency (λ < 1, p = 0.5, n = 100)



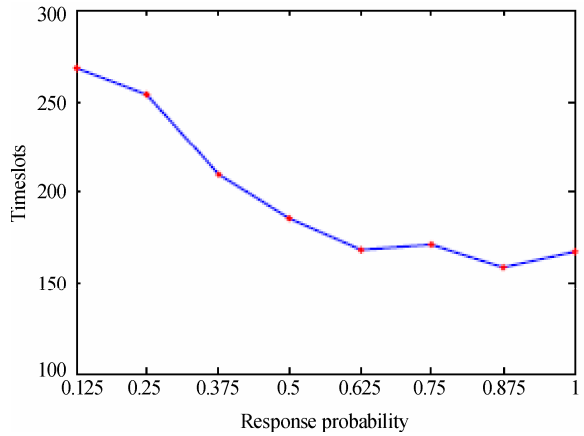
(f) Identification efficiency (λ ≥ 1, p = 0.5, n = 100)

Figure 5. Impact of card arrival rate.

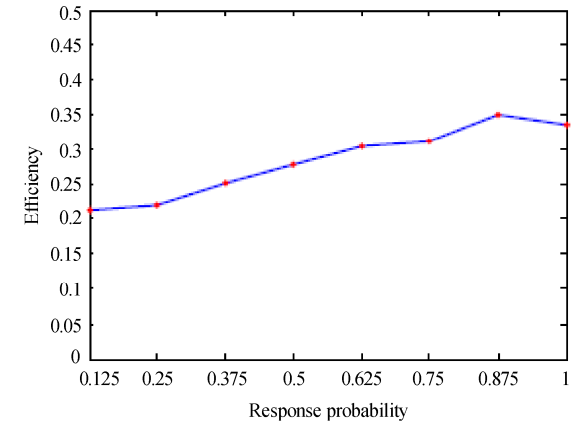
especially the maximal value occurs at λ=0.1. This is mainly due to the fact that the smaller arrival rate means the lager interarrival time and thus increases the number of empty timeslots in each read cycle. On the other hand, as λ increases, the read process also becomes longer because of the larger arrival rate will enlarge the card quantity rapidly and thus increases card collisions in each read cycle. Figure (c) and (d) illustrate that the card delay time gets longer as λ increases. This is due to the fact that the larger arrival rate means the earlier arrival time for each card, while the longer read time further extends the card delay time. The impact of the arrival rate on identification efficiency shown in Figure 5(e) and (f) is very similar to the case of the read time shown in Figure 5(a) and (b). SDA achieves better identification efficiency when λ varies within the range from 0.5 to 2, and the maximal efficiency occurs at λ=0.5. As λ has larger distance from this range, the lower efficiency occurs. However, the



(a) Total read time (λ = 1, *n* = 100)

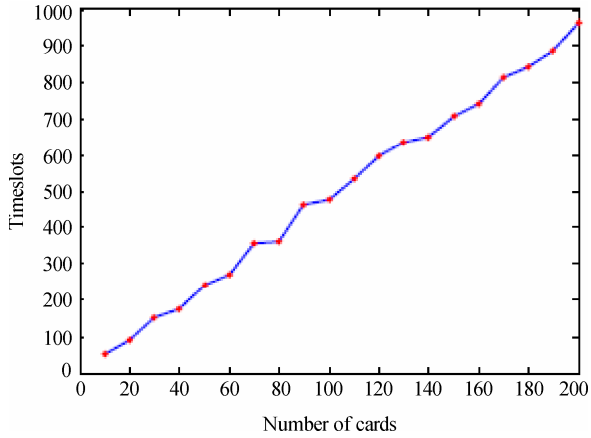


(b) Card delay time (λ = 1, *n* = 100)

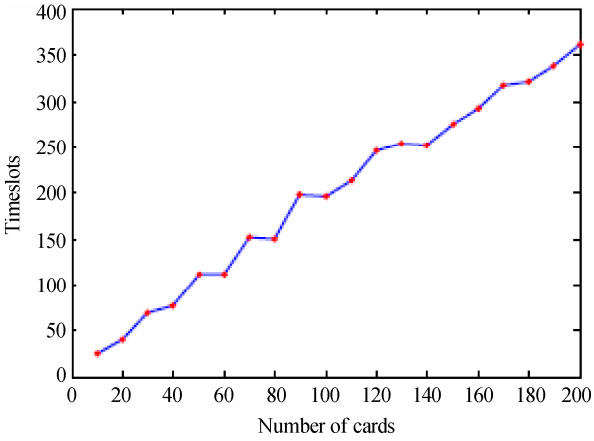


(c) Identification efficiency (λ = 1, *n* = 100)

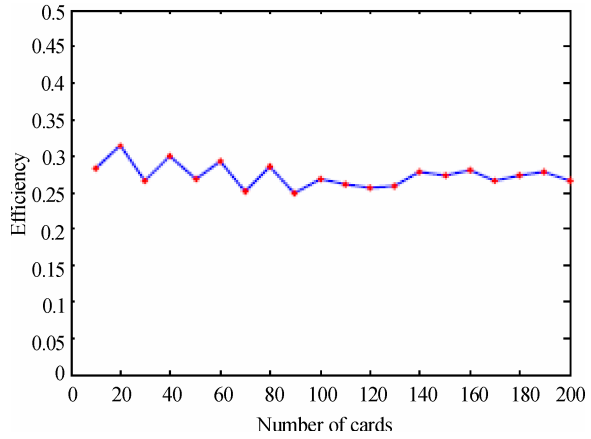
Figure 6. Impact of card initial response probability.



(a) Total read time (λ = 1, *p* = 0.5)



(b) Card delay time (λ = 1, *p* = 0.5)



(c) Identification efficiency (λ = 1, *p* = 0.5)

Figure 7. Impact of card quantity.

deviation of the identification efficiency is rather small for all λ values except 0.1, and the efficiency varies around 0.26.

Figure 6 shows the simulation results about the impact of the card initial response probability *p*. From Figure 6(a) and (b), we can see that both the total read time and the card delay time get shorter as *p* increases, especially the shortest time occurs at p = 0.875. Also, Figure 6(c) presents that SDA shows the better identification efficiency as *p* increases. The optimal efficiency achieved when p = 0.875 and is close to 0.35. These results are due to the fact that the larger response probability will decrease the number of empty timeslots rapidly as well as not increase card collisions obviously in each read cycle when the arrival rate is 1.0.

Figure 7 shows the simulation results about the impact of the card quantity *n*. From Figure 7(a) and (b), we can see that both the read time and the card delay time increase linearly with *n*. However, the read time grows by the larger incremental ratio than the delay time that the former is more than 4.5 times the card quantity while the latter is close to 2 times the card quantity. For example, if there are 100 cards, the required total read time and card delay time are about 450 and 200 timeslots respectively.

5.4. Performance Comparisons

In order to compare the performance of SDA to DFSA and BBEI, we further develop Java programs to simulate DFSA and BBEI in an RF-UCard system. For DFSA, we use the estimation function ε_{vd} which defined in (8) to estimate the card quantity in the current read cycle. In addition, we set the frame size to be the estimated card quantity to obtain an optimal frame size for the next read cycle. For each algorithm, we set the card quantity to 100 while varying the card arrival rate, whereas we set the arrival rate to 1 while varying the card quantity. In order to carry out an extensive comparison, we take the optimal performance achieved by three algorithms into comparisons under a given simulation scenario.

Table 4 and Figure 8 depict the simulation results while varying the card arrival rate. From Table 4, we find that for each algorithm, collision timeslots are always less than empty timeslots in a read process. However, SDA generates the minimal collision timeslots

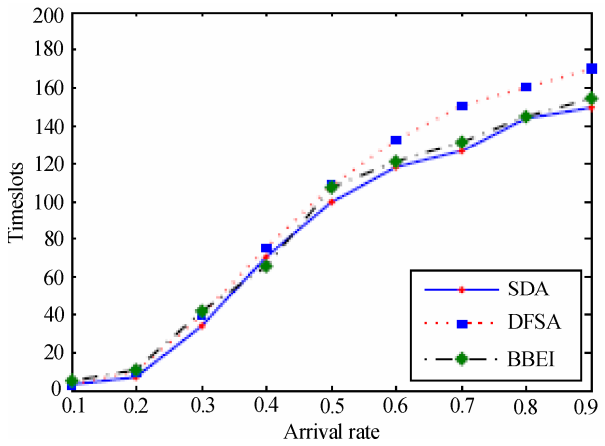
Table 4. The average number of timeslots with varying the arrival rate (*n* =100).

Arrival rate	Algorithm	Collision timeslots	Empty timeslots	Total timeslots
0.2	SDA	41.3	264.1	505.4
	DFSA	55.1	228.8	483.9
	BBEI	50.6	211	461.6
0.4	SDA	72.7	111.5	384.2
	DFSA	75.5	117.2	392.7
	BBEI	81.9	108.2	390.1
0.6	SDA	71.8	123.8	395.6
	DFSA	78.4	129.7	408.1
	BBEI	97.8	102.7	400.5
0.8	SDA	66	103.8	369.8
	DFSA	77.3	129.1	406.4
	BBEI	92.2	102.5	394.7
1	SDA	71.5	120.9	392.4
	DFSA	75.8	141	416.8
	BBEI	98.7	105.5	404.2
2	SDA	69.1	126	395.1
	DFSA	74.2	147.6	421.8
	BBEI	97.7	107.8	405.5
3	SDA	65.2	147.2	412.4
	DFSA	76.2	150.5	426.7
	BBEI	98	118.4	416.4
4	SDA	69.6	125.4	395
	DFSA	78.2	151.1	429.3
	BBEI	98.1	109.8	407.9
5	SDA	68.3	131.4	399.7
	DFSA	78.4	145.5	423.9
	BBEI	99.7	101.4	401.1
6	SDA	70	155.4	425.4
	DFSA	76.9	165.4	442.3
	BBEI	89.4	144.8	434.2

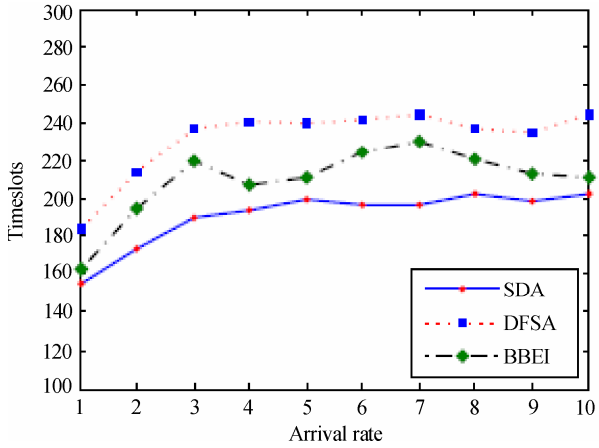
Table 5. The average number of timeslots with varying the card quantity (λ =1).

Card quantity	Algorithm	Collision timeslots	Empty timeslots	Total timeslots
20	SDA	13.7	25.5	79.2
	DFSA	12.9	31.4	84.3
	BBEI	15.3	18.8	74.1
40	SDA	26.2	53.3	159.5
	DFSA	30	58	168
	BBEI	37.7	37.6	155.3
60	SDA	42.6	76.4	239
	DFSA	46.5	80.8	247.3
	BBEI	55.5	63.5	239
80	SDA	55	97.1	312.1
	DFSA	59.3	111	330.3
	BBEI	72	92.2	324.2
100	SDA	71.5	120.9	392.4
	DFSA	75.8	141	416.8
	BBEI	98.7	105.5	404.2
120	SDA	92	142.7	474.7
	DFSA	102	170.8	512.8
	BBEI	124.9	127.2	492.1
140	SDA	107.3	169	556.3
	DFSA	112.5	182.8	575.3
	BBEI	129	153.7	562.7
160	SDA	123.4	213.2	656.6
	DFSA	126.9	219.7	666.6
	BBEI	169.7	173	662.7
180	SDA	130.4	219.6	710
	DFSA	145.3	229	734.3
	BBEI	165.4	187	712.4
200	SDA	148.7	229.7	778.4
	DFSA	159.3	263.2	822.5
	BBEI	198.5	213	811.5

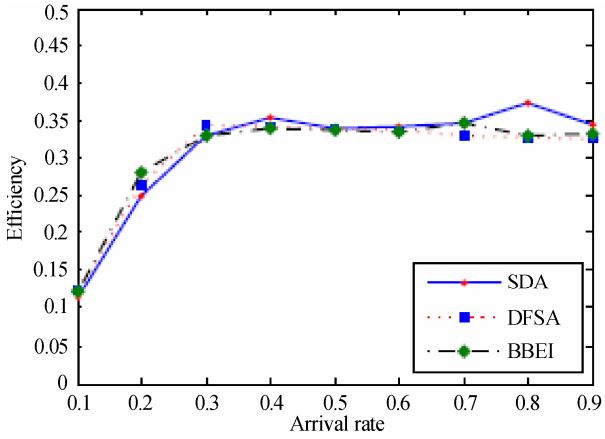
in all algorithms because of the synchronous dynamic adjusting scheme we employed for both the frame size and the card response probability. Due to the frame size is always set to 1 during the entire read process, BBEI generates the maximal collision timeslots as well as the minimal empty timeslots in all algorithms. SDA generates fewer empty timeslots than DFSA since the new arriving cards in each read cycle are taken into consideration to compute the new frame size. As a result, SDA requires fewer total timeslots than DFSA and BBEI to complete a read process, Figure 8(a) and (b) illustrate that the card delay time of SDA is the shortest of all algorithms, and the gap of delay time between SDA and other algorithms is much larger when the arrival rate is greater than 1. Figure 8(c) and (d) illustrate that the identification efficiency of SDA is



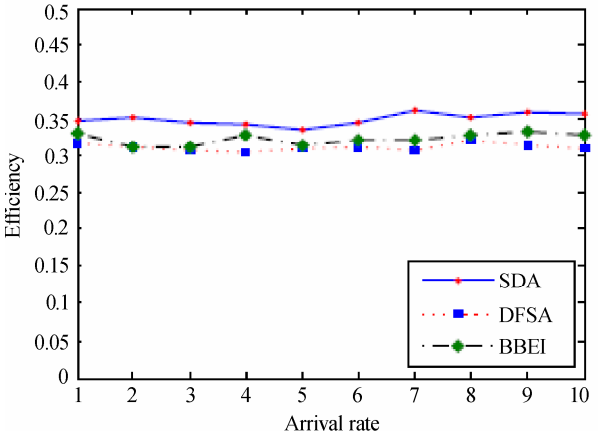
(a) Card delay time ($\lambda < 1$, $n = 100$)



(b) Card delay time ($\lambda \geq 1$, $n = 100$)

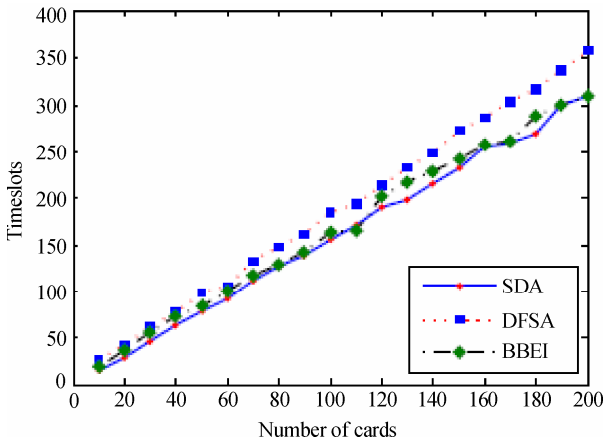


(c) Identification efficiency ($\lambda < 1$, $n = 100$)

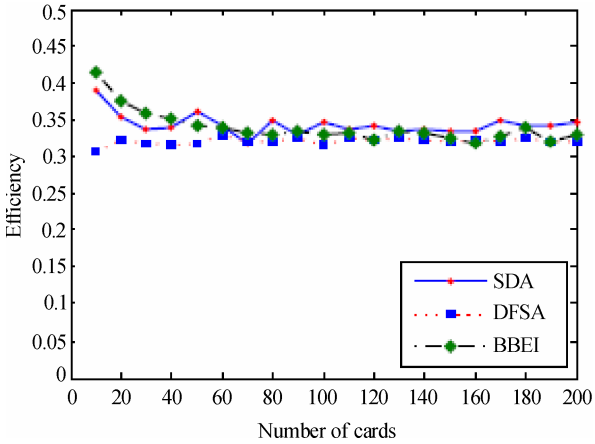


(d) Identification efficiency ($\lambda \geq 1$, $n = 100$)

Figure 8. Performance comparison with varying the card arrival rate.



(a) Card delay time



(b) Identification efficiency

Figure 9. Performance comparison with varying the card quantity.

the best of all algorithms at high arrival rate, while it is worse than DFSA and BBEI at arrival rate less than 0.3 because of more empty timeslots produced by larger interarrival time. Note that BBEI shows shorter delay time and better identification efficiency than DFSA at most simulation scenarios. These results show a good agreement with the results shown in Table 4 because fewer total timeslots mean better identification efficiency.

Table 5 and Figure 9 show the simulation results obtained by varying the card quantity. As the card quantity increases, the total read time and the card delay time get longer and the number of collision timeslots and empty timeslots gets larger for each algorithm. The results shown in Table 5 are very similar to the results shown in Table 4. SDA generates the minimal collision timeslots when the card quantity is greater than 40. Again, BBEI generates the maximal collision timeslots as well as the minimal empty timeslots. The total timeslots required by SDA is fewer than DFSA and BBEI when the card quantity is greater than 60. By restraining the occurrence of collisions, SDA has shorter delay time than DFSA and the gap gets larger as the card quantity increases. However, the delay time of BBEI is very close to the time of SDA because it generates fewer empty timeslots. All of algorithms show very similar identification efficiency when there are larger cards in the read field. SDA shows a little better efficiency than others when the card quantity is greater than 60. The values vary around 0.35, which is very close to 0.368, the maximal identification efficiency of the framed ALOHA applied in RFID systems.

6. Conclusions

Multi-card collision is a major factor in influencing the efficiency of an RF-UCard system. In this paper, a novel and enhanced multi-card anti-collision algorithm has been proposed and evaluated. Unlike DFSA and BBEI that they merely adjust the frame size or the tag response probability, the proposed SDA algorithm employs a synchronous dynamic adjusting scheme that dynamically adjusts the frame size in the reader and the response probability in cards to maximize the efficiency of card identification and processing. For the reader, the estimated card quantity and the new arriving cards in the current read cycle are both taken into consideration to adjust the frame size for the next read cycle. For a card, it increases or decreases its response probability according to the request commands sent from the reader. This scheme improves the card identification efficiency by reducing both the collision timeslots and the empty timeslots.

The simulation results indicate that the performance of SDA is seriously affected by the variations of the card arrival rate, the card initial response probability and the card quantity. SDA requires shorter read time and achieves better identification efficiency when the arrival rate varies within the range from 0.5 to 2. The card delay time gets longer as the arrival rate increases while gets shorter as the card initial response probability increases. Moreover, both the read time and the card delay time increase linearly with the card quantity. A simulation based comparison shows that SDA requires shorter read time and card delay time and achieves better identification efficiency than DFSA and BBEI by significantly reducing the collision timeslots and the empty timeslots. The optimal identification efficiency of SDA varies around 0.35, which is very close to 0.368, the maximal identification efficiency of the framed ALOHA applied in RFID systems.

7. References

[1] K. Finkenzeller, “RFID handbook,” 2nd edition, John Wiley and Sons, 2003.

[2] W. Rankl and W. Effing, “Smart card handbook,” 3rd edition, John Willey and Sons, 2004.

[3] L. Shu, J. C. Cao, and Z. D. Lu, “CIM: Hardware support for Multi-COS isolation of RF-UCard,” in Proceedings 2008 International Conference on Embedded Software and Systems, pp. 595-602, July 2008.

[4] D. R. Hush and C. Wood, “Analysis of tree algorithm for RFID arbitration,” in Proceedings IEEE International Symposium on Information Theory, pp. 107, 1998.

[5] ISO/IEC 18000-6: Information technology-Radio frequency identification for item management, Part 6, Parameters for air interface communications at 860 MHz to 960 MHz, 2004/Amd 1: 2006.

[6] J. Myung and W. Lee, “Adaptive binary splitting: A RFID tag collision arbitration protocol for tag identification,” Mobile Networks and Applications, Vol. 11, No. 5, pp. 711−722, May 2006.

[7] C. Law, K. Lee, and K. S. Siu, “Efficient memoryless protocol for tag identification,” in Proceedings 4th ACM International workshop on discrete algorithms and methods for mobile computing and communications, pp.75−84, August 2000.

[8] J. Myung and W. Lee, “Adaptive splitting protocols for RFID tag collision arbitration,” in Proceedings 7th ACM international symposium on Mobile ad hoc networking and computing, pp. 202−213, 2006.

[9] N. Abramson, “Development of the ALOHANET,” IEEE Transactions on Information Theory, Vol. IT-31, pp. 119−123, March 1985.

[10] L. G. Roberts, “ALOHA packet system with and without slots and capture,” ARPA Network Information Center, Stanford Institute, Menlo Park California, ASS Note 8 (NIC 11290), June 1972.

[11] F. C. Schoute, “Control of ALOHA signalling in a mobile radio trunking system,” in Proceedings International Conference on Radio Spectrum Conservation Techniques, pp. 38−42, 1980.

[12] F. C. Schoute, “Dynamic frame length ALOHA,” IEEE Transactions on Communications, Vol. 31, No. 4, pp. 565−568, April 1983.

[13] H. Vogt, “Efficient object identification with passive

RFID tags," in Proceedings International Conference on Pervasive Computing, Springer-Verlag, Vol. 2414, pp. 98−113, 2002.

[14] W. T. Chen and G. H. Lin, "An efficient anti-collision method for tag identification in a RFID system," IEICE Transaction on Communications, Vol. E89-B, No.12, pp. 3386−3392, December 2006.

[15] J. Zhai and G. N. Wang, "An anti-collision algorithm using two-functioned estimation for RFID tags," in Proceedings International Conference Computational Science and its Applications, Vol. 3843, pp. 702−711, May 2005.

[16] S. R. Lee, S. D. Joo, and C. W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in Proceedings 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 166−172, July 2005.

[17] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for multiple RFID objects identification," IEICE Transaction on Communications, Vol. E88-B, No. 3, pp. 991−999, March 2005.

[18] S. S. Yu, Y. Zhan, and Y. H. Wang, "RFID anti-collision algorithm based on bi-directional binary exponential index," in Proceedings the IEEE International Conference on Automation and Logistics, Jinan, China, pp. 2917–2921, August 2007.

[19] S. A. Tanenbaum, "Computer Networks," 4th edition, Prentice Hall PTR, 2002.

[20] R. Motwani and P. Raghavan, "Randomized algorithms," Cambridge University Press, 1995.

