Binary Tree Slotted ALOHA for Passive RFID Tag Anticollision

Haifeng Wu, Yu Zeng, Jihua Feng, and Yu Gu

Abstract—In order to enhance the efficiency of radio frequency identification (RFID) and lower system computational complexity, this paper proposes three novel tag anticollision protocols for passive RFID systems. The three proposed protocols are based on a binary tree slotted ALOHA (BTSA) algorithm. In BTSA, tags are randomly assigned to slots of a frame and if some tags collide in a slot, the collided tags in the slot will be resolved by binary tree splitting while the other tags in the subsequent slots will wait. The three protocols utilize a dynamic, an adaptive, and a splitting method to adjust the frame length to a value close to the number of tags, respectively. For BTSA, the identification efficiency can achieve an optimal value only when the frame length is close to the number of tags. Therefore, the proposed protocols efficiency is close to the optimal value. The advantages of the protocols are that, they do not need the estimation of the number of tags, and their efficiency is not affected by the variance of the number of tags. Computer simulation results show that splitting BTSA's efficiency can achieve 0.425, and the other two protocols efficiencies are about 0.40. Also, the results show that the protocols efficiency curves are nearly horizontal when the number of tags increases from 20 to 4,000.

Index Terms—RFID, anticollision, ALOHA, estimation of the number of tags, passive

1 Introduction

Internet of Things (IoT) is a vision where all objects can be uniquely identified and connected through a wireless or wired communication network [1]. The purpose of IoT is to link real-world objects with virtual information, to identify, track, and manage all goods available, and then to set up a global net among product, customer, company, enterprise, and government. Radio Frequency Identification (RFID) is a technical keystone of IoT since small passive RFID tags make it possible to link millions and billions of physical products with virtual information. When millions and billions of tags are used, it is probable that there will be more than one tag within the interrogatory zone of a reader at some time. When the tags transmit their signals simultaneously to the reader, collisions will happen because the reader identifies the tags through communication over a shared wireless channel. Therefore, RFID tag anticollision algorithms will play an important role in IoT.

In the algorithms for passive RFID tag anticollision, a binary tree slotted ALOHA (BTSA) algorithm is not new. The prototype of BTSA algorithm is proposed by Capetanakis [14], who applied BTSA to random multiaccess communication systems. Several RFID tag anticollision protocols, such as Modified Q [21], adaptive binary splitting (ABS) [22] and framed-slotted ALOHA with robust estimation and binary selection (EB-FSA) [23] actually adopt the idea of BTSA algorithm. BTSA algorithm initializes a frame constituting a number of slots and randomly assigns tags to these slots. If some tags collide in a slot, the collided tags in the slot will be resolved by binary tree (BinTree) splitting at once while the other tags in the subsequent slots will wait. When an initial frame length, i.e., the slot number in an initial frame is equal to the number of tags, BTSA can achieve a maximum efficiency value of about 0.43 [14], which is higher than the efficiency of dynamic frame slotted ALOHA (Dynamic FSA) [6], [7], [8], [9], [10] protocol and pure binary tree protocol [15], and nearly equal to the ideal tree slotted ALOHA (TSA) protocol's [11], [12], [13] optimal efficiency. In order to obtain the maximum efficiency, the initial frame length should be set according to the number of tags which is, however, usually unknown to a reader. Therefore, the number of tags needs to be estimated. The protocols adopting BTSA algorithm, such as ABS do not conduct the estimation, which will degrade its efficiency when the number of tags increases or decreases much. Although EB-FSA and Modified Q conduct the estimation to achieve higher efficiency, they will bring about the disadvantages of estimation of the number of tags.

Generally, the estimation of the number of tags will have the following disadvantages. First, the estimation increases computational cost. Most of conventional accurate estimates, such as Vogt estimate [7], maximum a posteriori (MAP) estimate [9], Bayesian estimate [10] and estimates in [16], [17] need to search a maximum value in a range of the number of tags. If the range is large, the times of searching will be very large. This will result in terrible computational complexity. Although estimates in [24], [28], [29] have lower cost, the estimates require tags to apply several hash functions and thus increase the cost of tags. Second, the estimation error degrades the efficiency performance. Even accurate estimates have to introduce estimation error. For example, when the number of tags is much larger than the frame length, the number of idle slot and successful slot in

• *The authors are with the School of Electrical and Information Technology, Yunnan University of Nationalities, Kunming 650500, China. E-mail: {whf5469, yo.zeng}@gmail.com, fengjihua@21cn.com, maggie.guyu@hotmail.com.*

Manuscript received 1 Dec. 2011; revised 28 Jan. 2012; accepted 16 Mar. 2012; published online 28 Mar. 2012.
Recommended for acceptance by E. Li.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-12-0875.
Digital Object Identifier no. 10.1109/TPDS.2012.120.

the frame may be zero. In this case, Vogt, MAP, and Bayesian estimate cannot estimate the number correctly. Thus, a frame length's adjustment according to the incorrectly estimated results will degrade the efficiency performance. Third, the variance of the number of tags will result in the variance of efficiency. In general, the estimation of the number of tags requires observation results, which can be collected by a reader after completing a frame, such as idle slot, collision slot, and successful slot quantity in a frame. For this reason, an initial frame length cannot be set according to the estimated number of tags, but be set to a fixed value beforehand because the reader cannot collect the observation results before completing the initial frame. If the number of tags suddenly increases or decreases much, the initial frame length may be much larger or less than the number of tags. This will lead to underutilization of channel and low efficiency.

In this paper, we utilize BTSA algorithm to propose three passive RFID tag anticollision protocols, which are dynamic BTSA protocol, adaptive BTSA protocol, and splitting BTSA protocol, respectively. Since all of the proposed protocols can adjust the frame length to a value close to the number of tags, their identification efficiency can achieve a value close to an optimal one. From computer simulations results, splitting BTSA's efficiency can achieve 0.425, and the other two protocols efficiency is about 0.40. The advantages of the three propose protocols are to require no estimation of the number of tags, and their efficiencies are not affected by the variance of the number of tags. Their efficiency is not only higher than the existing protocols without estimation, but also close to and even higher than the TSA-based protocols with estimation [11], [12], [13].

Our contributions are summarized as follows:

- We propose a novel protocol to adjust an initial frame length, called dynamic BTSA. The protocol can obtain the frame length close to the number of tags by judging only the first slot type.
- We make an improvement on Q algorithm, i.e., adaptive BTSA. The improvement advances the identification efficiency from 0.34 to 0.40.
- We prove that TSA will not have higher identification efficiency than binary tree under a condition that an initial frame length is equal to the number of tags. Since splitting BTSA makes the condition guaranteed, a reader can adopt binary tree to resolve collided tags, instead of TSA.
- We reduce the RFID system's computational cost. Since the proposed protocols do not need to estimate the number of tags and avoid the computational cost of the estimation, the RFID system has lower computational cost for running the proposed protocols on hardware of a reader.

The rest of this paper is organized as follows. Section 2 contains the related work. We describe problems of BTSA algorithm in Section 3. Section 4 proposes three BTSA protocols and Section 5 analyzes the BTSA protocols performance. In Section 6, we provide computer simulations to demonstrate the performance of the proposed protocols. Finally, conclusions are drawn in Section 7.

2 Related Work

In conventional protocols for passive RFID tag anticollision, ALOHA-based protocols are very popular. The protocols utilizing BTSA algorithm is also ALOHA-based protocol. Now, 13.56 MHz ISM band EPC Class 1 [2], ISO 18000-6 Type A [3], Type C [4], and EPCglobal Generation 2 (EPC Gen 2) [5] all use ALOHA-based protocols. The idea of ALOHA-based protocols is to divide access time of tags into a number of slots, and each tag responds at a random slot. If tags collide in a slot, which means that at least two tags responses in the slot. The collided tags need to randomly reselect slots. In ALOHA-based protocols, identification efficiency is related to the number of tags and slots. If the number of slots is much less than that of the tags, collision probability will increase. On the other hand, if the number of slots is much larger than that of tags, many idle slots may be produced. These will both decrease the identification efficiency. Hence, the slot number needs to be adjusted according to the number of tags which is, however, usually unknown to a reader. Therefore, from whether to estimate the number of tags or not, ALOHA-based protocols can be categorized into protocols with estimation and protocols without estimation.

In ALOHA-based protocols with estimation, dynamic frame slotted ALOHA [6], [7], [8], [9], [10] is widely applied. Dynamic FSA configures an identification process with some continuous frames consisting of slots, and dynamically adjusts a frame length. Compared with fixed framed slotted ALOHA, Dynamic FSA can achieve a higher efficiency value of 0.37. To improve the identification efficiency, someone integrates tree algorithms into ALOHA-based protocols, and proposes several hybrid protocols: TSA protocol [11], dynamic TSA protocol (DyTSA) [12], binary splitting TSA (BSTSA) protocol [13], adaptive splitting-based arbitration protocol (ASAP) [24] and a hybrid protocol in [30], where tree algorithms resolve a collision by successively muting subsets of tags that are involved in the collision [14], [15]. The hybrid protocols can achieve higher efficiency than Dynamic FSA. Since the protocols above require the information about the number of tags and thus require the estimation, however, they have to introduce the disadvantages of estimation mentioned in Section 1.

For the disadvantages of ALOHA-based protocols with estimation, there emerge ALOHA-based protocols without estimation, such as Q algorithm [4], [5] in ISO 18000-6 Type C and EPC Gen 2 standard. Q algorithm utilizes an adaptive method, slot-by-slot adjusting a frame size by judging every slot type. The advantage of Q algorithm is that the frame will converge on a reasonable size without the estimation. However, since no estimation is conducted, distance between the frame length and the number of tags in Q algorithm is more than that in the protocols with estimation. Q algorithm only has an efficiency value of about 0.34 [19], [21], which is not only lower than the hybrid protocols but also lower than Dynamic FSA protocols. Although some references adopt several methods to improve Q algorithm's efficiency, for example, of optimal Q algorithm [18], the method still requires the estimation before setting a frame size. Some improved Q algorithms [19], [20] propose methods of changing step size to enhance the efficiency.

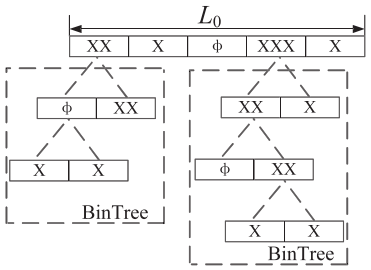

Fig. 1. An execution example of BTSA algorithm.

Although these improved algorithms conduct no estimation, they do not enhance the efficiency much and their efficiencies still does not surpass the hybrid protocols.

3 Problem Description

BTSA algorithm's prototype is proposed in [14]. The algorithm initializes a frame and divides the frame into a number of slots. Each tag will randomly selects a slot and transmit its ID to the reader only once in the slot. For a given time slot, the reader obtains only three possible outcomes: idle slot, collision slot, and successful slot. For a tag in a successful slot, the tag can be identified successfully, and the identified tag will not be activated in the subsequent slots. Tags in a collision slot collide and will be resolved by binary tree splitting at once, while other unidentified tags will wait until the collided tags are successfully resolved. In BinTree, the collided tags set will be continuously split into two sets until each set has only a tag. When tags in all slots are successfully identified, the identification completes. Fig. 1 shows an execution example of BTSA algorithm, where tags in the first slot and the fourth slot in the frame with length L_0 collide, and are resolved by two BinTrees. From Fig. 1, the algorithm is actually some BinTrees nested in a framed ALOHA algorithm. More examples of BTSA can be seen in Section 1 of supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.120.

Since BTSA is based on ALOHA algorithm, its identification efficiency will be related to the number of tags and the frame length. If the frame length is much less or larger than the number of tags, these both will produce excessive slots. Fig. 2 shows BTSA algorithm's efficiency under different Q, where the initial frame length $L = 2^Q$, and the efficiency is defined as a ratio between successful slot number and total slot number. From Fig. 2, we can see that, the efficiency is much related to the initial frame length. Only when the initial frame length is closed to the number of tags, higher efficiency can be obtained.

Therefore, if BTSA algorithm wants to obtain higher efficiency, it should let the initial frame length be close to the number of tags. Since the prior knowledge of the number of tags is generally unknown to a reader, however, the reader cannot set the initial frame length value according to the number of tags. Thus, the efficiency of BTSA algorithm cannot always obtain higher efficiency, either. Some protocols adopting BTSA algorithm conduct the estimation of the number of tags before setting the initial frame length. However, this conduction will bring about the estimation's disadvantages. The rest of this paper will propose three protocols which adjust the frame length without the tag estimation, and have higher efficiency in a wide range of the number of tags.

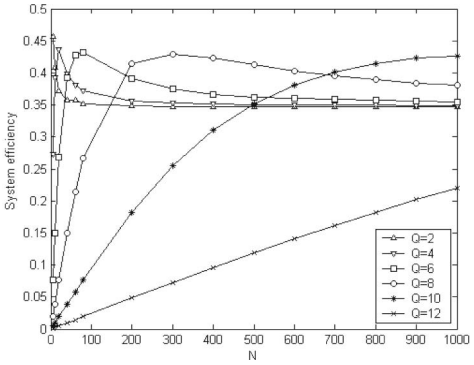

Fig. 2. BTSA algorithm's efficiency under different initial frame length.

4 BTSA Protocol

4.1 Dynamic BTSA Protocol

This section will propose a dynamic BTSA protocol, whose procedure involves dynamic frame length adjustment and BTSA algorithm. The benefit of dynamic BTSA is that its adjustment procedure is very simple because a reader can obtain a reasonable frame length by judging only the first slot type.

Fig. 3 shows a flowchart of dynamic BTSA protocol. Initially, a frame length $L = 2^Q$, where $Q_0 = 4.0$. Then, a reader broadcasts an AdjQuery command with L. After a tag receives the command with L, the tag's *Counter* selects a random integer from 0 to $L - 1$. Tags whose *Counter*s select 0 can transmit their IDs, and the reader will detect the tags responses in the first slot. If the first slot is collisional, $Q = Q + 1$, and the reader will broadcast a command with a new L and then detect tags responses in the first slot of next frame. If the first slot is idle, $Q = Q - 1$, and the reader will also broadcast a new L and then detect tags responses. If the first slot is successful, Q will not be changed and the reader's operation will transit to BTSA algorithm.

Figs. 4 and 5 show reader's pseudocode and tag's pseudocode of dynamic BTSA protocol, respectively. In Fig. 4, function DyBTSA() implements dynamic BTSA's

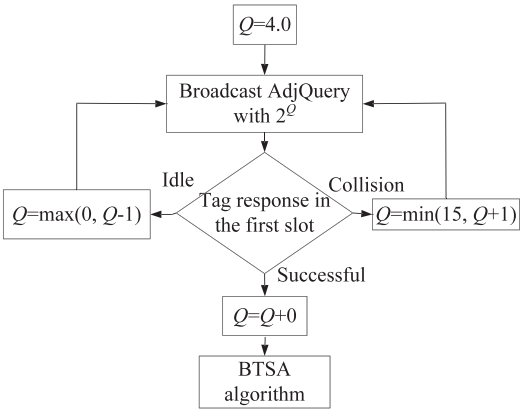

Fig. 3. Flowchart of dynamic BTSA protocol.

Dynamic BTSA protocol: reader operation

```
1    Q=4.0
2    DyBTSA(2^Q)
function DyBTSA(2^Q)
3    Broadcast AdjQuery with 2^Q
4    Receive tag response in the first slot and detect a collision
5    if tag collision
6      Q = min(Q +1, 15) and DyBTSA(2^Q)
7    elseif no tags respons
8      Q =max(Q -1, 0) and DyBTSA(2^Q)
9    elseif only a tag response
10     BTSA(2^Q)
11   end
function BTSA(L)
12   Broadcast Query with L, and SC= 0
13   do { Receive tag response and detect collision
14       if tag collision
15         Transmit f=collision, and BinTree( )
16       elseif no tag response
17         Transmit f=idle
18       else only a tag response
19         Receive ID from tag and transmit f=successful
20       end
21       SC=SC+1 } while SC<L
function BinTree( )
22   B=2
23   do { Receive tag response and detect collision
24       if tag collision
25         Transmit f=collision, and B=B+1
26       elseif no tag response
27         Transmit f=idle, and B=B-1
28       else only a tag response
29         Receive ID from tag and transmit f=successful
30         B=B-1
31       end } while B>0
```

Fig. 4. Pseudocode of dynamic BTSA protocol: reader procedure.

Dynamic BTSA protocol: tag operation

```
1    while receive Reader's AdjQuery with 2^Q
2      do{Counter=random number form 0 to 2^Q -1
3          if Counter =0   Send ID   end}
4    if receive Reader's Query with 2^Q
5      Counter=random number from 0 to 2^Q -1
6      while Counter >=0
7        do {Counter =TagIdentify(Counter)}
8    end
function TagIdentify(Counter)
9    if Counter =0
10     Send ID and receive f from reader
11     if f=successful
12       Identified, and Counter = Counter -1
13     else
14       Counter = Counter + a random binary number 0 or 1
15     end
16   else
17     Receive f from reader
18     if f=collision
19       Counter = Counter +1
20     else
21       Counter = Counter -1
22     end
23   end
24   return Counter and f
```

Fig. 5. Pseudocode of dynamic BTSA protocol: tag procedure.

procedure given in Fig. 3 and function BTSA() implements BTSA algorithm. In BTSA(), the reader will transmit a command starting a frame, Query. The reader has a *slot counter* (*SC*), whose value is initialized to 0 at the beginning of the frame and incremented by 1 at the end of each slot. When the value of *SC* is equal to the frame length L, the frame finishes. According to tags IDs received in a slot, the reader will know the type of the slot and inform all tags the type by transmitting a feedback f. If the slot is readable, that means only a tag transmit its ID to the reader. Then, the reader can identify the tag. If the slot is collisional, the reader will resolve the collided tags by BinTree splitting. The BinTree procedure is implemented by function BinTree() in reader's pseudocode and function TagIdentify() in tag's pseudocode, respectively. In BinTree(), a reader needs to judge whether a binary tree is finished or not. Let the initial value of a variable B be 2. If a current slot is collisional, $B = B + 1$, If not, $B = B - 1$. When $B = 0$, the reader know that the binary tree finishes. In TagIdentify(), a tag's *Counter* will be operated in two cases. First, when f is successful, the tag's *Counter* is decremented by 1 and the tag will not be activated in the subsequent slots. Second, when f is collision, the tag's *Counter* is incremented by a random binary number 0 or 1. If the tag's *Counter* does not select 0 when the tag receives a command Query with L, the *Counter* will also be operated in two cases. When f is collisional, the tag's *Counter* is incremented by 1; when noncollisional, the *Counter* decremented by 1.

Note that tags implementing BTSA algorithm do not need to distinguish whether they are in BinTree algorithm or ALOHA algorithm. The difference of BTSA from BinTree is only that the initial values of tag counters in BTSA can select random numbers, while those in BinTree are all 0. BTSA's counters will implement the same increment or decrement operation as BinTree algorithm after the counters select values. Therefore, BTSA will not increase the burden on tags.

Dynamic BTSA protocol adjusts a frame length actually based on a probabilistic method. If the frame length is larger than the number of tags, probability that the first slot in the frame is idle will increase; if a frame length is less then the number of tags, probability that the first slot in the frame is collision will also increase. Therefore, the adjusted frame length may be closer to the number of tags than the original frame length. For this reason, Dynamic BTSA protocol is similar to Q algorithm [3], [4] since Dynamic BTSA protocol is not sensitive to the size of tag population, either. However, Dynamic BTSA protocol adjusts the length by judging only the first slot, instead of judging every slot in Q algorithm. Furthermore, Dynamic BTSA protocol calls BinTree, which can reduce collision between tags and improve identification efficiency. More comparisons between Dynamic BTSA and Q algorithm can be seen in Section 2 of the supplementary file, available online.

4.2 Adaptive BTSA Protocol

Since EPC Gen2 has been widely applied and becoming more and more popular, we make an improvement on EPC's Q algorithm and propose adaptive BTSA in this section. The protocol will adjust a frame length based on tags responses in a current slot and can maintain a reasonable frame length for the number of tags.

Adaptive BTSA protocol first adopts the technique of Q algorithm [3], [4]: when a frame has excess collision slots, a reader will end the frame early and broadcast a command with another larger frame length; when a frame has excess idle slots, a reader will also end the frame early and

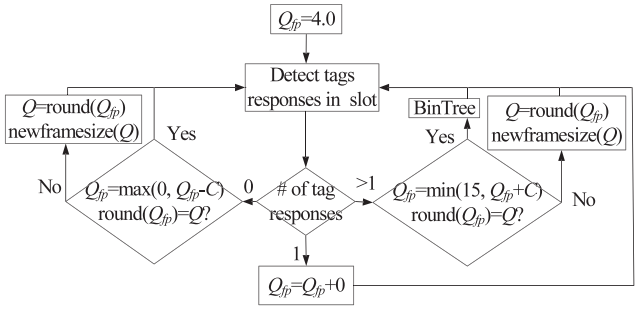

Fig. 6. Flowchart of adaptive BTSA protocol.

broadcast another less frame length. The technique can be realized by parameters Q and Q_{fp}. A frame length L is set by 2^Q and Q algorithm can adjust a frame length by adjusting Q. The value of Q is round (Q_{fp}), where Q_{fp} is a floating representation of Q. That is, a reader rounds Q_{fp} to an integer value and substitutes this integer value for Q. Initially, a frame length $L = 2^Q$, where $Q = 4.0$. Then, Q will be dynamically adjusted slot-by-slot, and Q's value is round (Q_{fp}), where Q_{fp} is given by [3], [4]

$$\begin{cases} Q_{fp} = Q_{fp} + C, & \text{if a collision slot} \\ Q_{fp} = Q_{fp} - C, & \text{if a idle slot} \\ Q_{fp} = Q_{fp} + 0, & \text{if a successful slot} \end{cases} \quad (1)$$

in which C is a step size. The selection of C is fully specified in Q algorithm, EPC C1 G2 [5]. Typical values for C are $0.1 < C < 0.5$, and small values of C when Q is large, and larger values of C when Q is small. A reader starts with a new frame by judging whether round $(Q_{fp}) = Q_{fp}$ or not. If round (Q_{fp}) is not equal to Q_{fp}, the reader will start with a new frame, whose frame length is set by a new value of Q.

Fig. 6 gives a flowchart of adaptive BTSA protocol, where function newframesize(Q) implements that a new frame start and it frame length is $2Q$. Actually, adaptive BTSA consists of Q algorithm and nested BinTrees, and its difference from Q algorithm is that when a current slot is collisional and round $(Q_{fp}) = Q$, collided tags in the slot will be resolved by BinTree at once. Only when all the collided tags in the slot are resolved successfully, tags in the next slot will be read. Figs. 7 and 8 give pseudocode of adaptive BTSA protocol.

In adaptive BTSA, an initial frame length may be much less or larger than the number of tags. However, the frame length will be slot-by-slot adjusted by (1). That is, the frame length may be adjusted along time, and eventually maintain a close value to the number of tags. Thus, the adjustment guarantees that the BinTree nested in Q algorithm has a higher value of efficiency.

4.3 Splitting BTSA Protocol

Although dynamic BTSA protocol and adaptive BTSA protocol can adjust a frame to a reasonable length for the number of tags, the adjustment cannot ensure that the frame length is exactly equal to the number of tags and thus cannot ensure that the protocols adopting BTSA algorithm achieve a maximum efficiency value. This section proposes a splitting BTSA protocol to make the frame length closer to the number of tags, and further enhance the efficiency.

Adaptive BTSA protocol: reader operation

```
1   Q=4.0
2   AdaptBTSA(Q)
function AdaptBTSA (Q)
3   Broadcast Query with 2^Q
4   SC=0, Ck=0 and Qfp= Q
5   //SC is a slot counter and Ck=is the number of collision slots
6   do { Receive tag response and detect a collision
7       if tag collision
8         f=collision, Ck= Ck+1 and Qfp= min(Qfp +C, 15)
9         if round(Qfp) != Q break
10        else
11          Transmit f , and BinTree( )
12          //the definition of BinTree( ) can be seen in Fig. 4
13        end
14      elseif no tag response
15        f=idle, and Qfp= max(Qfp - C, 0)
16        if round(Qfp) != Q break
17        else
18          Transmit f
19        end
20      elseif only a tag response
21        Receive ID from tag and store it
22        f=successful, and transmit f
23      end
24      SC=SC+1} while SC<L
25  if SC=L&&Ck=0    return
26  else
27    AdaptBTSA ( round(Qfp) )
28  end
```

Fig. 7. Pseudocode of adaptive BTSA protocol: reader procedure.

Splitting BTSA protocol consists of a splitting step and a BTSA algorithm step. Fig. 9 shows the execution of splitting BTSA protocol. The splitting step is similar to BSTSA protocol [13], where a reader will repeatedly splits a left tag set

Adaptive BTSA protocol: tag operation

```
1   Receive Reader's Query with 2^Q
2   TagQIdentify(2^Q)
function TagQIdentified(2^Q)
3   Counter=a random number form 0 to 2^Q -1
4   while Counter >=0
5     do {if Counter =0
6           Send ID
7           if receive Query with L   break
8           elseif receive f from reader
9             if f=successful
10              Identified, and Counter = Counter -1
11            else
12              Counter = Counter + a random number 0 or 1
13            end
14          end
15        else
16          if receive Query with L break
17          elseif receive f from reader
18            if f=collision
19              Counter = Counter +1
20            else
21              Counter = Counter -1
22            end
23          end
24        end }
25  if Counter <0 return
26  else
27    TagQIdentify(L)
28  end
```

Fig. 8. Pseudocode of adaptive BTSA protocol: tag procedure.

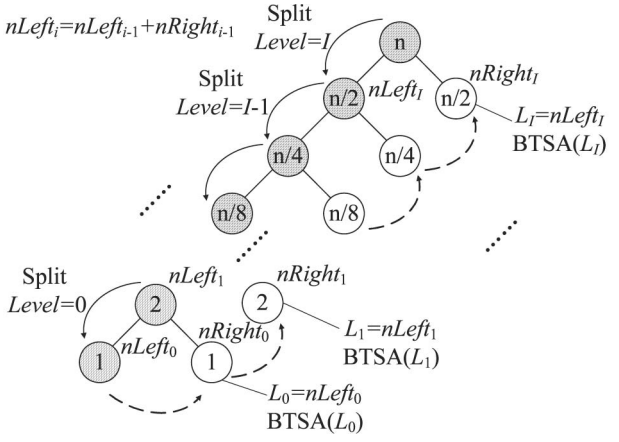

Fig. 9. Execution of splitting BTSA protocol.

produced in the last splitting process until the reader detects an idle or a successful slot. When the splitting step is finished, several right tag sets will be produced and the reader will perform the BTSA algorithm step on the right sets, respectively. If $nLeft_i$ denotes the number of tags in the ith level left set, $nLeft_i$ can be known because it is the number of identified tags and can be given by

$$nLeft_i = nLeft_{i-1} + nRight_{i-1},$$

where $nRight_{i-1}$ denotes the number of tags in the $(i-1)$th level right set. Then, an initial frame length in BTSA algorithm performed on the ith level right set, can be given by

$$L_i = nLeft_i. \tag{2}$$

Since the protocol adopts randomly binary splitting, we will have

$$nLeft_i \approx nRight_i, \quad \text{when } i \to +\infty.$$

Therefore, we can ensure that BTSA algorithm performed on each right set will have a higher efficiency value. Figs. 10 and 11 give pseudocode of splitting BTSA protocol. If a reader broadcasts a SplitQuery command, the splitting step will be started. When the splitting step is finished, levels number of binary splitting tree, *level* and feedback information of the last slot type, f can be obtained. Next, the reader performs BTSA algorithm step on each right set. An initial frame length in BTSA algorithm performed on the first right set, L_0 is set to 1, and the number of tags in the first right set, $nRight_0$ is set according to f. If f is successful, $nRight_0 = 1$; if idle, $nRight_0 = 0$. The initial frame length in the subsequent right sets, $nRight_i$, $i > 0$ is set by (1). A tag is operated on splitting step or BTSA algorithm step according to received SplitQuery or Query. If a reader broadcasts a SplitQuery command, the BTSA step will be started. In addition, BTSA algorithm in splitting BTSA protocol is a little different from that in dynamic and adaptive BTSA protocol: if a tag whose *Counter* is 0 receives Query, its *Counter* will select a random integer from 0 to $L_i - 1$; if a tag whose *Counter* is not 0 receives Query, its *Counter* will be added to a number of $L_i - 1$. The difference will make it easy for a reader to perform BTSA algorithm on each tag set, respectively.

Splitting BTSA protocol: reader operation

```
1   [level, f]=Splitting( )
2   SBTSA(level, f)
function Splitting( )
3   Broadcast SplitQuery, and level=0
4   do { Receive tag response and detect a collision
5        if tag collision
6          f=collision, and level= level +1
7        elseif no tag response
8          f=idle
9        elseif only a tag response
10         Receive ID from tag and store it
11         f=successful
12       end
13       Transmit f } while f=collision
14  return level and f
function SBTSA(level, f)
15  if f=idle
16    nLeft=0
17  elseif f=successful
18    nLeft=1
19  end
20  L=1
21  do { BTSA(L) //the definition of BTSA( ) can be seen in Fig. 4
22       nRight=CountNumSucces( )
23       //Count the number of successful slots
24       nLeft=nLeft+nRight
25       L=nLeft
26       level=level-1 } while level>0
```

Fig. 10. Pseudocode of splitting BTSA protocol: reader procedure.

5 Performance Analysis

5.1 BTSA Algorithm's Optimal Efficiency

In this section, we analyze the total slot number for identifying all tags in BTSA, and then analyze the system efficiency in BTSA. The total slot number is defined by a sum of the number of idle slots, successful slots and collision slots, and the system efficiency is defined by a ratio between the successful slot number and the total number.

Definition 1. *Let T denotes the slot number which BTSA expends on identifying tags inside a reader's range. Then, the slots number T can be given by*

$$T = T_C + T_I + T_S, \tag{3}$$

where T_C, T_I, and T_S denote the number of collision slots, idle slots, and successful slots, respectively.

Pseudocode of Splitting BTSA protocol: tag operation

```
1   if receive Reader's SplitQuery
2     Counter= a random binary number 0 or 1
3     do { [f, Counter]=TagIdentify(Counter) } while f=collision
4      //the definition of TagIdentify ( ) can be seen in Fig. 5
5   end
6   while Counter >=0
7     do {if receive Reader's Query with L
8           if Counter=0
9             Counter=a random number form 0 to L -1
10          else
11            Counter= Counter + L -1
12          end
13        elseif receive Reader's f
14          Counter=TagIdentify(Counter)
15        end }
```

Fig. 11. Pseudocode of splitting BTSA protocol: tag procedure.

Definition 2. *Let P denotes the system efficiency when BTSA identifies all tags inside a reader's range. Then, the efficiency P can be given by*

$$P = \frac{E(T_S)}{E(T_C) + E(T_I) + E(T_S)}, \tag{4}$$

where $E(\bullet)$ denotes an expectation value.

Lemma 1. *Let $T_{\mathrm{BinTree}}(r)$ denotes the number of slots expended by BinTree to identify r tags. Then, $T_{\mathrm{BinTree}}(r)$ is given by*

$$T_{\mathrm{BinTree}}(r) = \begin{cases} 4, r = 2 \\ \frac{2+\sum_{i=2}^{r-1}\binom{r}{i}0.5^{r-1}T_{\mathrm{BinTree}}(i)}{1-0.5^{r-1}}, r > 2. \end{cases} \tag{5}$$

Proof. Given one of the time slots, the number of tags allocated in the slot is a binomial distribution with r Bernoulli experiments and $1/L$ occupied probability. The probability of finding i tags in the slot is therefore given by [6], [7], [8], [9], [10], [11]

$$p(L, r, c_i) = \binom{r}{i}\left(\frac{1}{L}\right)^i\left(1 - \frac{1}{L}\right)^{r-i}. \tag{6}$$

The probability applies to all L slots; thus, the expected value of the number of slots with occupancy number is given by

$$E(L, r, c_i) = L\binom{r}{i}\left(\frac{1}{L}\right)^i\left(1 - \frac{1}{L}\right)^{r-i}. \tag{7}$$

Since the binary tree splitting can be considered as an ALOHA procedure with a frame length $L = 2$, the total number of slots expended by binary tree to identify two tags is

$$T_{\mathrm{BinTree}}(2) = 2 + E(2, 2, c_2)T_{\mathrm{BinTree}}(2).$$

Since $E(2, 2, c_2) = 0.5$, we have

$$T_{\mathrm{BinTree}}(r) = 4, r = 2. \tag{8}$$

How to obtain (8) can be seen in Section 3 of the supplementary file, available online. When $r > 2$,

$$\begin{aligned} T_{\mathrm{BinTree}}(n) &= 2 + E(2, c_n)T_{\mathrm{BinTree}}(n) \\ &\quad + \sum_{r=2}^{n-1} E(2, c_r)T_{\mathrm{BinTree}}(r). \end{aligned} \tag{9}$$

Likewise, substituting (7) into (9), we have

$$T_{\mathrm{BinTree}}(r) = \frac{2 + \sum_{i=2}^{r-1}\binom{r}{i}0.5^{r-1}T_{\mathrm{BinTree}}(i)}{1 - 0.5^{r-1}}, r > 2. \tag{10}$$

From (8) and (10), Lemma 1 can be yielded. □

Theorem 1. *The optimal number of slots expended by BTSA algorithm to identify n tags is*

$$T^*_{\mathrm{BTSA}}(n) \approx 2.33n. \tag{11}$$

Proof. Given an initial frame lengths L, the number of slots expended by BTSA algorithm to identify n tags is given by

$$T_{\mathrm{BTSA}}(L, n) = L + \sum_{r=2}^{n} E(L, c_r)T_{\mathrm{BinTree}}(r). \tag{12}$$

When $L = n$, BTSA algorithm expends the least number of slots for identifying n tags. In this case, the number of slots is optimal. When $L = n$ and $L \to +\infty$, from (7), we have

$$E(L, c_r)|_{L=n, L\to+\infty} = \frac{L}{r!e}. \tag{13}$$

Therefore, from Lemma 1 and (13), the optimal number of slots spent by BTSA algorithm to identify n tags is

$$\begin{aligned} T^*_{\mathrm{BTSA}}(n) &= T_{\mathrm{BTSA}}(L, n)|_{L=n, L\to+\infty} \\ &\approx L + L\sum_{r=2}^{n}\frac{T_{\mathrm{BinTree}}(r)}{r!e} \approx 2.33n. \end{aligned} \tag{14}$$

□

Theorem 2. *The optimal efficiency of BTSA algorithm is*

$$P^*_{\mathrm{BTSA}} \approx 0.429. \tag{15}$$

Proof. From Theorem 1 and Definition 2, we have

$$P^*_{\mathrm{BTSA}} \approx \frac{n}{2.33n} \approx 0.429.$$

Therefore, Theorem 2 can be given. □

5.2 Comparison with TSA Algorithm's Efficiency

TSA algorithm also integrates tree algorithm into ALOHA algorithm. In TSA algorithm, if tags collide in a slot, the collided tags will be L-ary splitting, where L is the number of the collided tags in the slot. In [11], [12], [13], computer results show that TSA algorithm has better efficiency performance than other pure ALOHA-based algorithms, such as Dynamic FSA and Q algorithm protocol. Since L is unknown to a reader, however, TSA algorithm requires the estimation of the number of tags. On the other hand, instead of L-ary splitting, BTSA algorithm adopts binary splitting if tags collide in a slot. In fact, if we guarantee that an initial frame is close to the number of tags, BTSA algorithm will not achieve lower efficiency than TSA algorithm. In this case, the estimation of the number of tags can be avoided, and hence the computational cost of the estimation can also be avoided. This section will further analyze BTSA algorithm's efficiency by comparing it with TSA algorithm.

Similar to conventional ALOHA-based algorithms, TSA algorithm randomly assigns tags to some slots. In TSA, if tags collide in the jth slot of the ith level frame, the collided tags in the jth slot will be reassigned to the next level frame consisting of L^j_{i+1} slots [11]. When L^j_{i+1} always satisfy

$$L^j_{i+1} = N^j_i \tag{16}$$

for any i and j, where N^j_i denotes the number of colliding tags in the jth collision slot of the ith level frame, TSA algorithm identifying n tags will expend the least slots number of $2.30n$ [11]. In this case, TSA algorithm's efficiency is an optimal efficiency of $1/2.30 = 0.435$. However, since a reader is generally difficult to know the exact number of tags in a collision slot, TSA algorithm actually set

the frame length by the following method. Given n_i tags assigned to the ith level frame where $c_{1,i}$ successful slots and $c_{k,i}$ collision slots are produced, collided tags in each collision slot of the ith level frame will be reassigned to a frame with L_{i+1} slots. The value of L_{i+1} is identical for all collision slots in the ith level frame, and is given by [11]

$$L_{i+1} = \left\lfloor \frac{n_i - c_{1,i}}{c_{k,i}} \right\rfloor. \tag{17}$$

From (17), we can see that L_{i+1} may be not exactly equal to the number of collided tags in a collision slot since $(n_i - c_{1,i})/c_{k,i}$ is only an average value of the number of colliding tags in a collision slot of the ith level frame. Therefore, the actual optimal efficiency of TSA algorithm for (17) may be lower than the ideal optimal efficiency for (16), 0.435. Next, we will derive the actual TSA algorithm efficiency to compare it with BTSA algorithm's efficiency.

Lemma 2. *The optimal number of slots expended by TSA algorithm to identify n tags is*

$$T^*_{\mathrm{TSA}}(n) \approx 2.33n. \tag{18}$$

Proof. Substituting (7) into (18), we have

$$E\left(\frac{n_0 - c_{1,0}}{c_{k,0}}\right)|_{L_0=n_0, L_0\to\infty} = \frac{e-1}{e-2} \approx 2.39. \tag{19}$$

Thus,

$$E(L_1)|_{L_0=n_0, L_0\to\infty} = \lfloor 2.39 \rfloor = 2. \tag{20}$$

From (20), the optimal slots number expended by TSA algorithm to identify n tags is

$$\begin{aligned} T^*_{\mathrm{TSA}}(n) &= T_{\mathrm{TSA}}(L_0, n_0)|_{L_0=n_0, L_0\to+\infty} \\ &\approx L_0 + L_0 \sum_{r=2}^{n} \frac{T_{\mathrm{TSA}}(2,r)}{r!e}, \end{aligned} \tag{21}$$

where $T_{\mathrm{TSA}}(L_0, n_0)$ denotes the number of slots expended by TSA algorithm, given n_0 tags and an initial frame length L_0. $T_{\mathrm{TSA}}(2,r)$ can be given by

$$T_{\mathrm{TSA}}(2,r) = 2 + \sum_{i=2}^{r} \binom{r}{i} 0.5^{r-1} T_{\mathrm{TSA}}(i,i). \tag{22}$$

For $T_{\mathrm{TSA}}(i,i)$, we can use a recursive method to obtain it. When $n = 2$,

$$T_{\mathrm{TSA}}(2,2) = 2 + E(2,c_2) T_{\mathrm{TSA}}(2,2). \tag{23}$$

Substituting (7) into (23) will produce

$$T_{\mathrm{TSA}}(i,i) = 4, i = 2. \tag{24}$$

When $i > 2$,

$$\begin{aligned} T_{\mathrm{TSA}}(i,i) &= i + E(i,c_i) T_{\mathrm{TSA}}(i,i) \\ &\quad + \sum_{j=2}^{i-1} E(i,c_j) T_{\mathrm{TSA}}(j,j). \end{aligned} \tag{25}$$

Likewise, substituting (7) into (25) can produce

$$T_{\mathrm{TSA}}(i,i) = \frac{i + \sum_{j=2}^{i-1} \binom{i}{r} (1i)^{j-1} (1-1i)^{i-j} T_{\mathrm{TSA}}(r,r)}{1 - (\frac{1}{i})^{i-1}}, i > 2 \tag{26}$$

Substituting (22), (24), and (26) into (21), (18) can be yielded. □

Theorem 3. *The optimal efficiency of TSA algorithm is not higher than that of BTSA algorithm, i.e.,*

$$P^*_{\mathrm{TSA}} \le P^*_{\mathrm{BTSA}}. \tag{27}$$

Proof. From Theorem 1 and Lemma 2, the optimal slot number of BTSA algorithm is nearly equal to that of TSA algorithm. To obtain L_{i+1}, however, TSA algorithm should first estimate the number of tags assigned to the ith level frame, n_i in (17) because n_i is generally unknown to a reader. Due to estimation error, actual TSA algorithm's efficiency will not be higher than theoretical value of $1/2.33 = 0.429$. Therefore, (27) is yielded. □

Theorem 3 signifies that, if we guarantee that an initial frame length is close to the number of tags, it is not necessary to resolve collided tags in a collision slot by L-ary splitting where L is the estimated number of the colliding tags because the efficiency in L-ary splitting will not be larger than that in binary splitting. This can avoid estimating L and hence also avoid computational cost of the estimation. In the proposed protocols, we adopt a dynamic, an adaptive, and a splitting method to make the frame length be close to the number of tags, respectively. Next, we will analysis the performance of the proposed protocols.

5.3 Analysis of the Proposed BTSA Protocol

It is seen from (12) that BTSA algorithm's efficiency is a function about an initial frame length L and the number of tags n. Also, from Fig. 2, we can see that, the efficiency is much related to the initial frame length. Only when the initial frame length is closed to the number of tags, a higher efficiency value can be obtained. Since the three proposed BTSA protocols do not adopt linear methods to adjust the initial frame length, it is some difficult to directly derive the efficiency. Therefore, we will approximately analyze the three protocols efficiency from aspect of the adjusted initial frame length.

In dynamic BTSA protocol, a reader adjusts the initial frame length by judging whether the first slot is idle or collisional and then adopts BTSA algorithm. Thus, we can analyze the efficiency from the relation of the adjusted frame length and the number of tags. We define *distance* between the adjusted initial frame length $\hat{L}$ and the number of tags n as

$$distance = \left| \frac{n - \hat{L}}{n} \right| \times 100\%. \tag{28}$$

Fig. 12 gives simulation results for *distance* of dynamic BTSA protocol. From Fig. 12, dynamic BTSA protocol's *distance* is about 40 percent. That is, $\hat{L} = 1.4n$ or $\hat{L} = 0.6n$.

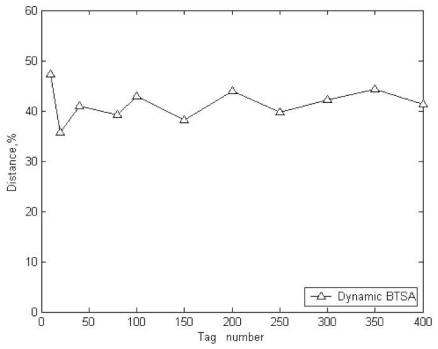

Fig. 12. Distance of adjusted frame length and the number of tags.

Substituting them into (12), we can approximately have that, dynamic BTSA protocol's efficiency is 0.40.

In adaptive BTSA protocol, we use average frame length to express the relation of the initial frame length and the number of tags. In an ALOHA protocol, suppose that the initial frame length and the number of tags satisfy $L = \beta n$ and then the efficiency can be given by

$$P = \lim_{L \to +\infty} \frac{n}{L}\left(1 - \frac{1}{L}\right)^{n-1} = \frac{e^{-1/\beta}}{\beta}. \tag{29}$$

Since a part of frame length adjustment in adaptive BTSA protocol is the same as Q algorithm, we approximately substitute an average frame length in Q algorithm for that in adaptive BTSA protocol. Authors in [19], [20], [21] show that Q algorithm's efficiency is about 0.34. Substituting 0.34 into (29), we can obtain $\beta = 0.69$ or 1.52. Substituting them into (28), we know that adaptive BTSA protocol's *distance* between the average frame length and the number of tags is about 31-52 percent. Therefore, it is concluded that adaptive BTSA protocol's efficiency may be close to dynamic BTSA protocol, 0.40.

In splitting BTSA protocol, left tag sets produced in previous splitting process will be repeatedly split. Since an initial frame length in BTSA algorithm on a right tag set is given by the value of the number of tags in a left set on the same level, we define

$$distance' = \left|\frac{nLeft - nRight}{nRight}\right| \times 100\%. \tag{30}$$

Fig. 13 shows that simulation results for *distance*′ between a left tag set and a right tag set in splitting BTSA protocol. It is seen from Fig. 13 that, *distance*′ decreases with the number of tags, and will be less than 10 percent when the number of tags is larger than 200. According to the results, we may conclude that, an initial frame length given by the number of tags of a left tag set will produce a higher efficiency value than dynamic BTSA protocol and adaptive BTSA protocol because its distance is lower than the two protocols when the number of tags increases. Likewise, substituting $\hat{L} = 1.1n$ or $\hat{L} = 0.9n$ into (12), we have that, splitting BTSA protocol's efficiency is about 0.42.

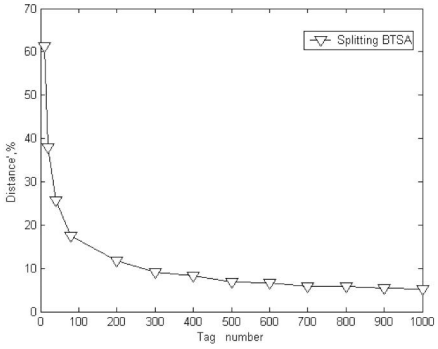

Fig. 13. Distance of left and right tag set in splitting BTSA protocol.

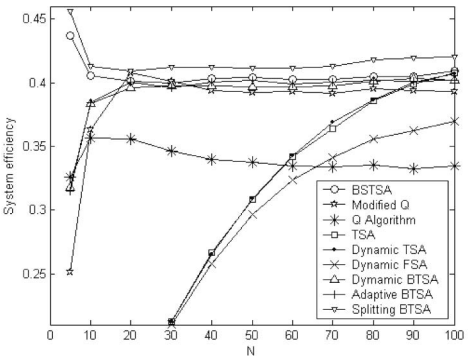

Fig. 14. Simulation results: system efficiency, $5 \leq n \leq 100$.

6 Computer Simulation Results

We evaluate the performance of the three proposed BTSA protocols by computer simulations. We individually perform each simulation 500 times, and average 500 simulation results into the final results.

Fig. 14 presents the system efficiencies of BSTSA [13], Modified Q [21], Q algorithm [3], [4], TSA [11], Dynamic TSA [12], Dynamic FSA [6], [7], [8], [9], [10], dynamic BTSA, adaptive BTSA, and splitting BTSA protocol when the number of tags increasing from 5 to 100. Here, the system efficiencies are slot efficiencies. More details can be seen in Section 4 of the supplementary file, available online. In Modified Q, Q algorithm, dynamic BTSA and adaptive BTSA protocol, an initial value of Q selects 4.0, and step C selects values as follows: If $0 \leq Q \leq 2$, $C = 0.5$; if $Q \geq 10$, $C = 0.1$; else, $C = 1/Q$. An initial frame length in TSA protocol selects the same value as in [11], $L_0 = 128$, and the initial frame length in Dynamic TSA and Dynamic FSA will also select 128. Furthermore, the estimate for the number of tags in TSA protocol and BSTSA protocol adopts the same method as in [11], i.e., Vogt estimate, whose number range is an interval of $c_{1,i} + 2c_{k,i}$ to $2(c_{1,i} + 2c_{k,i})$.

From Fig. 14, when the number of tags increases from 20 to 100, the efficiency curves of BSTSA, Modified Q, dynamic BTSA, adaptive BTSA, and splitting BTSA protocol are nearly horizontal at around 0.4. The efficiencies of the six cures range from the highest to the lowest as follows: splitting BTSA, BSTSA, adaptive BTSA, dynamic BTSA, Modified Q, and Q algorithm. For TSA protocol and Dynamic TSA protocol, their efficiencies are lower than 0.2 when the number of tags is less than 30, and their efficiencies increase with the number of tags and arrive at 0.4 when the number of tags is 100. Dynamic FSA protocol's efficiency is also lower than 0.2 when the number of tags is less than 30,

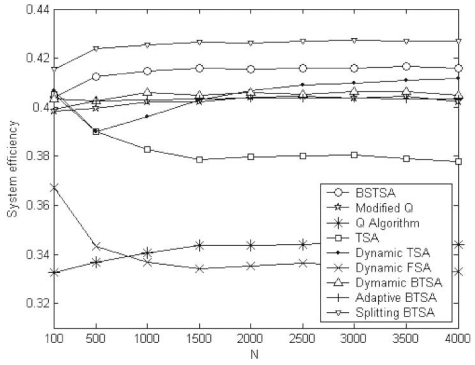

Fig. 15. Simulation results: system efficiency, $100 \leq n \leq 4{,}000$.

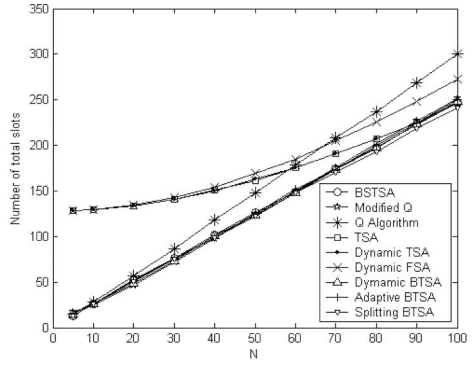

Fig. 16. Simulation results: number of total slots, $5 \leq n \leq 100$.

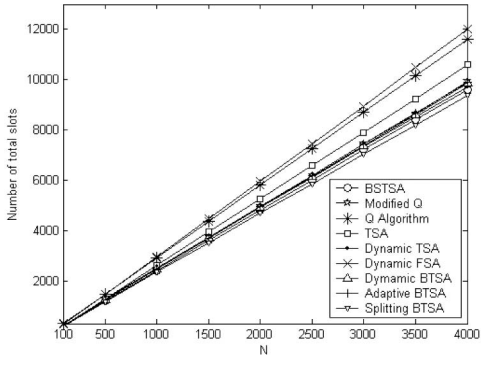

Fig. 17. Simulation results: number of total slots, $100 \leq n \leq 4{,}000$.

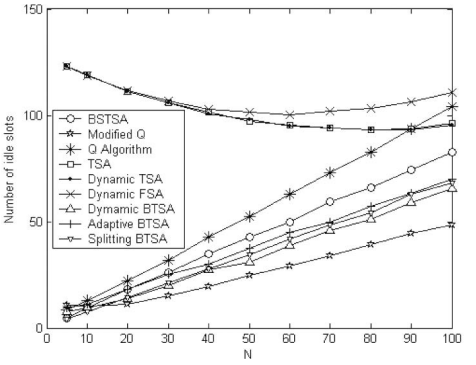

Fig. 18. Simulation results: number of idle slots, $5 \leq n \leq 100$.

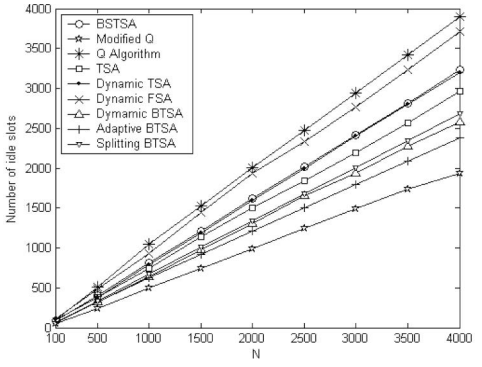

Fig. 19. Simulation results: number of idle slots, $100 \leq n \leq 4{,}000$.

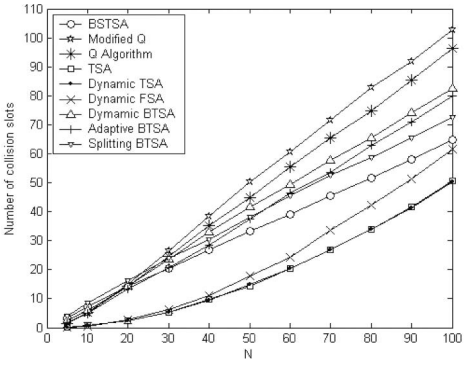

Fig. 20. Simulation results: number of collision slots, $5 \leq n \leq 100$.

and its efficiency increases with the number of tags and arrive at 0.36 when the number of tags is 100. These results show that, the variance of the number of tags will result in much variance of efficiency for TSA, Dynamic TSA, and Dynamic FSA, while the variance will not do for BSTSA, Modified Q, Q algorithm, dynamic BTSA, adaptive BTSA, and splitting BTSA protocol. In addition, the reason why splitting BTSA's efficiency is higher than BSTSA is the estimation of the number of tags. Since BSTSA adopts the estimation that will introduce error between a frame length and the number of tags, an optimal efficiency is difficult to be obtained. On the contrary, splitting BTSA protocol does not adopt the estimation, and thus it does not introduce estimation error. Splitting BTSA protocol adopts binary splitting to make sure that an initial frame length is closer to the number of tags. Hence, the efficiency can be closer to an optimal efficiency. Fig. 15 presents system efficiencies for the protocols above when the number of tags increases from 100 to 4,000, where other parameters are the same as in Fig. 14. Simulation results of Fig. 15 are similar to those of Fig. 14. Splitting BTSA has the highest value of efficiency, around 0.425 and its efficiency is not affected by the number of tags increasing or decreasing. From Figs. 14 and 15, we can see that the gap between the three proposed protocols efficiencies and BSTSA's efficiency does not surpass 0.01. That is, the proposed protocols have approximate efficiency performance to BTSA, but require no estimation and hence avoid the computational cost of the estimation.

Figs. 16 and 17 give number of slots for identifying all tags when $5 \leq n \leq 100$ and $100 \leq n \leq 4{,}000$, respectively. Similar to the results for the efficiency, splitting BTSA has better performance on identifying slot number than the other protocols. Also, Figs. 18, 19, 20, and 21 give the number of idle slots and collision slots for identifying all tags, respectively. In these figures, our proposed protocols may not have the least number of collision slots. From aspect of time efficiency which is defined as a ratio between durations for successful slots and those for total slot, the

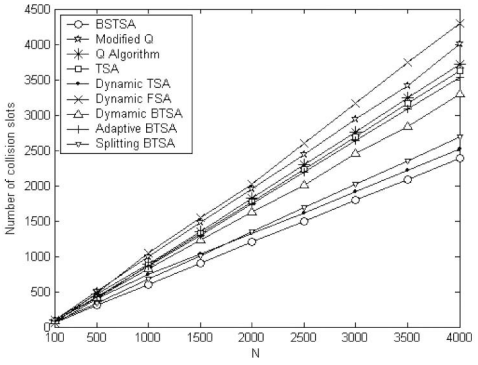

Fig. 21. Simulation results: number of collision slots, $100 \leq n \leq 4{,}000$.

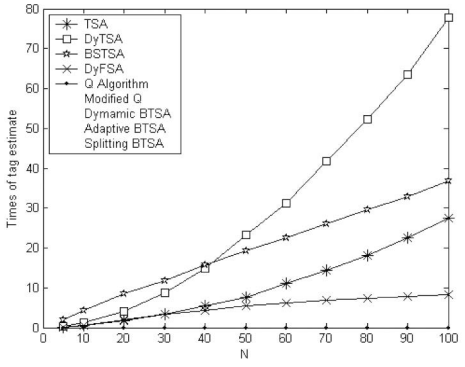

Fig. 22. Simulation results: times of estimation, $5 \leq n \leq 100$.

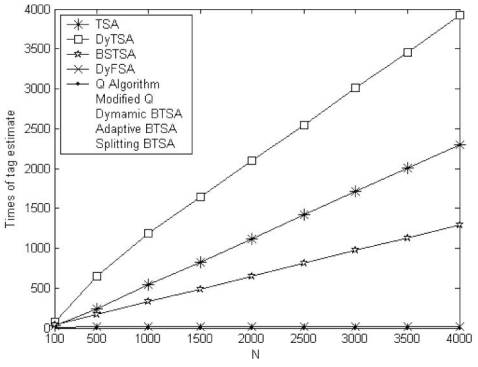

Fig. 23. Simulation results: times of estimation, $100 \leq n \leq 4{,}000$.

proposed protocols efficiency is likely lower than other protocols because a nonidle slot usually has longer duration than an idle slot [25], [26]. However, a collision slot duration can also be shorter than a successful slot duration, and be close to an idle slot. For example, ISO 18000-6C standard uses a random signature sequence with fewer bits in length than ID sequence. If a tag's signature sequence collides with others at a reader, the tag does not need to transmit its ID. In this scenario, the collision slot duration is just the shorter signature sequence duration, not the longer ID duration. From the simulation results, the gap between our proposed protocol's collision slots number and other protocols with estimation is very small. Therefore, our proposed protocols still have approximate time efficiency to those protocols with estimation.

Fig. 22 presents average times of estimation in a read cycle for the protocols mentioned above. Since Q algorithm, dynamic BTSA, adaptive BTSA, and splitting BTSA do not require the estimation of the number of tags, the times of

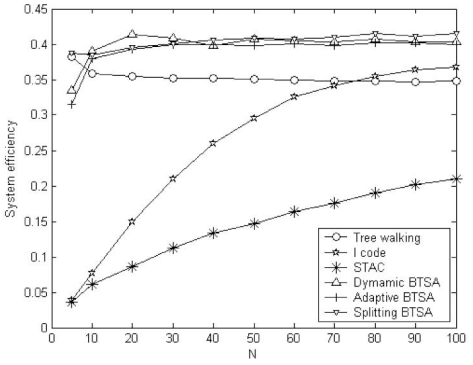

Fig. 24. Simulation results: comparison with commercialized solutions, $5 \leq n \leq 100$.

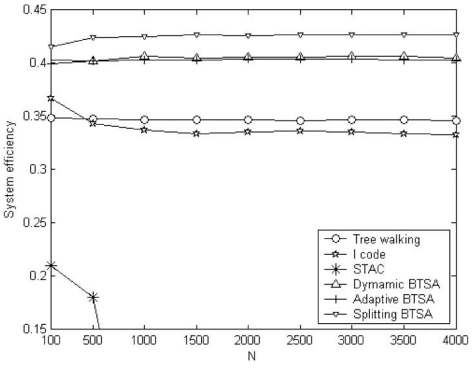

Fig. 25. Simulation results: comparison with commercialized solutions, $100 \leq n \leq 4{,}000$.

the estimation in these protocols are the least. When the number of tags is larger than 40, the times of tags estimation in the other protocols ranges from the most to the least as follows: Dynamic TSA, BSTSA, TSA, Dynamic FSA, and Modified Q. Fig. 23 also presents the times of estimation for the protocols when the number of tags increases from 100 to 4,000, and the curves ranging from the highest to the lowest in Fig. 19 are similar to those in Fig. 17.

Figs. 24 and 25 give the proposed protocol's comparison with some commercialized solutions for anticollision: Texas Instrument company's tree walking [27], Philips company's I Code [7], and TagSys Company's STAC [2]. Tree walking is a tree-based protocol and is similar to query tree protocol, where the reader queries tags whether any of their IDs contain a certain prefix, and then the tags can be split into two sets. In tree walking, we assume that each tag has a 96-bit ID. Although tree walking protocol requires no estimation, its efficiency is lower than the proposed protocols, and could only achieve an efficiency value of about 0.35. I Code is a framed slotted ALOHA protocol. In I Code, since all tags will resend their data on a second request regardless of whether they are previously successful in sending their message, some tags may be never recognized. For this reason, Vogt [7] improves I code protocol and change it into a dynamic frame length protocol. Here, the I Code solution refers to Vogt scheme. In the I code, i.e., Vogt scheme, only when the initial frame length is equal to the number of tags, the efficiency will achieve a maximum value, 0.37, which is, moreover, lower than our proposed protocols. That is, the I Code protocol may not be robust from small number to large one. In addition, the I Code needs to adjust a frame

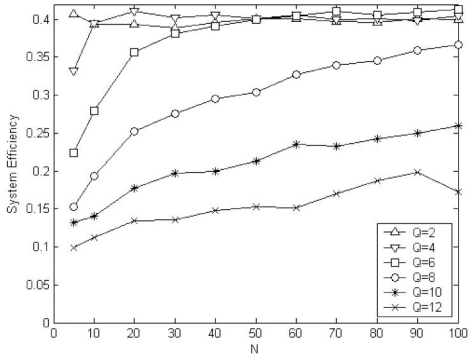

Fig. 26. Simulation results: dynamic BTSA efficiency under different Q, $5 \leq n \leq 100$.

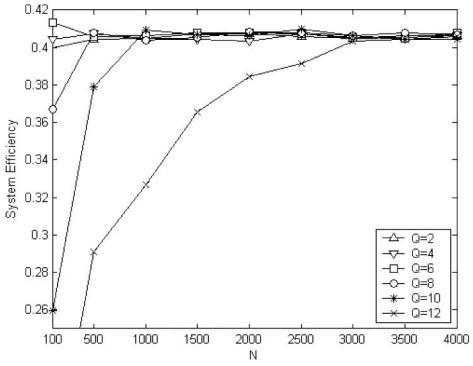

Fig. 27. Simulation results: dynamic BTSA efficiency under different Q, $100 \leq n \leq 4{,}000$.

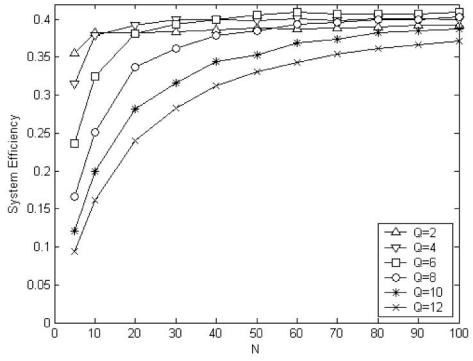

Fig. 28. Simulation results: adaptive BTSA efficiency under different Q, $100 \leq n \leq 4{,}000$.

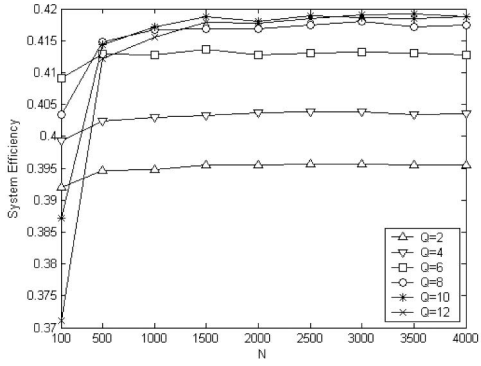

Fig. 29. Simulation results: adaptive BTSA efficiency under different Q, $5 \leq n \leq 4{,}000$.

length according to the number of tags. Thus, it also requires the estimation of the number of tags and increases the computational cost. STAC is a fixed frame length ALOHA protocol and requires no estimation. Instead of dynamically being adjusted, however, each frame length in STAC is fixed. Thus, the efficiency of STAC is much lower than our proposed protocols.

Figs. 26 and 27 present the efficiency of dynamic BTSA protocol under different initial values of Q when the number of tags increases from 5 to 100 and from 100 to 4,000, respectively. When the number of tags $5 \leq n \leq 100$, the system efficiency under $Q \geq 8$ does not arrive at 0.4; only when $n > 100$, the efficiency will surpass 0.4. The system efficiency under $Q \leq 2$ can arrive at about 0.4 when $n < 100$; however, the efficiency will be lower than 0.4 when $n > 100$. Therefore, it is seen from the results that, the initial value of Q, 4.0 is a compromise, which makes the efficiency have a higher value both when the number of tags increases and decreases. Figs. 28 and 29 present the system efficiencies of adaptive BTSA protocol under different initial values of Q when the number of tags increases from 5 to 100 and from 100 to 4,000, respectively. From the two figures, if the initial value of Q selects 4.0, the efficiencies will also have a higher value both when the number of tags increases and decreases.

7 Conclusion

When an RFID system identifies multiple tags, tag collisions will happen. The RFID system generally applies a tag anticollision protocol to resolve the multitag collisions. This paper utilizes BTSA algorithm to propose three protocols: dynamic BTSA protocol, adaptive BTSA protocol, and splitting BTSA protocol. The proposed protocols not only have higher efficiency but also require no estimation of the number of tags, and hence can avoid the computational cost of the estimation. Furthermore, since the proposed protocols always can adjust a frame to a reasonable length for the number of tags, their efficiency will not be affected by the variance of the number of tags. When the number of tags suddenly increases or decreases much, the underutilization of channel and low efficiency will not happen.

Acknowledgments

This work was supported in part by Applied and Basic Research Foundation of Yunnan Province under Grant No. 2011FB083, the Scientific Research Foundation of Yunnan Provincial Department of Education under Grant No. 2011Y217, the Open Foundation of Key Laboratory of Wireless Sensor Network Technology of Yunnan Province under Grant No. ZK2011001, the Major Special Project of Scientific Research Foundation of Yunnan Provincial Department of Education under Grant No. ZD2011009, and a grant from Innovative Research Team in Yunnan University of Nationalities.

References

[1] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, and G. Borriello, "Building the Internet of Things Using RFID: The RFID Ecosystem Experience," *IEEE Internet Computing*, vol. 13, no. 3, pp. 48-55, May/June 2009.

[2] "13.56 MHz ISM Band Class 1 Radio Frequency Identification Tag Interface Specification: Recommended Standard," technical report, Version 1.0.0, Auto-ID Center, 2003.
[3] *Information Technology - Radio Frequency Identification (RFID) for Item Management - Part 6: Parameters for Air Interface Communications at 860 MHz to 960 MHz,* Int'l Standard ISO/IEC 18000-6, 2004.
[4] *Information Technology - Radio Frequency Identification (RFID) for Item Management Part 6: Parameters for Air Interface Communications at 860 MHz to 960 MHz, Amendment1: Extension with Type C and Update of Types A and B,* Int'l Standard ISO/IEC 18000-6, 2006.
[5] *EPC Radio-Frequency Identification Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz-960 MHz,* Version 1.1.0 Draft1. EPCglobal, Inc, 2005.
[6] F.C. Schoute, "Dynamic Frame Length Aloha," *IEEE Trans. Comm.,* vol. C-31, no. 4, pp. 565-568, Apr. 1983.
[7] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. Int'l Conf. Pervasive Computing,* pp. 98-113, 2002.
[8] S.R. Lee, S.D. Joo, and C.W. Lee, "An Enhanced Dynamic Framed ALOHA Algorithm for RFID Tag Identification," *Proc. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services,* pp. 1-5, 2005.
[9] W.T. Chen, "An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length ALOHA," *IEEE Trans. Automation Science and Eng.,* vol. 6, no. 1, pp. 9-15, Jan. 2009.
[10] H. Wu and Y. Zeng, "Bayesian Tag Estimate and Optimal Frame Length for Anti-Collision ALOHA RFID System," *IEEE Trans. Automation Science and Eng.,* vol. 7, no. 4, pp. 963-969, Oct. 2010.
[11] M.A. Bonuccelli, F. Lonetti, and F. Martelli, "Tree Slotted ALOHA: A New Protocol for Tag Identification in RFID Networks," *Proc. Int'l Symp. World of Wireless, Mobile and Multimedia Networks,* pp. 1-6, 2006.
[12] G. Maselli, C. Petrioli, and C. Vicari, "Dynamic Tag Estimation for Optimizing Tree Slotted ALOHA in RFID Networks," *Proc. ACM 11th Int'l Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '08),* pp. 315-322, 2008.
[13] L.T. Porta, G. Maselli, and C. Petrioli, "Anti-Collision Protocols for Single-Reader RFID Systems: Temporal Analysis and Optimization," *IEEE Trans. Mobile Computing,* vol. 10, no. 2, pp. 267-279, Feb. 2011.
[14] J.I. Capetanakis, "Tree Algorithms for Packet Broadcast Channels," *IEEE Trans. Information Theory,* vol. IT-25, no. 5, pp. 505-515, Sept. 1979.
[15] D.R. Hush and C. Wood, "Analysis of Tree Algorithm for RFID Arbitration," *Proc. IEEE Int'l Symp. Information Theory,* pp. 107-107, 1998.
[16] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. 12th ACM Ann. Int'l Conf. Mobile Computing and Networking Table of Contents,* pp. 322-333, 2006.
[17] H. Wu and Y. Zeng, "Efficient Framed Slotted ALOHA Protocol for RFID Tag Anticollision," *IEEE Trans. Automation Science and Eng.,* vol. 8, no. 3, pp. 581-588, July 2011.
[18] Y. Maguire and R. Pappu, "An Optimal Q-Algorithm for the ISO 18000-6C RFID Protocol," *IEEE Trans. Automation Science and Eng.,* vol. 6, no. 1, pp. 16-24, Jan. 2009.
[19] D. Lee, K. Kim, and W. Lee, "Q+-Algorithm: An Enhanced RFID Tag Collision Arbitration Algorithm," *Proc. Conf. Ubiquitous Intelligence and Computing (UIC '07),* pp. 1-10, 2007.
[20] I. Joe and J. Lee, "A Novel Anti-Collision Algorithm with Optimal Frame Size for RFID System," *Proc. Fifth ACIS Int'l Conf. Software Eng. Research, Management & Applications,* pp. 424-428, 2007.
[21] Y. Cui and Y. Zhao, "A Modified Q-Parameter Anti-Collision Scheme for RFID Systems," *Proc. Int'l Conf. Ultra Model Telecomm. and Workshop (ICUMT '09),* pp. 1-4, 2009.
[22] J. Myung, W. Lee, J. Srivastava, and T.K. Shih, "Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 6, pp. 763-775, June 2007.
[23] J. Park, M.Y. Chung, and T.J. Lee, "Identification of RFID Tags in Framed-Slotted ALOHA with Robust Estimation and Binary Selection," *IEEE Comm. Letters,* vol. 11, no. 5, pp. 452-454, May 2007.
[24] C. Qian, Y. Liu, H. Ngan, and L.M. Ni, "ASAP: Scalable Identification and Counting for Contactless RFID Systems," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '10),* pp. 52-61, 2010.
[25] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem" *Proc. IEEE INFOCOM,* 2010.
[26] H. Han, B. Sheng, C.C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," *Proc. IEEE INFOCOM,* 2010.
[27] A. Juels, R.L. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy," *Proc. ACM Conf. Computer and Comm. Security (CCS '03),* pp. 1-9, 2003.
[28] C. Qian, H. Ngan, and Y. Liu, "Cardinality Estimation for Large-Scale RFID Systems," *Proc. Sixth IEEE Int'l Conf. Pervasive Computing and Communications (ICPCC '08),* pp. 30-39, 2008.
[29] Y. Zheng, M. Li, and C. Qian, "PET: Probabilistic Estimating Tree for Large-Scale RFID Estimation," *Proc. 31st IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 37-46, 2011.
[30] D. Simplot-Ryl, I. Stojmenovic, A. Micic, and A. Nayak, "A Hybrid Randomized Protocol for RFID Tag Identification," *Sensor Rev.,* vol. 26, no. 2, pp. 147-154, 2006.

Haifeng Wu received the MS degree in electrical engineering from Yunnan University, Kunming, China, in 2004, and the PhD degree in electrical engineering from Sun Yat-Sen University, Guangzhou, China, in 2007. He is currently an assistant professor at the Department of Information Engineering, Yunnan University of Nationalities. Prior to that, he was a postdoctoral scholar in the Kunchuan Institute of Technology from 2007 to 2009. His research interests include RF engineering, mobile communications and cooperative sensor networks.

Yu Zeng received the MS degree in electrical engineering from Yunnan University, Kunming, China, in 2006. She is currently an assistant professor at the Department of Information Engineering at the Yunnan University of Nationalities. Prior to that, she was an electrical engineer in Kunming Institute of Physics from 2006 to 2009. Her research interests include wireless network, mobile communications.

Jihua Feng received the MS degree in electrical engineering from Yunnan University, Kunming, China, in 2006, and the PhD degree in electrical engineering from Sun Yat-Sen University, Guangzhou, China, in 2010. He is currently an assistant professor at the Department of Information Engineering, Yunnan University of Nationalities. His research interests include adaptive signal processing and RF engineering.

Yu Gu received the MS degree in basic mathematics from Yunnan Normal University, China, in 2003, and the PhD degrees in computer science and engineering from Xi'an Jiaotong University, in 2009, respectively. He is now a professor at the School of Education, Yunnan University of Nationalities. His research interests include wireless sensor network, peer-to-peer computing, and pervasive computing.