Open access • Journal Article • DOI:10.1109/TPDS.2007.1098

Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification

Jihoon Myung, Wonjun Lee, Jaideep Srivastava, Timothy K. Shih

Published on: 01 Jun 2007 - IEEE Transactions on Parallel and Distributed Systems (IEEE Computer Society)

Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification

Jihoon Myung, *Student Member*, *IEEE*, Wonjun Lee, *Senior Member*, *IEEE*, Jaideep Srivastava, *Fellow*, *IEEE*, and Timothy K. Shih, *Senior Member*, *IEEE*

Abstract—Tag identification is an important tool in RFID systems with applications for monitoring and tracking. A RFID reader recognizes tags through communication over a shared wireless channel. When multiple tags transmit their IDs simultaneously, the tag-to-reader signals collide and this collision disturbs a reader's identification process. Therefore, tag collision arbitration for passive tags is a significant issue for fast identification. This paper presents two adaptive tag anticollision protocols: an Adaptive Query Splitting protocol (AQS), which is an improvement on the query tree protocol, and an Adaptive Binary Splitting protocol (ABS), which is based on the binary tree protocol and is a de facto standard for RFID anticollision protocols. To reduce collisions and identify tags efficiently, adaptive tag anticollision protocols use information obtained from the last process of tag identification. Our performance evaluation shows that AQS and ABS outperform other tree-based tag anticollision protocols.

Index Terms—Collision resolution, RFID, tag anticollision, tag identification.

1 Introduction

Radio frequency identification (RFID) is an automatic identification system which consists of readers and tags. A tag has an identification number (ID) and a reader recognizes an object through consecutive communications with the tag attached to it [1]. The reader sends out a signal which supplies power and instructions to a tag. The tag transmits its ID to the reader and the reader consults an external database with the received ID to recognize the object. RFID is fast replacing bar-code-based identification mechanisms because 1) communication between a reader and a tag is not limited by the requirement of "line-of-sight" reading and 2) each tag is allowed to have a unique ID.

Reader transmissions or tag transmissions in a RFID system lead to collisions because readers and tags communicate over a shared wireless channel. Collisions make both communication overhead and transmission delay often lose their usefulness. As a result, either the reader may not recognize all objects or retransmissions are required for successful transmission. Collisions are divided into reader collisions and tag collisions [2], [3]. Reader collisions occur when neighboring readers interrogate a tag simultaneously [4], [5]. Tag collisions occur when multiple tags transmit IDs to a reader at the same time and prevent the reader from recognizing any tag [6]. Especially, since low-functional passive tags can neither detect collisions nor figure out neighboring tags, a tag collision gives rise to the need for a tag anticollision protocol that enables the recognition of tags with few collisions and also executes in real-time.

Tag anticollision protocols can be grouped into two broad categories: aloha-based protocols and tree-based protocols. Aloha-based protocols such as aloha [7], slotted aloha [8], [9], [10], [11], [12], and frame slotted aloha [13], [14], [15], [16], [17], [18], [19] reduce the occurrence probability of tag collisions since tags transmit at distinct times. Since aloha-based protocols, however, cannot completely prevent collisions, they have the serious problem that a specific tag may not be identified for a long time, leading to the so-called "tag starvation problem." On the other hand, tree-based protocols such as the binary tree protocol [18], [19], [20], [21], [22], [23] and the query tree protocol [24], [25], based on the collision resolution algorithm studied in [26], [27], continuously split a set of tags into two subsets until each set has only one tag. Although they have relatively long identification delay, they do not cause the tag starvation problem.

Based on the analysis above, a good tag collision arbitration protocol for RFID passive tags should have the following characteristics: First, a reader ought to recognize all the tags inside its own reading range. The tag starvation problem results in the failure of object tracking and monitoring. Since the reader, however, cannot presume the number of tags precisely, the guarantee of recognizing all tags must be taken into consideration in the design of the tag anticollision protocol. Second, a reader has to recognize tags promptly. Since an object with a tag is potentially mobile, tag identification must keep pace with the object's velocity. If tag identification is carried out slower than the

- *J. Myung is with the Department of Computer Science and Engineering, College of Information and Communications, Korea University, Room 543B, Asan Science Building, 1, 5-ga, Anam-dong, Sungbuk-gu, Seoul 136-701, Republic of Korea. E-mail: jmyung@korea.ac.kr.*
- *W. Lee is with the Department of Computer Science and Engineering, College of Information and Communications, Korea University, Room 507, Asan Science Building, 1, 5-ga, Anam-dong, Sungbuk-gu, Seoul 136-701, Republic of Korea. E-mail: wlee@korea.ac.kr.*
- *J. Srivastava is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. E-mail: srivasta@cs.umn.edu.*
- *T.K. Shih is with the Department of Computer Science and Information Engineering, Tamkang University, 151 Ying-chuan Road, Tamsui, Taipei County, Taiwan 251, Republic of China. E-mail: tshih@cs.tku.edu.tw.*

Manuscript received 9 Nov. 2005; revised 18 Aug. 2006; accepted 24 Aug. 2006; published online 9 Jan. 2007.
Recommended for acceptance by C. Shahabi.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0468-1105.
Digital Object Identifier no. 10.1109/TPDS.2007.1020.

object's velocity, the reader cannot recognize it and the RFID system fails in monitoring or tracking. Finally, a tag should be recognized while consuming a small amount of resource. Since the tag supplements power from the reader's wave, the tag's available power is limited. Also, the tag has low computational capability and limited memory. Thus, the tag anticollision protocol must load the tag with the least possible communication and computation overheads.

We propose adaptive tag anticollision protocols which are specializations of tree-based protocols. An Adaptive Query Splitting protocol (AQS) is an improvement on the query tree protocol which has a deterministic methodology. The preliminary version of AQS was presented in [28]. An Adaptive Binary Splitting protocol (ABS) is based on the binary tree protocol which adopts a probabilistic approach. For decreasing collisions, the proposed protocols use information obtained from the last identification process in an environment where a reader executes tag identification repeatedly for object tracking and monitoring. The reduction in collisions facilitates tag identification with a small delay and low communication overhead while still recognizing all tags. Simulation results show that AQS and ABS suppress the occurrence of collisions and shorten the total delay for recognizing all tags while preserving low communication overhead.

The rest of this paper is organized as follows: Section 2 describes existing tree-based tag anticollision protocols. We discuss and analyze problems of existing tree-based protocols in Section 3. Section 4 explains our approach for the resolution of problems in tree-based protocols. Sections 5 and 6 describe AQS and ABS, respectively. In Sections 7 and 8, the performance analysis, including the derivations of several major equations referred to in the analysis, is presented. Finally, the conclusions of our analysis are presented in Section 9.

2 Tree-Based Tag Anticollision Protocols

Tree-based tag anticollision protocols perform tag identification in units of "*interrogation cycle.*" The reader recognizes all the tags within its reading range in a frame, which consists of several interrogation cycles. In an interrogation cycle, a reader transmits a query (or a feedback) to tags and then tags transmit IDs to the reader. Since a passive tag cannot detect a collision, the reader detects whether a tag collision occurs or not after its transmission and determines the query of the next interrogation cycle according to the result of the detection. Upon receiving a query from the reader, the tag decides whether to transmit or not. Only if a single tag transmits in an interrogation cycle can the reader recognize it successfully.

The reader attempts to recognize a set of tags in an interrogation cycle. A set includes tags, which transmit at the same interrogation cycle. If a set has more than one tag, tag transmissions lead to a collision. When a tag collision occurs, the mechanisms split the set into two subsets by tag IDs or random numbers. After that, the reader tries to recognize two subsets one by one in the same frame. By continuing the splitting procedure until each set has only one tag, tree-based protocols are capable of recognizing all tags in the reader's range. The performance of tag identification is influenced significantly by how efficiently it splits the tag set.

2.1 Binary Tree Protocol

The binary tree protocol (BT) [18], [19], [20], [21], [22], [23] uses random binary numbers generated by colliding tags for the splitting procedure. The tag has a counter initialized to 0 at the beginning of the frame. The tag transmits its ID when the counter value is 0. Therefore, all the tags within the reader's reading range, at the start of the frame, form one set and transmit concurrently. The reader transmits a feedback to inform tags of the occurrence of a tag collision. According to the reader's feedback, all the tags change its counter. The tag randomly selects a binary number when its transmission causes the collision (i.e., the counter value is 0). By adding the selected binary number to the counter, a set is split into two subsets. When a tag collision occurs, the tag which is not involved in this collision (i.e., the counter value is greater than 0) increases its counter by 1. When the reader's feedback indicates no collision, all tags decrease their counter by 1. The tag infers the successful transmission from the following feedback indicating no collision. The tag recognized by a reader does not transmit any signal until the ongoing frame is terminated.

In BT, the reader also has a counter in order to terminate a frame. It initializes the counter with 0 in every frame. The counter value of the reader indicates the number of tag sets which are not recognized. If a tag collision occurs, the reader adds 1 to the counter since the number of tag sets, which the reader should recognize, increases. Otherwise, it decreases the counter by 1. When the counter is less than 0, the reader terminates the frame.

2.2 Query Tree Protocol

The query tree protocol (QT) [24], [25] uses tag IDs to split a tag set. The reader transmits a query including a bit string. The tag whose first bits of the ID equal the bit string of the query responds by transmitting its ID. For query $q_1q_2 \dots q_x$ ($q_i \in \{0,1\}$), the reader uses two 1-bit longer queries, $q_1q_2 \dots q_x0$ and $q_1q_2 \dots q_x1$, in the next interrogation cycles if tag responses collide. The set of tags which match $q_1q_2 \dots q_x$ is split into two subsets; one is a set of tags which match $q_1q_2 \dots q_x0$ and the other is a set of tags which match $q_1q_2 \dots q_x1$.

The reader has queue Q for bit strings of queries. At the beginning of the frame, Q is initialized with two 1-bit strings, 0 and 1. The reader pops a bit string from Q and transmits a query at a time. If tag responses collide, the reader pushes two 1-bit longer bit strings into Q. By expanding the query until either a response or no response follows, all the tags are recognized.

Contrary to BT, QT imposes simple functions on tags. QT is also called a "memoryless" protocol because tags need not have additional memory except IDs for identification. However, the identification delay is affected by the distribution of tag IDs. As tags have very similar IDs, delay is increased.

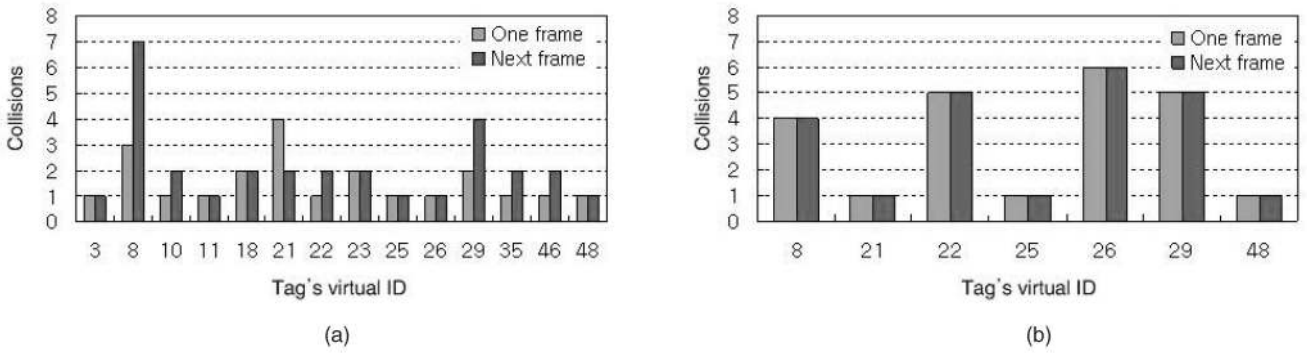

Fig. 1. Collisions between staying tags in the tree-based protocols. (a) Binary tree protocol. (b) Query tree protocol.

3 Problem Description

We assume that a reader performs tag identification processes, namely, frames, repeatedly for object tracking and monitoring.[1] For reader r, let $A_{r,i}$ be the set of tags which dwell inside reader r's range in the ith frame of reader r. To consider the tag's mobility, we also classify tags into staying tags, arriving tags, and leaving tags. We say that tag a_s is the *staying tag* in the $i+1$th frame of reader r if $a_s \in A_{r,i} \cap A_{r,i+1}$. We say that tag a_a is the *arriving tag* in the $i+1$th frame of reader r if $a_a \in A_{r,i+1} - A_{r,i}$. We say that tag a_l is the *leaving tag* in the $i+1$th frame of reader r if $a_l \in A_{r,i} - A_{r,i+1}$. Tag identification should recognize staying tags and arriving tags promptly.

Staying tags have been recognized in the last frame and the reader will rerecognize staying tags. Since the reader already has information on staying tags, tag collision arbitration can prevent collisions between transmissions of staying tags. However, the existing tag anticollision protocols cause collisions between staying tags because they do not take staying tags into consideration. To show collisions between staying tags, we measure interstaying tag collisions through simulations of BT and QT. The simulation area is $10\ \mathrm{m} \times 10\ \mathrm{m}$. There exists a reader at the center of the area and the reader's reading range is 3 m. Fifty tags have randomly selected 96-bit IDs and move randomly inside the area. To pin-point an individual tag, we give tags virtual IDs from 1 to 50. We fix tag 1 in the vicinity of the reader to make it the staying tag. Fig. 1 shows the number of collisions between tag 1 and other staying tags during two consecutive frames. Under BT, tag 1 suffers from more collisions with some staying tags during the next frame in comparison to the last frame as shown in Fig. 1a. This is due to the fact that BT resets the counters of tags to 0 at the beginning of the frame and uses random numbers. Under QT, tag 1 causes the same number of collisions with other staying tags again as shown in Fig. 1b. At the beginning of the frame, QT initializes queue Q of the reader with 1-bit queries.

When a collision occurs in tag identification of tree-based protocols, the colliding tag needs to retransmit its ID. The retransmission brings about an increase in identification delay, which handicaps the identification capability of the reader. To make matters worse, readers and tags consume significantly extra energy for successful transmission. Therefore, eliminating interstaying tag collisions can shorten total delay for tag identification and reduce tag communication overhead.

1. Prominent retailers such as Wal-Mart, Target, and Best Buy, as well as logistics companies like UPS and Fed-Ex, have made this a requirement of all their operations. Manufacturers are following suit.

4 Adaptive Tag Anticollision Protocols

A frame of tree-based protocols is represented by a tree structure as shown in Fig. 2. Each node in the tree corresponds to an interrogation cycle and a number in a node is the number of tag transmissions in that interrogation cycle. According to the number of tag transmissions in an interrogation cycle, interrogation cycles can be divided into three types as follows:

- *Idle cycle*: No transmission is attempted. The idle cycle does not make the reader fail to notice a tag, but it is a source of an unnecessary increment of identification delay.
- *Readable cycle*: Exactly one transmission is attempted. The reader recognizes a tag successfully.
- *Collision cycle*: More than one transmission is attempted. A tag collision occurs and the reader is unable to recognize any tags. The collision cycle defers tag identification and the tag's communication is pure overhead.

Only a node of a collision cycle has two child nodes, since a set is split into two subsets only in the collision cycle. Consequently, all intermediate nodes in the tree correspond to collision cycles and all the leaf nodes correspond to either readable cycles or idle cycles.

Tag identification is coincident with a tree search, which starts at the root of the tree for finding nodes of readable cycles. The reduction of delay in tag identification can be accomplished by skipping collision cycles. However, once a frame is started, the tree searches of BT and QT depart from

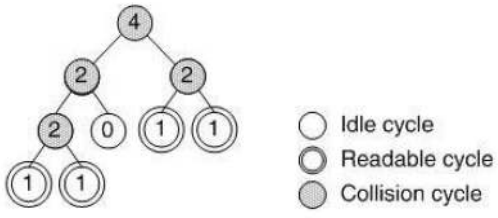

Fig. 2. Tag identification of tree-based protocols.

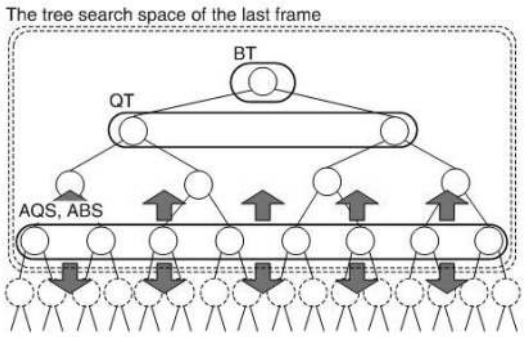

Fig. 3. Starting points of tag identification in tree-based protocols.

the root or the 1-level nodes of the tree and investigate all intermediate nodes. In short, the unreasonable starting point of existing protocols prolongs the identification delay.

Adaptive tag anticollision protocols have the characteristic of fast tag identification. AQS uses readers' queries and tag IDs analogous to QT, but it starts the tree search from the leaf nodes in the tree of the last frame as shown in Fig. 3. Since some arriving tags may not match any nodes of readable cycles of the last frame, the tree search starts at the nodes of idle cycles as well as the nodes of readable cycles of the last frame. On the other hand, ABS uses random numbers and starts the tree search from the nodes of readable cycles of the last frame. An arriving tag gets to belong to a node of a readable cycle of the last frame by a random selection at the beginning of a new frame. AQS and ABS restrain collisions between staying tags by skipping interrogation cycles in which a tag collision occurred in the last frame. They are still simple and recognize all tags quickly.

5 Adaptive Query Splitting

AQS recognizes tags using queries sent by a reader. A query includes a bit string. The tag responds with its ID when its first bits of the ID are equal to the bit string of the query as shown in Fig. 4a. Collisions are resolved by two 1-bit longer queries. The reader has queue Q which maintains bit strings for queries. At the beginning of the frame, Q is initialized with queries of all the leaf nodes in the tree of the last frame. To do this, the reader also has candidate queue CQ, which compiles queries of readable cycles and idle cycles of the ongoing frame. When a new frame starts, the reader initializes Q with bit strings in CQ and makes CQ empty. Accordingly, the reader does not transmit queries which caused tag collisions in the last frame. Tags still require very simple functions such as matching their IDs with the queries and are recognized with few collisions.

5.1 Query Insertion

Fig. 4b shows the pseudocode of the reader in AQS. In lines 11 and 12, the reader pops a bit string from Q and transmits a query in an interrogation cycle. For query $q_1q_2 \ldots q_x (q_i \in \{0,1\}, 1 \leq x \leq b$, and b is the number of bits in the tag ID), the reader pushes $q_1q_2 \ldots q_x0$ and $q_1q_2 \ldots q_x1$ into Q in lines 16 and 17 if tag responses collide. The reader pushes $q_1q_2 \ldots q_x$ into CQ in lines 20 and 22 if $q_1q_2 \ldots q_x$ engenders an idle cycle or a readable cycle. Note that all leaf nodes correspond to either readable cycles or idle cycles. At the end of the frame, CQ has queries of all the leaf nodes of the tree. The query expansion from leaf nodes resolves collisions with few interrogation cycles. By queries of leaf nodes of the last frame, the reader can recognize all tags with few collisions and a small delay in the next frame.

```
Algorithm 1. AQS Tag Operation
/* Respond to the reader's query
   Query q is q_1q_2...q_x
      (q_i = a binary value, x = the length of the query)
   Tag ID is t_1t_2...t_b
      (t_i = a binary value, b = the length of ID)
   isResponsible is a flag to determine whether the tag transmits
its ID or not */

1   Receive the command starting a frame from the reader
2   Receive message m from the reader
3   while m != the command terminating a frame do
4       q = m
5       isResponsible = 1
6       for all i such that 0 • i < the number of bits of q do
7           if q_i != t_i then
8               isResponsible = 0
9               break
10          end if
11      end for
12      if isResponsible = 1 then
13          Transmit ID
14      end if
15      Receive message m from the reader
16      end while
```

(a)

```
Algorithm 2. AQS Reader Operation (candidate queue CQ)
/* Transmit queries and receive tag responses */

1   /* Initialize Q and CQ */
2   Q = CQ
3   CQ = NULL
4   if Q = NULL then
5       Push (Q, 0)
6       Push (Q, 1)
7   end if
8   /* Identify tags and form CQ */
9   Transmit the command starting a frame
10  while Q != NULL do
11      q = Pop(Q)
12      Transmit a query including q
13      Receive tag responses and detect a collision
14      if tag collision then
15          /* Push 1-bit longer bit strings into Q */
16          Push (Q, q0)
17          Push (Q, q1)
18      else if only a tag response then
19          Store the tag ID
20          Push (CQ, q)  /* Push q into CQ */
21      else if no tag response then
22          Push (CQ, q)  /* Push q into CQ */
23      end if
24  end while
25  QueryDeletion(CQ)  /* Remove unnecessary queries */
26  Transmit the command terminating a frame
```

(b)

Fig. 4. Pseudocode of AQS. (a) Tag's operation. (b) Reader's operation.

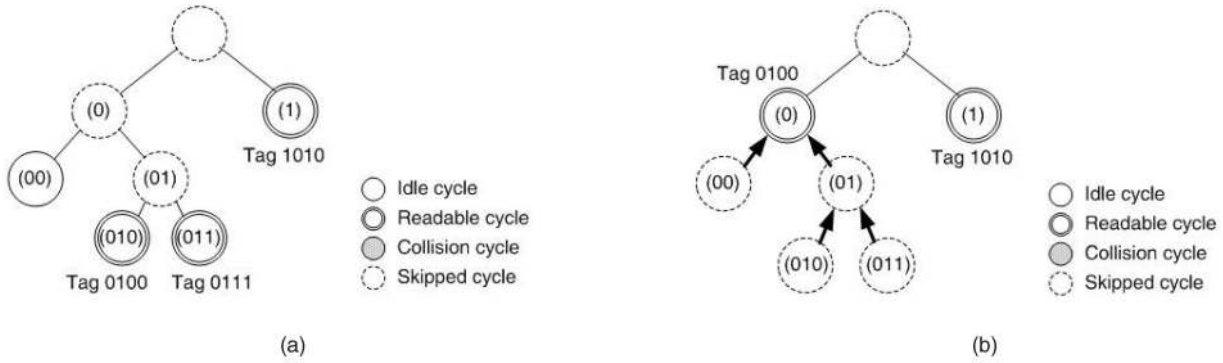

Fig. 5. An example of the query deletion procedure. (a) Tree of the last frame when the reader recognized tag 0100, tag 0111, and tag 1010 under AQS. (b) Query deletion after tag 0111 has reached out of the reader's range.

Theorem 1. *Maintaining CQ at the reader does not need additional memory.*

Proof. Let $S_Q(x)$ and $S_{CQ}(x)$ be the size of Q and CQ, respectively, after transmitting the xth query. Denote $l(x)$ as the number of bits of the xth query. When a new frame starts, $S_{CQ}(0) = 0$. If the interrogation cycle of the xth query is the collision cycle, the reader puts two 1-bit longer queries into Q and $S_{CQ}(x) = S_{CQ}(x-1)$. Otherwise, the reader puts the xth query into CQ.

$$S_Q(x) = S_Q(x-1) - l(x), \tag{1}$$

$$S_{CQ}(x) = S_{CQ}(x-1) + l(x), \tag{2}$$

$$S_Q(x) + S_{CQ}(x) = S_Q(x-1) + S_{CQ}(x-1). \tag{3}$$

$S_{CQ}(x)$ increases, but the total size of Q and CQ is not changed. Therefore, maintaining CQ at the reader does not require additional memory. □

5.2 Query Deletion

In performing the procedure, the query insertion augments the leaf nodes in the tree and increases the length of transmitted queries. Since the number of readable cycles is the same as the number of tags, idle cycles will proliferate and worsen the reader's ability for tag identification. It results from which readable cycles of the last frame are transformed into idle cycles by leaving tags. To address this problem, AQS eliminates unnecessary idle cycles in line 25 in Fig. 4b. Since there is more than one response in the collision cycle, a node of a collision cycle should have two child nodes which are a pair of types as follows:

1. two collision cycles,
2. a collision cycle and a readable cycle,
3. a collision cycle and an idle cycle, and
4. two readable cycles.

If leaving tags transform a pair of child nodes, the reader deletes unnecessary queries from CQ as follows:

- A readable cycle and an idle cycle: Tag a_x can be recognized by $q_1q_2 \dots q_x$ with no collision if only tag a_x responds to query $q_1q_2 \dots q_x0$ (or $q_1q_2 \dots q_x1$) and no tag responds to $q_1q_2 \dots q_x1$ (or $q_1q_2 \dots q_x0$). The reader deletes $q_1q_2 \dots q_x0$ and $q_1q_2 \dots q_x1$ from CQ and puts $q_1q_2 \dots q_x$ into CQ.
- Two idle cycles: There is no tag which responds to $q_1q_2 \dots q_x$ if both $q_1q_2 \dots q_x0$ and $q_1q_2 \dots q_x1$ are queries of idle cycles. Therefore, query $q_1q_2 \dots q_x$ is also the query of the idle cycle. The reader deletes $q_1q_2 \dots q_x0$ and $q_1q_2 \dots q_x1$ from CQ and puts $q_1q_2 \dots q_x$ into CQ.

After a frame, the reader deletes unnecessary queries except two 1-bit queries from CQ. Fig. 5 depicts the query deletion procedure. A number in parentheses indicates the query of that interrogation cycle. In the last frame, the reader recognized three tags of which the IDs are 0100, 0111, and 1010, respectively, as shown in Fig. 5a. Fig. 5b illustrates the query deletion after tag 0111 has reached out of the reader's range. As the query deletion is done under the condition that all branches in the tree are included in CQ, all tags are recognized promptly in the next frame.

Theorem 2. *All tags are recognized by queries of all the leaf nodes in the tree of the last frame.*

Proof. A 1-bit query, 0 (or 1), recognizes all tags of which the first bit of IDs is 0 (or 1) if it is a leaf node. For query $q_1q_2 \dots q_x$, all tags which match $q_1q_2 \dots q_x$ are recognized by $q_1q_2 \dots q_x0$ and $q_1q_2 \dots q_x1$. Therefore, any tag can be recognized by all the leaf nodes in the tree of the last frame. □

6 Adaptive Binary Splitting

AQS reduces collisions, but it produces idle cycles. To guarantee identification of all tags, the reader uses not only queries of readable cycles but also queries of idle cycles of the last frame. Though the reader eliminates unnecessary idle cycles by leaving tags, some idle cycles cannot be avoided in order to cover all possible ranges of the tag ID. On the contrary, ABS starts tag identification from only readable cycles of the last frame and uses random numbers for the splitting procedure. A staying tag revises its counter into the ranking that it was recognized by the reader in the last frame. A transmission of an arriving tag is decided by a random number selected among possible values inside a reader's range. Tag transmissions are aligned in the increasing order of counter values. ABS achieves fast

Algorithm 3. ABS Tag Operation (*ASC*)
/* Transmit the ID with controlling *PSC* and *ASC*
f is the reader's feedback indicating readable, idle or collision */

```
1   Receive the command starting a frame with reader's TSC
2   /* Initialize PSC and ASC */
3   PSC = 0
4   if ASC = NULL or ASC > TSC then
5       ASC = random number from 0 to TSC
6   end if
7   /* Process PSC and ASC for transmission */
8   while PSC <= ASC do
9       if PSC = ASC then
10          Transmit ID
11          Receive feedback f from the reader
12          if f = collision then
13              Select a binary value i randomly
14              ASC = ASC + i
15          else
16              PSC = PSC + 1
17          end if
18      else if PSC < ASC then
19          Receive feedback f from the reader
20          if f = collision then
21              ASC = ASC + 1
22          else if f = readable then
23              PSC = PSC + 1
24          else if f = idle then
25              ASC = ASC – 1
26          end if
27      end if
28  end while
```

(a)

Algorithm 4. ABS Reader Operation (*TSC*)
/* Transmit a feedback according to
the number of received tag responses
f is the flag indicating readable, idle or collision */

```
1   PSC = 0
2   if TSC = NULL then
3       TSC = 0
4   end if
5   Transmit the command starting a frame with TSC
6   while PSC <= TSC do
7       Receive tag responses and detect a collision
8       if tag collision then
9           TSC = TSC + 1
10          f = collision
11      else if only a tag response then
12          Store the tag ID
13          PSC = PSC + 1
14          f = readable
15      else if no tag response then
16          TSC = TSC - 1
17          f = idle
18      end if
19      Transmit feedback f
20  end while
21  Transmit the command terminating a frame
```

(b)

Fig. 6. Pseudocode of ABS. (a) Tag's operation. (b) Reader's operation.

identification by diminishing not only collisions but also unnecessary idle cycles.

6.1 Tag Transmission Control

Fig. 6a shows the pseudocode of the tag in ABS. A tag has two counters, a *Progressed Slot Counter (PSC)* and an

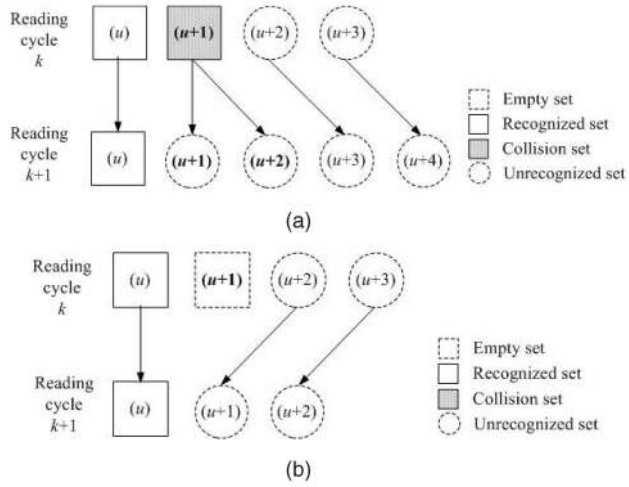

Fig. 7. An example of tag identification at ABS. (a) Operation in the collision cycle. (b) Operation in the idle cycle.

Allocated Slot Counter (ASC). PSC is initialized to 0 at the beginning of the frame and is increased by 1 only in the readable cycle. All the tags have the same value of PSC at all times. ASC signifies the interrogation cycle that the tag can transmit its ID. That is, the tag is allowed to transmit when its ASC is equal to PSC. If the tag has ASC less than PSC, it does not attempt the transmission until the completion of the frame because it has already been recognized in the ongoing frame. To control PSC and ASC, the reader informs tags of the type of the last interrogation cycle by transmitting a feedback. With the aim of splitting a set of tags transmitting at the same interrogation cycle, colliding tags increase ASCs by random numbers. When a tag collision occurs, the tag which has ASC equal to PSC randomly selects one of two binary numbers, 0 and 1, and then adds it to ASC in lines 13 and 14. To prevent the second subset (tags which select 1) from combining with another set (tags which have already had ASC of 1), the tag which has ASC greater than PSC adds 1 to ASC in the collision cycle in line 21. Since PSC is not changed, the first subset (tags which select 0) tries to retransmit in the following interrogation cycle. The second subset transmits after the first subset is recognized. In an idle cycle, the tag which has not been recognized, i.e., the tag which has ASC greater than PSC, decreases ASC by 1 in line 25. Since PSC is not changed, the decrement of ASC gets to pull the schedule of the tag transmission.

At the end of the frame, a recognized tag gets a unique ASC. A tag collision may occur in the next frame if an arriving tag and a staying tag have identical ASCs. The next frame may have an idle cycle if a tag becomes a leaving tag. Fig. 7 depicts the operation in the collision cycle and in the idle cycle under ABS. A square or a circle corresponds to a tag set and a number in parentheses indicates the ASC that tags in that set have. Preserving ASC at the boundary of two consecutive frames makes it possible that the tree search starts from the readable cycles of the last frame.

6.2 Frame Termination

For terminating a frame at once after identifying all tags, the reader of ABS acts as the tag which has the largest ASC. Fig. 6b shows the pseudocode of the reader. The reader

determines the end point of the frame with a *Progressed Slot Counter* (*PSC*) and a *Terminated Slot Counter* (*TSC*). PSC of the reader represents the number of tags recognized successfully. In a readable cycle, the reader adds 1 to PSC. TSC signifies the number of tag sets in the reader's range. If a collision occurs, the reader increases TSC by 1 because the number of tag sets has increased. When an interrogation cycle of type "idle" is encountered, the reader decreases TSC by 1. This is to reflect the effect of the elimination of idle cycles. As soon as PSC is greater than TSC, the reader concludes that all tags have been recognized and transmits the command terminating the frame to all tags. For fast identification in the next frame, the reader preserves TSC after the end of the frame.

In an environment with multiple readers, an arriving tag can be recognized with its ASC given by other readers. If an arriving tag has ASC less than TSC, it is obvious that the arriving tag has the same ASC as a staying tag or a leaving tag. To cope with ASC greater than TSC, the reader supports the TSC value when a frame starts. A tag having ASC greater than TSC changes its ASC to a random number from 0 to TSC. ABS recognizes all tags quickly through scaling the ASCs of arriving tags into the range of TSC.

7 Performance Analysis

In this section, we analyze the total delay for recognizing all tags with AQS and ABS. At first, the identification delay is defined by the number of interrogation cycles.

Definition 1. *Let $A_{r,i}$ denote the set of tags which dwell inside reader r's range in the ith frame of reader r. The identification delay caused by recognizing $A_{r,i}$, $d_{total}(A_{r,i})$, is*

$$d_{total}(A_{r,i}) = \sum (d_{reader} + d_{tag}) \approx T(A_{r,i}) \cdot d_{cycle}, \quad (4)$$

where d_{reader} is the delay of delivering the reader query (or feedback), d_{tag} is the delay of delivering the tag ID, d_{cycle} is the average delay of an interrogation cycle, and $T(A_{r,i})$ is the number of interrogation cycles in a frame when reader r recognizes tags in $A_{r,i}$. The identification delay is determined by $T(A_{r,i})$.

Let a_x denote tag x. Assume that tags have unique b-bit IDs and $A_{r,i}$ has n tags, i.e., $A_{r,i} = \{a_1, a_2, \ldots, a_n\}$. Let $B_{arrive} = \{a_{n+1}, a_{n+2}, \ldots, a_{n+\alpha}\}$ be the set of arriving tags and $B_{leave} = \{a_{f(_1)}, a_{f(_2)}, \ldots, a_{f(\beta)}\}$ $(1 \le f(x) \le n, \beta \le n)$ be the set of leaving tags in the $i+1$th frame of reader r.

7.1 Adaptive Query Splitting

As described in the previous section, AQS uses tag IDs for the splitting procedure. Since AQS adopts a deterministic approach like QT, we analyze the worst case identification delay of AQS through the derivation of the identification delay of QT.

Lemma 1. *Let us define $C_{QT}(A_{r,i})$ as the number of collision cycles in the ith frame when reader r recognizes $A_{r,i}$ under QT. For given $C_{QT}(A_{r,i})$, the identification delay of QT, $T_{QT}(A_{r,i})$, is*

$$T_{QT}(A_{r,i}) = 2C_{QT}(A_{r,i}) + 1. \quad (5)$$

Proof. In the tree of tag identification at QT, only a node of a collision cycle has two child nodes since a set is split into two subsets only in the collision cycle. Therefore, the tree of QT is a full binary tree in which all the intermediate nodes correspond to collision cycles. □

Lemma 2. *For b-bit tag IDs and n tags in $A_{r,i}$,*

$$T_{QT}(A_{r,i}) = n(b + 2 - \log_2 n) - 3. \quad (6)$$

Proof. By Lemma 1, the worst case in the identification delay of QT is that collisions are most numerous. Since tags have unique b-bit IDs, two tag IDs can, at most, equal first $b-1$ bits except the last bit. At this time, two tags make $b-1$ collisions. In the depth k of the tree, a node can be a collision cycle when at least two tags select it for transmission. The maximum number of collision cycles in the depth k of the tree when reader r recognizes n tags under QT, $C_{QT}(n,k)$, is

$$C_{QT}(n,k) = \begin{cases} 2^k & (0 < k \le \log_2 n - 1) \\ n/2 & (\log_2 n - 1 < k < b). \end{cases} \quad (7)$$

The total number of collision cycles is

$$\begin{aligned} C_{QT}(A_{r,i}) &\le \sum_{k=1}^{b-1} C_{QT}(n,k) \\ &= \sum_{k=1}^{\lfloor \log_2 n - 1 \rfloor} 2^k + \sum_{k=\lfloor \log_2 n \rfloor}^{b-1} \frac{n}{2} \\ &\le \frac{n}{2}(b + 2 - \log_2 n) - 2. \end{aligned} \quad (8)$$

From Lemma 1,

$$T_{QT}(A_{r,i}) = 2C_{QT}(A_{r,i}) + 1 \le n(b + 2 - \log_2 n) - 3. \quad (9)$$

□

Theorem 3. *Let $T_{AQS}(A_{r,i+1}|A_{r,i})$ be the number of interrogation cycles in a frame when reader r recognizes $A_{r,i+1}$ under AQS after it has recognized $A_{r,i}$ in the last frame. When $A_{r,i+1} = A_{r,i} + B_{arrive}$,*

$$\begin{aligned} &T_{AQS}(A_{r,i} + B_{arrive} \mid A_{r,i}) \\ &\quad \le (n+\alpha)\{b + 2 - \log_2(n+\alpha)\} - (n+2). \end{aligned} \quad (10)$$

Proof. Suppose that there exist two readers and they recognize $A_{r,i+1}$ under QT and AQS, respectively. Tag identification at QT starts at the l-level nodes in the tree. On the other hand, tag identification at AQS starts from the leaf nodes in the tree of the ith frame. Since $A_{r,i} \subset A_{r,i+1}$, the tree of the ith frame at QT is the part of the tree of the $i+1$th frame at QT. From Lemma 2,

$$\begin{aligned} T_{AQS}(A_{r,i+1}|A_{r,i}) &= T_{QT}(A_{r,i+1}) - C_{QT}(A_{r,i}) \\ &\le (n+\alpha)\{b + 2 - \log_2(n+\alpha)\} - 3 - (n-1) \end{aligned} \quad (11)$$

because the minimum value of $C_{QT}(A_{r,i})$ is $n-1$ when the ith frame has no idle cycle. When there is no leaving tag, AQS reduces at least $n-1$ interrogation cycles compared with QT. □

Theorem 4. *When $A_{r,i+1} = A_{r,i} + B_{arrive} - B_{leave}$,*

$$\begin{aligned} T_{AQS}(A_{r,i+1} \mid A_{r,i}) \le (n+\alpha)\{b + 2 - \log_2(n+\alpha)\} \\ - (n+2). \end{aligned} \quad (12)$$

Proof. AQS is able to eliminate unnecessary idle cycles after the reader has detected nonexistence of tags. However, reader r does not know leaving tags before the completion of the $i+1$th frame. Therefore,

$$T_{AQS}(A_{r,i}+B_{arrive}-B_{leave}\mid A_{r,i}) = T_{AQS}(A_{r,i}+B_{arrive}\mid A_{r,i}). \quad (13)$$

Theorem 4 is derived from (10) and (13). □

7.2 Adaptive Binary Splitting

We analyze the average case in the identification delay of ABS because the performance of ABS is determined by random numbers.

Lemma 3. *The number of collision cycles in the ith frame when reader r recognizes $A_{r,i}$ under BT, $C_{BT}(A_{r,i})$, is*

$$C_{BT}(A_{r,i}) = \sum_{k=0}^{\infty}\left\{2^k-\left(2^k+n-1\right)\left(1-2^{-k}\right)^{n-1}\right\}. \quad (14)$$

Proof. A frame of BT can be represented in a binary tree. Let $E_{BT}(n,k)$, $R_{BT}(n,k)$, and $C_{BT}(n,k)$ be the number of idle cycles, readable cycles, and collision cycles, respectively, in depth k of the binary tree when BT recognizes n tags. Since the number of nodes in depth k of the binary tree is 2^k, $E_{BT}(n,k)=2^k(1-2^{-k})^n$ and $R_{BT}(n,k)=n(1-2^{-k})^{n-1}$.

$$\begin{aligned} C_{BT}(A_{r,i}) &= \sum_{k=0}^{\infty} C_{BT}(n,k) \\ &= \sum_{k=0}^{\infty}\left\{2^k-E_{BT}(n,k)-R_{BT}(n,k)\right\} \\ &= \sum_{k=0}^{\infty}\left\{2^k-2^k\left(1-2^{-k}\right)^n-n\left(1-2^{-k}\right)^{n-1}\right\}. \end{aligned} \quad (15)$$

□

Lemma 4. *The number of interrogation cycles in the ith frame when reader r recognizes $A_{r,i}$ under BT, $T_{BT}(A_{r,i})$, is*

$$T_{BT}(A_{r,i}) = 1+2\sum_{k=0}^{\infty}\left\{2^k-\left(2^k+n-1\right)\left(1-2^{-k}\right)^{n-1}\right\}. \quad (16)$$

Proof. Lemma 1 is also applied into tag identification at BT. As a result, every intermediate node in the binary tree corresponds to the collision cycle and the total number of nodes in the tree is $2C_{BT}(A_{r,i})+1$. From Lemma 3,

$$T_{BT}(A_{r,i}) = 1+2C_{BT}(A_{r,i}) = 1+2\sum_{k=0}^{\infty} C_{binary}(n,k).$$

□

Theorem 5. *Let $T_{ABS}(A_{r,i+1}\mid A_{r,i})$ be the number of interrogation cycles in the $i+1$th frame when reader r recognizes $A_{r,i+1}$ under ABS after it has recognized $A_{r,i}$ in the ith frame. When $A_{r,i+1}=A_{r,i}+B_{arrive}-B_{leave}$,*

TABLE 1
Simulation Setup

Parameter	Value
Simulation Area	100 m × 100 m
The number of readers	100
The identification range of the reader	3 m
The number of tags	1000
Tag ID	Randomly selected 96-bit ID
Maximum Meter Per Frame (MFP)	2 m/frame

$$\begin{aligned} T_{ABS}(A_{r,i+1}\mid A_{r,i}) &= n-\alpha\beta n^{-1}\sum_{k=0}^{\infty}\left(1-2^{-k}\right)^{\alpha/n-1} \\ &+ n\sum_{k=0}^{\infty}\left\{2^{k+1}-\left(2\alpha n^{-1}-2^{-k}+3\right)\left(1-2^{-k}\right)^{\alpha/n}\right\}. \end{aligned} \quad (18)$$

Proof. When there exist n tags in the ith frame, ABS makes n tag sets before the beginning of the $i+1$th frame. Each set has a staying tag or a leaving tag. Among n sets, $n-\beta$ sets include staying tags and β sets include leaving tags. Without loss of generality, assume that arriving tags select a set randomly. Arriving tags in B_{arive} are assigned to n sets uniformly. Since BT and ABS have the same splitting procedure,

$$T_{ABS}(A_{r,i}+B_{arrive}-B_{leave}\mid A_{r+i}) = (n-\beta)\cdot T_{BT}\left(1+\alpha n^{-1}\right)+\beta\cdot T_{BT}\left(\alpha n^{-1}\right). \quad (19)$$

Theorem 5 is derived from (16) and (19). □

8 Performance Evaluation

We evaluate the performance of AQS and ABS compared to BT and QT. To measure the efficiency of tag identification in the tree-based protocols, we consider the following aspects:

- Number of collisions: We measure the number of collisions between tag-to-reader signals. A collision defers identification and increases power consumption of tags.
- Number of idle cycles: The idle cycle is a factor of identification delay.
- Identification delay: We measure the total delay for recognizing all tags by the interrogation cycle. Fast identification is the most significant factor in the tree-based anticollision protocols because they do not cause the tag starvation problem.
- Tag communication overhead: This metric is the average number of bits transmitted by a tag in a frame. This influences the amount of power consumption. Due to lack of power source in tags, this must be low.

8.1 Simulation Setup

The simulation setup is shown in Table 1. To avoid the reader collision problem [3], we place readers in such a

TABLE 2
Simulation Scenarios

	U	A_i	r_s	r_a
Scenario I	100	50	0 ~ 1	0
Scenario II	100	50	0 ~ 1	0.5
Scenario III	100	50	0	0 ~ 1
Scenario IV	100	50	0.5	0 ~ 1

manner that their reading ranges do not intersect. To appreciate the impact of tag's mobility, we define Meter Per Frame (MPF). The MPF of tag a_x, $MPF(a_x)$, is given by

$$MPF(a_x) = \frac{m_a(t_1, t_2)}{F_p(t_1, t_2)} (\text{m/frame}), \tag{20}$$

where $m_a(t_1, t_2)$ is the distance that tag a_x moves in the time interval $[t_1, t_2]$ and $F_p(t_1, t_2)$ is the number of frames executed by protocol p in the interval $[t_1, t_2]$. By using MPF, we can ensure that tree-based protocols recognize the same tags in a frame. In our simulations, initial positions and destinations of tags are randomly selected under the simulation area. A tag moves from its initial position toward its destination with MPF, which is randomly selected from 0 to the maximum MPF. We run each simulation 50 times under the above parameters and investigate the average results for the performance evaluation.

8.2 Impact of Staying Tags and Arriving Tags

First, we simulate tree-based protocols while varying the number of staying tags and arriving tags. Let U be the set of all tags in the simulation area and let A_i be the set of tags which are recognized in the ith frame. We measure the performance in the $i+1$th frame by changing the ratio of staying tags to A_i, denoted by r_s, and the ratio of arriving tags to $U - A_i$, denoted by r_a. We consider four scenarios according to the values of r_s and r_a as shown Table 2.

Fig. 8, Fig. 9, and Table 3 depict the simulation results about the four scenarios. In the first scenario, there is no arriving tag. From Fig. 8a and Table 3, we can see that AQS and ABS achieve collisionless tag identification. They can block collisions between staying tags completely because they do not allocate more than one staying tag to a set. On the other hand, as the ratio of staying tags increases, BT and QT make tag collisions more frequently. Fig. 8b illustrates that the identification delay of ABS is the shortest of all protocols at high ratios of staying tags, while it is slower than BT and QT at ratios less than 0.3 because of idle cycles produced by leaving tags. AQS has longer delay than ABS because the reader transmits additional queries for covering all possible ranges of the ID. The second scenario is tag identification with increasing the fraction of staying tags when some arriving tags exist. Though AQS and ABS perfectly eliminate collisions between staying tags, they cannot block collisions by arriving tags as shown in Fig. 8d and Fig. 8e. As the fraction of staying tags increases, collisions between staying tags overwhelm collisions by arriving tags and the superiority of identification delay of AQS and ABS becomes stronger.

Fig. 9a, Fig. 9b, and Fig. 9c show the results when the ratio of arriving tags increases without staying tags. Though tag identification of AQS and ABS lead to fewer collisions, idle cycles in AQS and ABS cause delay. Since collisions between staying tags do not occur, the increment of idle cycles is more than the decrement of collisions at low ratios of arriving tags. In the fourth scenario, we increase the fraction of arriving tags while setting some staying tags. The results in Fig. 9d and Fig. 9e show faster identification of

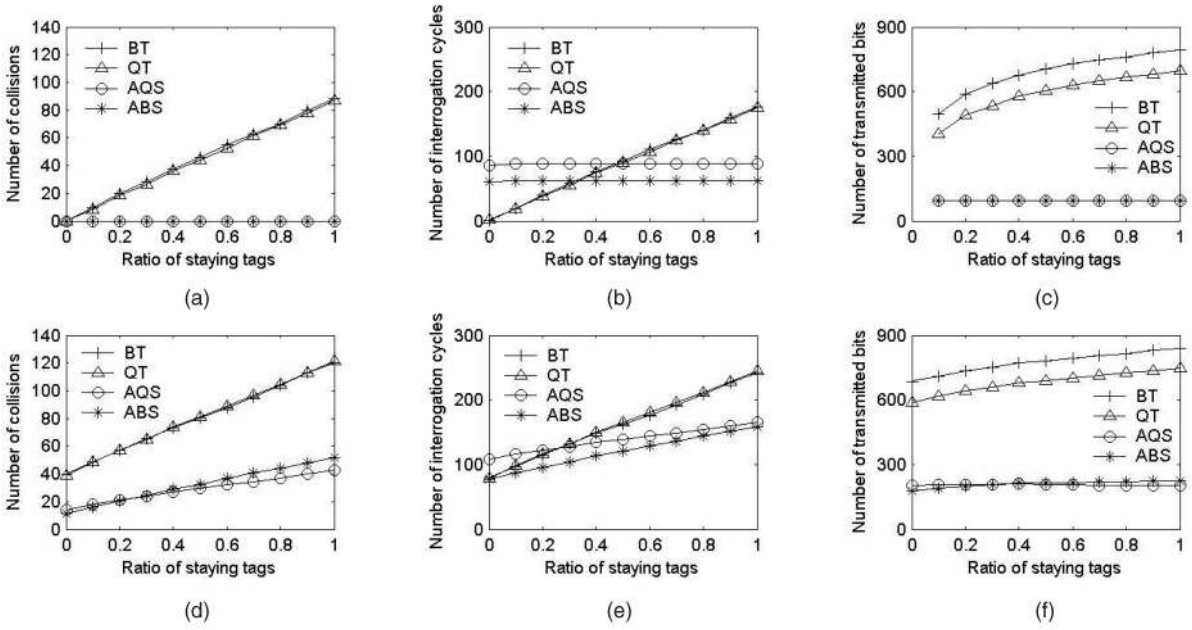

Fig. 8. The simulation results according to the values of r_s. (a) Collisions $(r_a = 0)$. (b) Identification delay $(r_a = 0)$. (c) Tag communication overhead $(r_a = 0)$. (d) Collisions $(r_a = 0.5)$. (e) Identification delay $(r_a = 0.5)$. (f) Tag communication overhead $(r_a = 0.5)$.

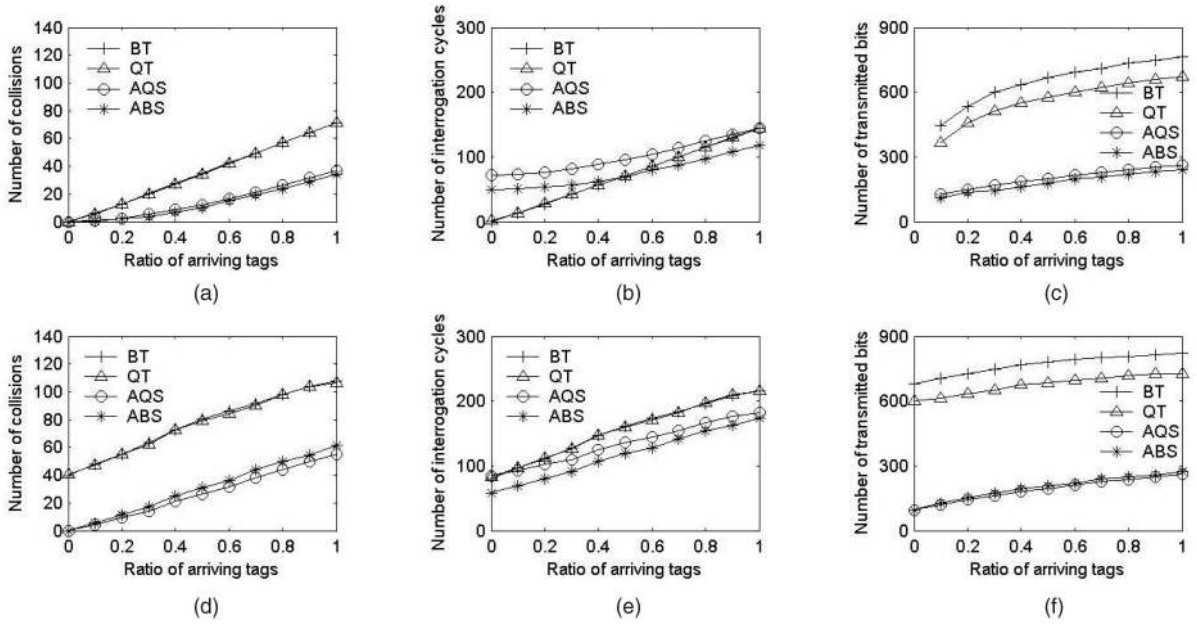

Fig. 9. The simulation results according to the values of r_a. (a) Collisions $(r_s = 0)$. (b) Identification delay $(r_s = 0)$. (c) Tag communication overhead $(r_s = 0)$. (d) Collisions $(r_s = 0.5)$. (e) Identification delay $(r_s = 0.5)$. (f) Tag communication overhead $(r_s = 0.5)$.

AQS and ABS than BT and QT in spite of additional idle cycles created by leaving tags. Additionally, the decrement of collisions involves low communication overhead at a tag in all the scenarios.

8.3 Impact of the Number of Tags

Fig. 10 shows the simulation results obtained by changing the number of tags in the simulation area. In the readers' reading ranges, there averagely exist 31.62 percent of tags. As the number of tags increases, the identification delay gets longer and tag collisions occur more often. BT and QT show comparable delay curves. The subtle difference in the delay results from the starting points of tag identification. Note that BT departs from the root of the tree and QT departs from the 1-level nodes of the tree. By restraining the occurrence of collisions, AQS and ABS have shorter delay than BT and QT. Small collisions activate small tag communication overhead. ABS has the shortest delay because it eliminates many idle cycles. AQS generates more idle cycles than others due to additional queries in order to guarantee recognizing all tags. On the other hand, AQS makes fewer collisions and less tag communication than ABS because idle cycles prevent arriving tags from colliding with other tags.

TABLE 3
Average Number of Timeslots According to the Ratio of Staying Tags and the Ratio of Arriving Tags

	Protocol	Collision cycle	Idle cycle	Total cycle
Scenario I	Binary tree	45.06	14.02	91.12
	Query tree	43.76	13.72	89.53
	AQS	0	56.51	88.55
	ABS	0	29.81	61.85
Scenario II	Binary tree	80.40	24.89	161.81
	Query tree	80.78	26.26	163.55
	AQS	28.96	52.69	138.16
	ABS	32.15	31.18	119.84
Scenario III	Binary tree	35.14	10.99	71.28
	Query tree	34.86	11.71	71.72
	AQS	14.75	60.68	100.58
	ABS	13.15	37.49	75.79
Scenario IV	Binary tree	76.84	23.67	154.69
	Query tree	76.35	24.17	154.71
	AQS	26.79	52.94	133.91
	ABS	30.65	32.04	116.88

8.4 Impact of Tag Movement

We evaluate the performance by increasing the tag velocity. Fig. 11 presents the simulation results obtained by varying the maximum MPF. We normalize the measured values by the number of recognized tags. When tags move at low speeds, AQS and ABS outperform BT and QT considerably. This is because tags would like to be staying tags and AQS and ABS eliminate collisions between staying tags perfectly. Especially, there is no collision when the maximum MPF is 0. As tags move faster, the performance of AQS and ABS deteriorates. When the maximum MPF is more than 6 m/frame, AQS has longer delay than QT. At a high speed, there are few staying tags and collisions between staying tags at BT and QT hardly occur. Additionally, AQS and ABS generate idle cycles because leaving tags increase and leaving tags make idle cycles. Hence, AQS and ABS show the performance similar to BT and QT at high speed.

8.5 Impact of the Similarity of ID

For the purpose of another comparison, we evaluate the impact of the similarity among IDs. QT and AQS may be influenced by the distribution of IDs because they use tag ID

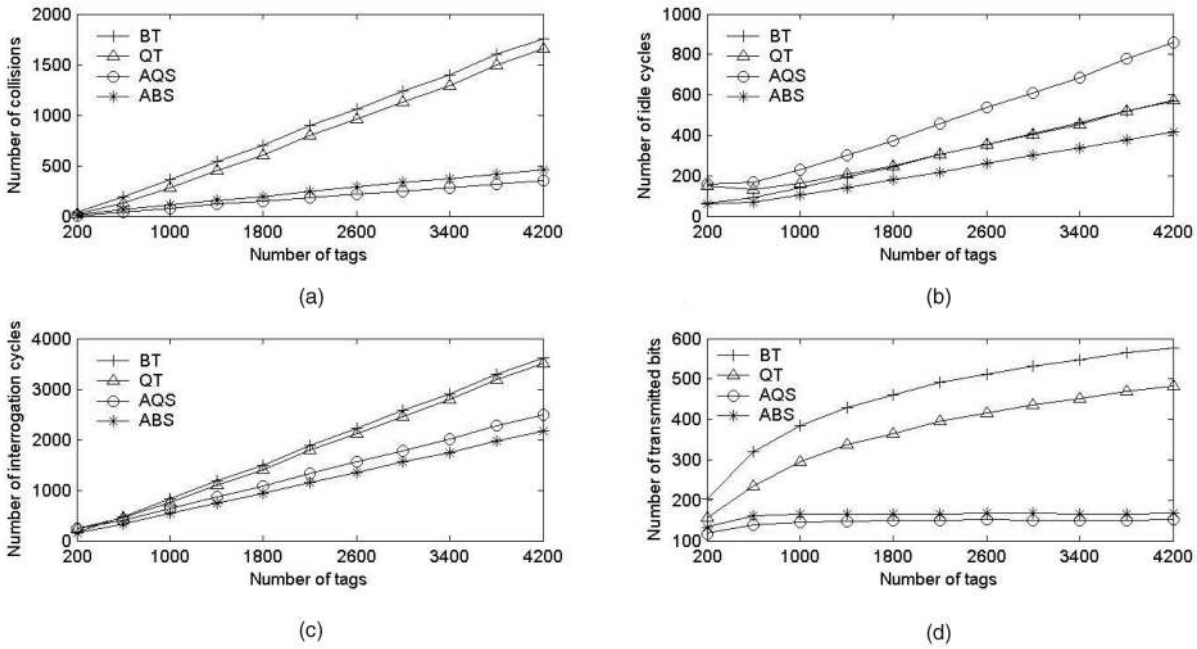

Fig. 10. Performance comparison with varying the number of tags. (a) Collisions. (b) Idle cycles. (c) Identification delay. (d) Tag communication overhead.

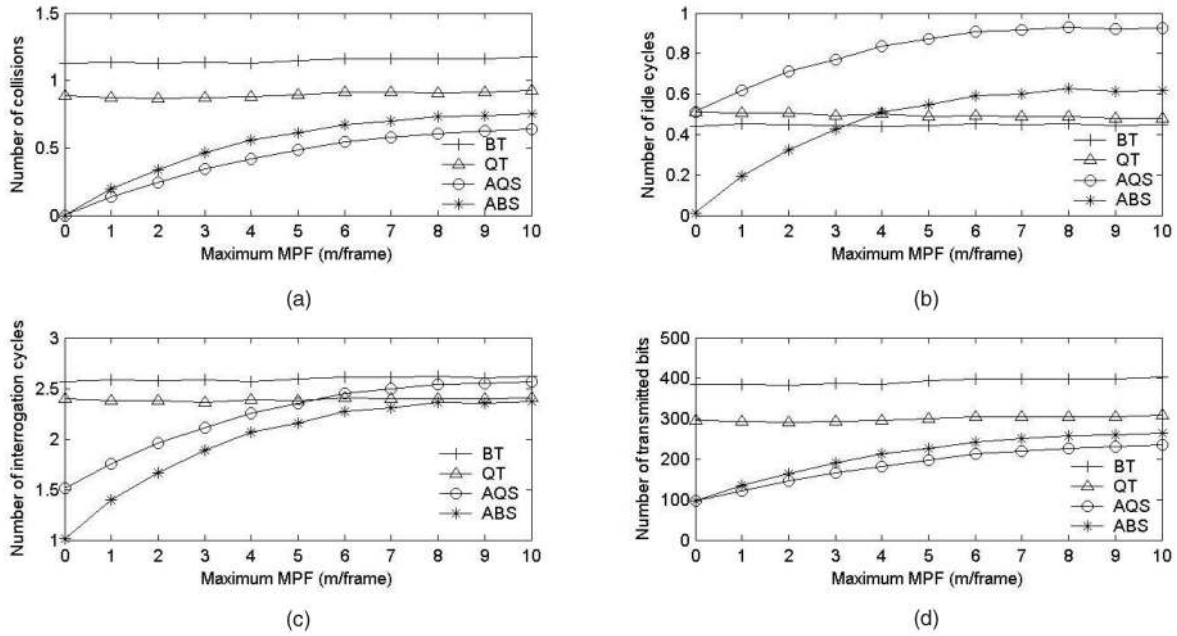

Fig. 11. Impact of tag mobility. (a) Collisions. (b) Idle cycles. (c) Identification delay. (d) Tag communication overhead.

for splitting a tag set. To quantify the similarity of IDs, we define an *identical bit* as the length of the identical prefix all tag IDs have. The tag ID is depicted by $x_1x_2 \ldots x_a x_{a+1} \ldots x_{96}$ (x_i is a binary digit, $1 \leq a < 96$) and all tag IDs have the same $x_1x_2 \ldots x_a$ if the identical bit is a and each tag has a 96-bit ID. Fig. 12 gives the simulation results for various identical bits from 0 (IDs are completely randomly selected) to 80. We normalize the measured values by the number of recognized tags. As the identical bit increases, QT rapidly degenerates as expected. QT has the highest communication overhead because the reader transmits all queries causing collisions in every frame. On the other hand, the performance of AQS is not seriously affected by the similarity of IDs. Since candidate queue CQ excludes queries of collision cycles of the last frame, AQS uses a collision query only once. However, as the identical bit increases, the tree of QT and AQS has more leaf nodes and AQS generates more idle cycles. When the identical bit is greater than 48, AQS has longer delay than BT because of an increment of idle cycles. ABS and BT are not affected by the identical bit because they do not use the patterns of IDs. As in the previous scenario, ABS shows the shortest identical delay.

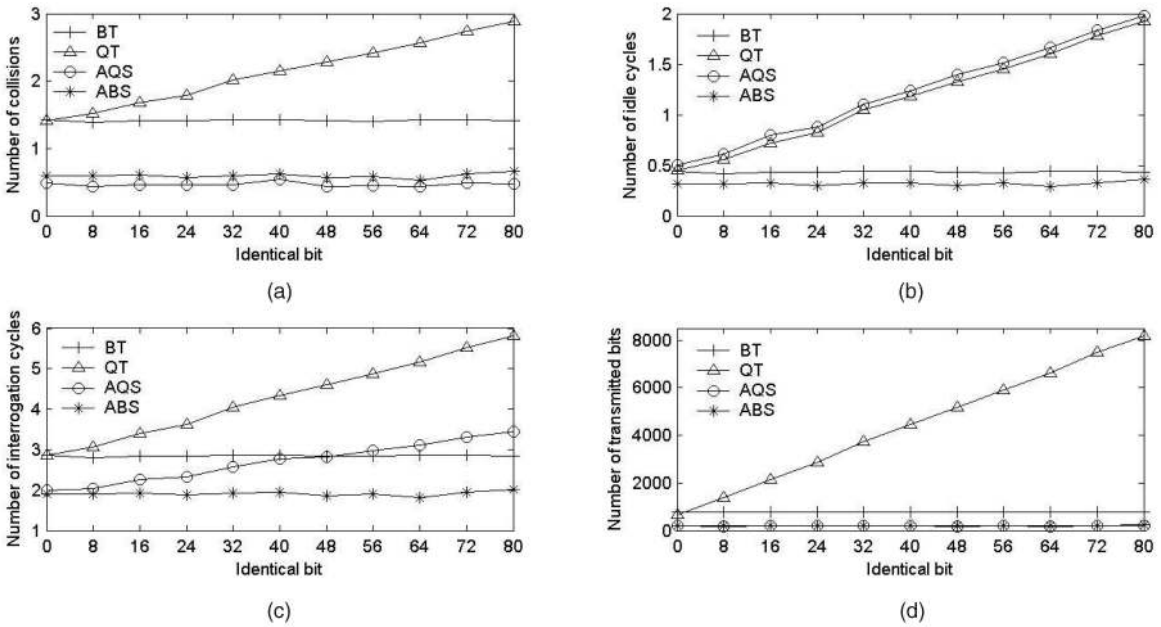

Fig. 12. Impact of ID distribution. (a) Collisions. (b) Idle cycles. (c) Identification delay. (d) Tag communication overhead.

9 Conclusion

A collision caused by tags transmitting simultaneously is a major factor in deferring tag identification of RFID systems. In this paper, adaptive tag anticollision protocols for passive tags have been proposed and evaluated. We develop novel and enhanced tree-based protocols to reduce collisions by exploiting information obtained from the last process of tag identification. The key institution behind our proposed approach is that, in most applications employing RFID tags, the set of objects encountered in successive readings from a particular reader does not change substantially and information from one reading can be used for the next. A simulation-based evaluation shows that AQS and ABS significantly reduce delay and communication overhead for the tag reading process.

Acknowledgments

This research was supported by the Ministry of Information and Communication, Korea, under the ITRC program supervised by IITA, IITA-2005-(C1090-0501-0019).

References

[1] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification.* John Wiley & Sons, 2003.
[2] S. Sarma, D. Brock, and D. Engels, "Radio Frequency Identification and the Electronic Product Code," *IEEE Micro,* vol. 21, no. 6, pp. 50-54, Nov. 2001.
[3] S. Sarma, S. Weis, and D. Engels, "RFID Systems and Security and Privacy Implications," *Proc. Workshop Cryptographic Hardware in Embedded Systems,* pp. 454-470, Aug. 2002.
[4] D. Engels and S. Sarma, "The Reader Collision Problem," Technical Report MIT-AUTOID-WH007, Auto-ID Center, Nov. 2001.
[5] J. Waldrop, D. Engels, and S. Sarma, "Colorwave: An Anticollision Algorithm for the Reader Collision," *Proc. IEEE Int'l Conf. Comm.,* pp. 1206-1210, May 2003.
[6] C. Floerkemeier and M. Lampe, "Issues with RFID Usage in Ubiquitous Computing Applications," *Proc. Second Int'l Conf. Pervasive Computing,* pp. 188-193, Apr. 2004.
[7] N. Abramson, "The Aloha System—Another Alternative for Computer Communications," *Proc. Fall Joint Computer Conf., Am. Federation of Information Processing Soc. Conf.,* vol. 37, pp. 281-285, Nov. 1970.
[8] R. Metcalfe, "Steady State Analysis of a Slotted and Controlled Aloha System with Blocking," *Proc. Sixth Hawaii Conf. System Science,* pp. 375-380, Jan. 1973.
[9] S. Lam and L. Kleinrock, "Packet Switching in a Multi Access Broadcast Channel: Dynamic Control Procedures," *IEEE Trans. Comm.,* vol. 23, no. 9, pp. 891-904, Sept. 1975.
[10] R. Rao and A. Ephremides, "On the Stability of Interacting Queues in a Multiple-Access System," *IEEE Trans. Information Theory,* vol. 34, no. 5, pp. 918-930, Sept. 1988.
[11] V. Anatharam, "The Stability Region of the Finite-User Slotted ALOHA Protocol," *IEEE Trans. Information Theory,* vol. 37, no. 3, pp. 535-540, May 1991.
[12] I. Teletar and R. Gallager, "Combining Queuing Theory and Information Theory for Multiaccess," *IEEE J. Selected Areas in Comm.,* vol. 13, no. 6, pp. 963-969, Aug. 1995.
[13] F. Schoute, "Dynamic Frame Length ALOHA," *IEEE Trans. Comm.,* vol. 31, no. 4, pp. 565-568, Apr. 1983.
[14] J. Wieselthier, A. Ephremides, and L. Michaels, "An Exact Analysis and Performance Evaluation of Framed ALOHA with Capture," *IEEE Trans. Comm.,* vol. 38, no. 2, pp. 125-137, Feb. 1989.
[15] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. Int'l Conf. Pervasive Computing,* pp. 98-113, Apr. 2002.
[16] J. Zhai and G. Wang, "An Anti-Collision Algorithm Using Two-Functioned Estimation for RFID Tags," *Proc. Int'l Conf. Computational Science and Its Applications,* pp. 702-711, May 2005.
[17] *EPC Radio-Frequency Identification Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.0.8,* EPCglobal, Dec. 2004.
[18] *Information Technology Automatic Identification and Data Capture Techniques—Radio Frequency Identification for Item Management Air Interface—Part 6: Parameters for Air Interface Communications at 860-960 MHz,* Int'l Standard ISO 18000-6, Nov. 2003.
[19] Philips Semiconductors, *UCODE,* http://www.semiconductors.philips.com, 2005.
[20] D. Hush and C. Wood, "Analysis of Tree Algorithms for RFID Arbitration," *Proc. IEEE Int'l Symp. Information Theory,* p. 107, Aug. 1998.
[21] M. Jacomet, A. Ehrsam, and U. Gehrig, "Contactless Identification Device with Anticollision Algorithm," *Proc. IEEE Conf. Circuits, System, Computers and Comm.,* pp. 269-273, July 1999.

[22] "Draft Protocol Specification for a 900 MHz Class 0 Radio Frequency Identification Tag," Auto-ID Center, Feb. 2003.
[23] S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," *Proc. First Ann. Conf. Security in Pervasive Computing*, pp. 201-212, Mar. 2003.
[24] C. Law, K. Lee, and K. Siu, "Efficient Memoryless Protocol for Tag Identification," *Proc. Fourth Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm.*, pp. 75-84, Aug. 2000.
[25] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min, "Evaluating and Optimizing Power Consumption of Anti-Collision Protocols for Applications in RFID Systems," *Proc. Int'l Symp. Low Power Electronics and Design*, pp. 357-362, Aug. 2004.
[26] J. Capetanakis, "Tree Algorithms for Packet Broadcast Channels," *IEEE Trans. Information Theory*, vol. 25, no. 5, pp. 505-515, Sept. 1979.
[27] J. Moseley and P. Humblet, "A Class of Efficient Contention Resolution Algorithms for Multiple Access Channels," *IEEE Trans. Comm.*, vol. 33, no. 2, pp. 145-151, Feb. 1985.
[28] J. Myung and W. Lee, "An Adaptive Memoryless Tag Anti-Collision Protocol for RFID Networks," *Proc. IEEE INFOCOM*, Mar. 2005.

Jihoon Myung received the BS degree in computer science and engineering from Korea University, Seoul, Korea, in 2004. He is currently working toward the PhD degree in computer science and engineering from Korea University, Seoul, Korea. His research interests include RFID anticollision, wireless sensor networking, and ad hoc networking. He is a student member of the IEEE.

Wonjun Lee received the BS and MS degrees in computer engineering from Seoul National University (SNU), Seoul, Korea, in 1989 and 1991, respectively. He also received the MS degree in computer science from the University of Maryland, College Park, in 1996 and the PhD degree in computer science and engineering from the University of Minnesota, Minneapolis, in 1999. He is an associate professor in the Department of Computer Science and Engineering at Korea University, Seoul, Korea. His research interests include mobile wireless communication protocols and architectures, wireless sensor networking, wireless mesh network protocols, and Internet architecture technology. He is a senior member of the IEEE.

Jaideep Srivastava received the BS degree from IIT Kanpur, India, and the PhD degree from the University of California, Berkeley. He is a professor of computer science and engineering at the University of Minnesota. He has authored/coauthored more than 185 papers in journals and conferences. He has provided technology and technology strategy advice to a number of large corporations, including Cargill, United Technologies, IBM, Honeywell, 3M, and Persistent Systems. Dr. Srivastava is currently on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems* (TPDS), the *VLDB Journal*, the *Knowledge and Information Systems Journal* (KAIS), and the *World-Wide Web Journal*. He has served on the editorial board of the *IEEE Transactions on Knowledge and Data Engineering* (TKDE). He has been elected a fellow of the IEEE and has been appointed a Distinguished Visitor by the IEEE Computer Society.

Timothy K. Shih is a professor of the Department of Computer Science and Information Engineering at Tamkang University, Taiwan. His current research interests include multimedia computing and distance learning. He is the founder and co-editor-in-chief of the *International Journal of Distance Education Technologies* published by Idea Group Publishing. Dr. Shih is the associate editor of the *ACM Transactions on Internet Technology*. He was also the associate editor of the *IEEE Transactions on Multimedia*. He is a member of the ACM, a senior member of IEEE, and a member of the Educational Activities Board of the IEEE Computer Society. Information regarding his publications, demonstrations, and contact address can be retrieved from http://www.mine.tku.edu.tw/chinese/teacher/tshih.htm.