Improved Bit-by-bit Binary Tree Algorithm in Ubiquitous ID System

Ho-Seung Choi, Jae-Ryong Cha, and Jae-Hyun Kim

School of Electrical and Computer Engineering,
AJOU University, San 5 Wonchon-Dong, Youngtong-Gu, Suwon 442-749, Korea
{lastjoin, builder, and jkim}@ajou.ac.kr

Abstract. We propose and analyze the fast wireless anti-collision algorithm(Improved Bit-by-bit Binary-Tree algorithm(IBBT)) in Ubiquitous ID system. We mathematically compare the performance of the proposed algorithm with that of Binary Search algorithm(BS), Slotted Binary-Tree algorithm(SBT) using time slot, and Bit-by-bit Binary-Tree algorithm(BBT). We also validated analytic results using simulation. According to the analytical result, comparing IBBT with BBT which is the best of existing algorithms, the performance of IBBT is about 304% higher when the number of the tags is 20, and 839% higher for 200 tags.

1 Introduction

In a world of ubiquitous computing, the object identification is most useful for applications such as asset tracking(e.g. libraries, animals), automated inventory and stock-keeping, toll collecting, and similar tasks where physical objects are involved. The RFID(Radio Frequency Identification) systems are a simple form of ubiquitous sensor networks that are used to identify physical objects. In this paper, we call RFID system as Ubiquitous ID(*u*-ID) system. Instead of sensing environmental conditions, the *u*-ID system identifies the unique tags' ID or detailed information saved in them attached to objects. Passive *u*-ID systems generally consist of three components - a reader, passive tags, and a controller. A reader interrogates tags for their ID or detailed information through an RF communication link, and contains internal storage, processing power, and so on. Tags get processing power through RF communication link from the reader and use this energy to power any on-tag computations and to communicate to the reader. A reader in *u*-ID system broadcasts the request message to the tags. Upon receiving the message, all tags send the response back to the reader. If only one tag responds, the reader receives just one response. If there are more than one tag response, their responses will collide on the RF communication channel, and thus cannot be received by the reader[1]. The problem of resolving this collision is referred to as the Anti-Collision Problem, and the ability to resolve this problem is crucial in *u*-ID system performance. In *u*-ID system, important measures of performance include the time required to identify the tags, and the power consumed by the tags. If the data from the tags are small, we can reduce the time to identify the tags and the power consumed by the tags. This paper mathematically compares the performance of the proposed algorithm with that of Binary Search algorithm(BS), Slotted Binary-Tree algorithm(SBT), and Bit-by-bit Binary Tree algorithm(BBT). We also validate analytical results using OPNET simulation.

2 Existing binary anti-collision algorithms

2.1 Binary Search algorithm(BS) [2]

BS resolves the collision by reducing gradually collided bit in a tag ID. When a collision is occurred, the reader knows the position in which the collision occurs. If there are tags whose first collided bit is 1, they do not respond to the reader's next request. But tags whose first collided bit is 0 transfer their ID to the reader's next request. By repeating this procedure, the reader can identify all the tags. More details are in [2]. Assuming that there are tags, the number of iterations of BS(I_{BS}) is

$$I_{BS} = \frac{\log(n)}{\log(2)} + 1. \tag{1}$$

2.2 Slotted Binary Tree algorithm(SBT) [3]

According to this algorithm when a collision occurs, in slot i, all tags that are not involved in the collision wait until the collision is resolved. The tags involved in the collision split randomly into two groups, by(for instance) each selecting 0 or 1. The tags in the first group, those that selected 0, retransmit in slot i+1 while those that selected 1 wait until all tags that selected 0 successfully transmit their ID. If slot i+1 is either idle or contains a successful transmission, the tags of the second group, those that selected 1, retransmit in slot i+2. If slot i+1 contains another collision, the procedure is repeated[3]. Assuming that there are n tags, the number of iterations of SBT(I_{SBT}) is

$$I_{SBT} = 1 + \sum_{k=2}^{n} \binom{n}{k} \frac{2\,(k-1)\,(-1)^k}{\left[1 - p^k - (1-p)^k\right]}. \tag{2}$$

2.3 Bit-by-bit Binary Tree algorithm(BBT) [4],[5]

In BBT, if the reader requests ID bit to the tags, all the tags transfer 0 or 1 out of their ID bit. If not colliding, the reader saves the received bit in the memory before it requests next bit. But if colliding, the reader selects one group out of two groups according to the algorithm, then requests next bit. The procedure is as follows. If the reader requests k^{th} bit from the tags, the tags transfer k^{th} bit.(In this algorithm, the initial value of k is 1). If the reader receives k^{th} bit without collision, it saves k^{th} bit in the memory. But if colliding, the reader saves k^{th} bit as 0 in the memory. The reader then makes all the tags whose k^{th} bit is

1 inactive. The tags in inactive state do not temporarily transfer their ID bits until one tag is identified. The tags in inactive state do not temporarily transfer their ID bits until one tag is identified. The reader repeats this procedure until all the bits of an ID are received. Assuming that there are n tags and the length of tag ID is j bits, then the number of iterations of BBT(I_{BBT}) is

$$I_{BBT} = n \times j. \tag{3}$$

Table 1. Used tags' ID

Tag1	0001
Tag2	0010
Tag3	1010
Tag4	1011

Fig.1 shows the process to identify four tags in Table 1. In Fig.1, 'X' means collision and the number of total transferred bits from the tags to identify four tags is 16(4×4 bits).

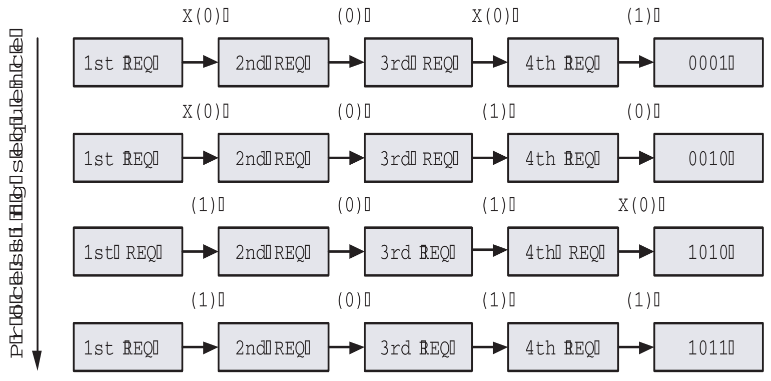

Fig. 1. Bit-by-bit binary-tree algorithm

3 The Improved Bit-by-bit Binary-Tree algorithm(IBBT)

In BBT, the reader always requests all the bits of tags' ID, so it takes much time to identify all the tags. To resolve this problem, we propose IBBT. IBBT procedure is as follows. First of all, the reader requests all the bits of tags' ID. When the reader receives k^{th} bit of tags' ID and if all the bits are 0(1), the reader saves k^{th} bit as 0(1). But if the collision occurs, the reader saves k^{th} bit as X in the memory. After the reader receives all the bits of tags' ID, the reader knows which bits are collided. The reader sequentially re-requests only collided bits to the tags by the method of bit-by-bit. If there are two tags whose ID except for the last bit is identical, the last bit transferred from tags will definitely collide. In this case, we do not need to request next bit in IBBT. Accordingly, we can identify two tags simultaneously. As a result, the number of iterations of IBBT is less than that of BBT. If the tags' ID is sequential, the number of collided bits reduces. Therefore, the performance of IBBT is better than that of BBT.

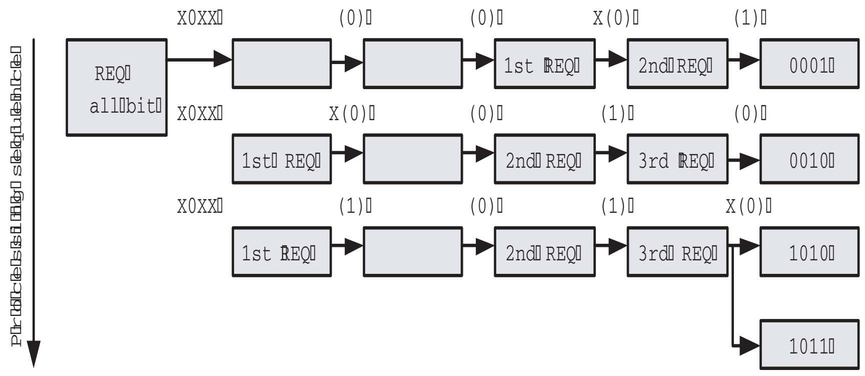

Fig. 2. The proposed IBBT algorithm

Fig.2 shows the process to identify four tags in Table 1 using IBBT. In this example two tags, whose IDs are '1010' and '1011' respectively, have identical ID except the last bit. When the reader requests these tags' last bit, the collision occurs. However the reader can identify two tags simultaneously without further request. In Fig.2, the number of total transferred bits from the tags to identify four tags is 12.

4 Performance analysis

In this section, we analyze the performance of IBBT from two points of view. One is the number of iterations and the other is total transferred bits from the tags.

4.1 The number of iterations

We assume that arbitrary m tags are used out of n tags whose ID is sequential, and the length of ID is 36 bits[5]. Let Π be the number of iterations which does not consider the tags whose last ID bit collides. Then Π is driven by (4).

$$\begin{aligned}
\Pi &= \left(\frac{2^r}{n} + m + 1\right)(r+1) + \frac{m\left(1-\frac{2^r}{n}\right)}{2} r + \frac{m\left(1-\frac{2^r}{n}\right)}{4}(r-1) + \cdots \\
&\quad + \frac{m\left(1-\frac{2^r}{n}\right)}{2^{k_{\max}}}(r+1-k_{\max}) + \left(\frac{m\left(1-\frac{2^r}{n}\right)}{2^{k_{\max}}} - 1\right)(r-k_{\max}) \\
&= \sum_{k=1}^{\log_2 m\left(1-\frac{2^r}{n}\right)} \frac{m\left(1-\frac{2^r}{n}\right)}{2^k}(r+1-k) + \left(\frac{2^r}{n} \times m + 1\right)(r+1) \\
&\quad + \left(\frac{m\left(1-\frac{2^r}{n}\right)}{2^{k_{\max}}} - 1\right)(r-k_{\max}) \qquad (4)
\end{aligned}$$

When $k_{\max} = \left[\log_2 m\left(1-\frac{2^r}{n}\right)\right]$, $2^r < n \leq 2^{(r+1)}$, and $0 < r \leq 35$(r is a integer). And $[*]$ is a maximum integer less than or equal to $*$. We can calculate the number of iterations of IBBT by multiply Π by the term which considers the tags whose last ID bit is collided. When the number of used tags(m) is even, the number of iterations of IBBT(I_{EBBT}) is calculated by (5).

$$I_{IBBT} = \begin{cases} \Pi \times \frac{1}{m} \sum\limits_{k=0}^{\frac{m}{2}} \dfrac{\binom{\frac{n}{2}}{k}\binom{\frac{n}{2}-k}{m-2k} 2^{(m-2k)}}{\binom{n}{m}} \cdot (m-k), & 0 < m \leq \frac{n}{2} \\ \Pi \times \frac{1}{m} \sum\limits_{k=m-\frac{n}{2}}^{\frac{m}{2}} \dfrac{\binom{\frac{n}{2}}{k}\binom{\frac{n}{2}-k}{k-m+\frac{n}{2}} 2^{(m-2k)}}{\binom{n}{m}} \cdot (m-k), & \frac{n}{2} < m \leq n \end{cases} \qquad (5)$$

Otherwise, the number of used tags(m) is odd, the number of iterations of IBBT(I_{IBBT}) is calculated by (6).

$$I_{IBBT} = \begin{cases} \Pi \times \frac{1}{m} \sum\limits_{k=0}^{\frac{m-1}{2}} \dfrac{\binom{\frac{n}{2}}{k}\binom{\frac{n}{2}-k}{m-2k} 2^{(m-2k)}}{\binom{n}{m}} \cdot (m-k), & 0 < m < \frac{n}{2} \\ \Pi \times \frac{1}{m} \sum\limits_{k=m-\frac{n}{2}}^{\frac{m-1}{2}} \dfrac{\binom{\frac{n}{2}}{k}\binom{\frac{n}{2}-k}{k-m+\frac{n}{2}} 2^{(m-2k)}}{\binom{n}{m}} \cdot (m-k), & \frac{n}{2} < m < n \end{cases} \qquad (6)$$

4.2 Total transferred bits from the tags

Let I be the number of iterations for each algorithm and B_I be the transferred bits from the tags in each iteration, then the number of total transferred bits(B_{total}) from the tags is

$$B_{total} = I \cdot B_I. \tag{7}$$

In IBBT, when the reader initially requests all ID bits from the tags, the tags send all ID bits to the reader. So, the number of total transferred bits from the tags should be added by 35(the length of ID−1).

5 Numerical and simulation results

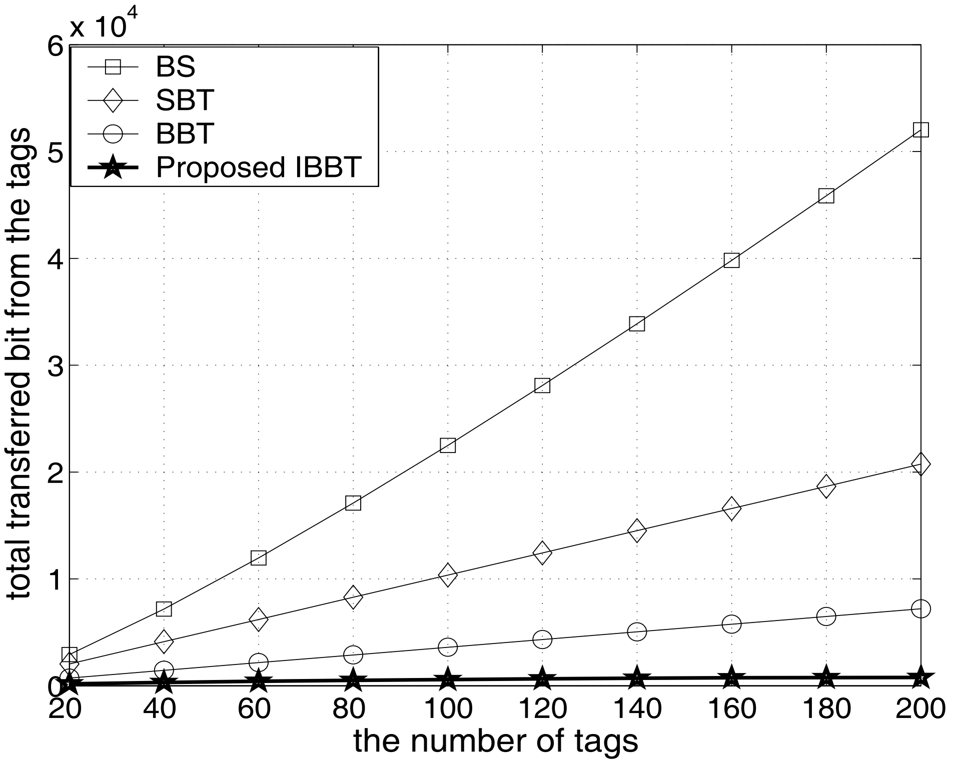

Fig. 3. Total transferred bits from the tags vs. the number of the tags

Fig.3 shows total transferred bits from the tags for the number of tags in each algorithm and represents the analytic results. In proposed algorithm, all tags transfer one bit whenever the reader requests ID from the tags. But all other algorithms except for BBT transfer all the bits of tags' ID. So, the proposed algorithm reduces the time to identify tag and the energy consumed by the tag, because the number of transferred bits from the tags is smaller compared to the existing algorithms. In Fig.3, to identify 100 tags, BS, SBT, and BBT transfer 22491, 9482, and 3600 bits respectively. For the same scenario, IBBT only transfers 578 bits. Therefore, we found that proposed algorithm has very higher performance compared to the existing algorithms.

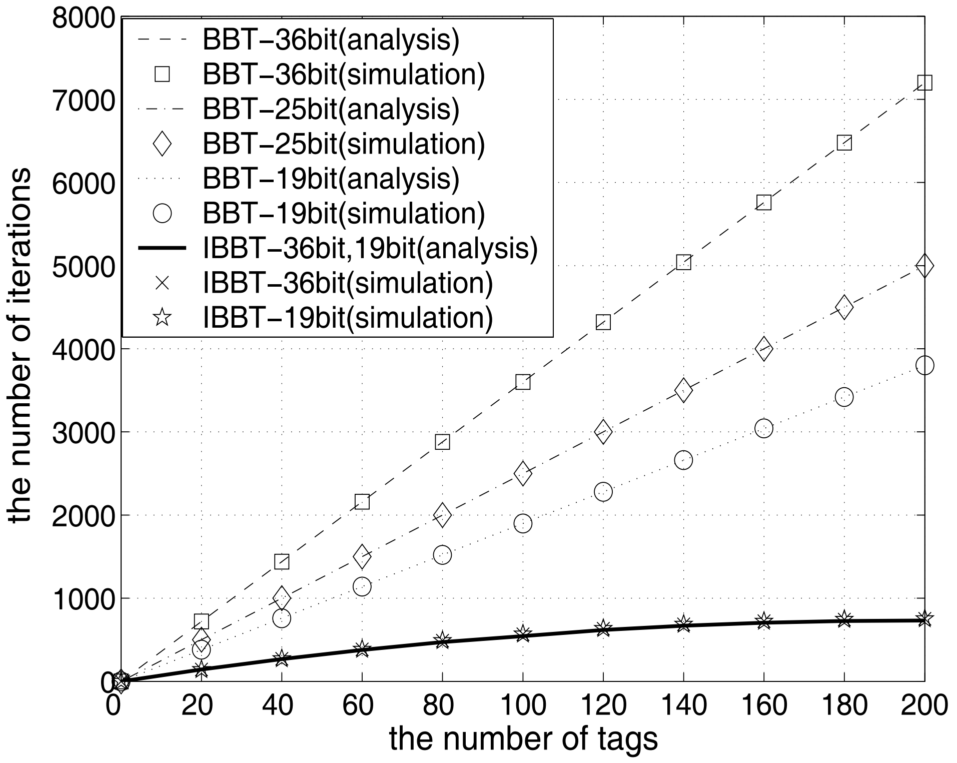

Fig. 4. The number of iterations vs. the number of tags

Fig.4 shows the number of iterations for the number of tags in BBT and IBBT according to the number of tag's ID bit[6]. In Fig.4, lines represent analytic results and symbols represent simulation results using OPNET. Analytic results are very closed to the simulation results. Comparing IBBT with BBT, the number of iterations in BBT increases as the number of tag's ID bit increases, while that of IBBT does not increase according to the number of tag's ID bit. That is why the IBBT shows much better performance for the larger tag's ID bit. We found that the performance of IBBT is 304% higher when the number of tags is 20 for the 36 bit tag's ID. Moreover, we found that the performance of IBBT is about 839% higher when the number of tags is 200. As the number of tags increases the performance of IBBT is much better than that of exiting algorithms.

6 Conclusion

We proposed and analyzed the fast wireless anti-collision algorithm(IBBT) in u-ID system. We mathematically compared the performance of the proposed algorithm with that of BS, SBT, and BBT. We also validated analytic results using simulation. IBBT shows better performance than existing algorithms as the number of tag's ID bit increases. The energy consumption is also lower than existing algorithms since the number of bits transferred from the tags is much smaller. Furthermore, the performance of IBBT is much better than exiting algorithms as the number of tags increases. On conclusion, if we apply the proposed algorithm to the u-ID system, it will contribute to improve the performance of the u-ID system because the reader can identify more tags with shorter time and less energy.

Acknowledgment

This research is partially supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Science and Technology(MOST) 21st Century Frontier R&D Program in Korea.

References

[1] H. Vogt, Efficient Object Identification with Passive RFID tags, In International Conference on Pervasive Computing, Zurich, (2002).

[2] K. Finkenzeller, RFID Handbook: *Radio-Frequency Identification Fundamentals and applications*. John Wiley and Sons Ltd, (1999).

[3] J. L. Massey, Collision resolution algorithms and random-access communications, Univ. California, Los Angeles, Tech. Rep. UCLAENG -8016, Apr. (1980).

[4] M. Jaoccurt, A. Ehrsam, U. Gehrig, Contactless Identification Device With Anticollision Algorithm, IEEE Computer Society CSCC99, Conference on Circuits, Systems, Computers and Communications, Jul. 4-8 Athens (1999).

[5] Auto-ID Center, *Draft Protocol Specification for a Class 0 Radio Frequency Identification tag.*, (2003).

[6] EPC Global, EPC^{TM} *Tag Data Standards Version 1.1 Rev.1.24*, Apr. (2004).