RFID 시스템에서의 고속 충돌 방지 알고리즘

이충희, 김재현
아주대학교 전자공학부 무선인터넷연구실
{hedreams, jkim}@ajou.ac.kr

A Fast Anti-collision Algorithm for RFID System

Choong-Hee Lee and Jae-Hyun Kim
Wireless Internet & Network Engineering Research Lab., Ajou University

요 약

본 논문에서는 기존에 나와있는 RFID 시스템의 태그 인식 과정과 충돌 방지 알고리즘을 분석하고, 이를 개선하기 위해 새로운 알고리즘을 제안한다. 또한, 성능 분석을 위해 기존의 알고리즘과 제안한 알고리즘을 수학적으로 분석하고 시뮬레이션을 통해 이를 검증한다. 결과에 따르면, 제안한 충돌 방지 알고리즘을 사용할 경우, 기존의 알고리즘을 사용할 때보다 같은 수의 태그를 인식하는데 걸리는 시간이 훨씬 적게 소요되는 것을 알 수 있다.

I. 서론

* † Radio frequency identification system(RFID) 시스템은 사물을 인식하는데 사용하는 센서 네트워크의 한 형태로써, 일반적으로 리더와 태그로 구성된다. 리더는 태그의 고유한 ID 또는 태그에 저장되어 있는 정보를 인식한다. 리더는 저장장치, 전력장치 등을 내장하고 있으며, 무선 통신을 통해 자신의 범위 내에 있는 태그들의 고유한 ID를 요청한다. 태그는 리더로부터 오는 요청 신호로부터 작동에 필요한 에너지를 얻어 연산을 하고 응답을 한다. RFID 리더는 명령을 브로드캐스트하며, 이를 받은 모든 태그들은 리더에게 응답 메시지를 보낸다. 이때, 하나의 태그만이 응답을 하면 리더는 이를 인식할 수 있지만, 다수의 태그가 동시에 응답을 할 경우 이들의 응답 메시지가 무선 통신 채널 상에서 충돌을 일으켜 리더는 응답을 받을 수 없게 된다. 이러한 문제를 “태그 충돌 문제”라 한다. 태그 충돌 문제를 해결하는 능력은 RFID 시스템의 성능을 결정하는데 결정적인 요소이다[1]-[3].

본 논문에서는 860MHz-930MHz의 주파수 범위에서 동작하는 EPC Class 1 시스템을 위한 충돌 방지 알고리즘을 제안한다. AutoID Center의 문서[4]상에는 일반적인 충돌 방지 알고리즘에 대한 자세한 언급이 없기 때문에 기존에 나와있는 시스템을 실험을 통하여 분석하였다. 실험에서 기존 시스템의 리더의 송신부 파형을 관찰하였고, 이를 리더가 브로드캐스팅하는 명령어의 시퀀스로 바꾸었으며, 이를 바탕으로 기존 시스템의 태그 인식 절차를 추정하고, 충돌 방지 알고리즘을 분석하였다. 마지막으로, 기존 알고리즘과 제안한 알고리즘을 수학적으로 분석하고, 이를 시뮬레이션을 통하여 검증하였다.

II. EPC Class 1 충돌 방지 알고리즘

EPC Class 1 system에서는 보통 다음과 같은 5 개의 명령어를 사용한다.[4]

Talk: [PTR]부터 [LEN] 길이만큼의 [VALUE] 값이 일치하는 태그는 리더의 명령에 응답할 수 있는 active 상태가 된다.
Quiet: [PTR]부터 [LEN] 길이만큼의 [VALUE] 값이 일치하는 태그는 quiet 상태가 되어 더 이상 리더의 명령에 응답을 하지 않는다.
ScrollAllID: 모든 태그가 8비트 프리엠블 뒤에 전체 identifier tag memory (ITM)을 전송한다.
PingID: [PTR]부터 [LEN] 길이만큼의 [VALUE] 값이 일치하는 태그는 [PTR] + [LEN]부터 8비트를 자신이 해당하는 빈 슬롯에 전송한다.
ScrollID: [PTR]부터 [LEN] 길이만큼의 [VALUE] 값이 일치하는 태그는 8비트 프리엠블 뒤에 전체 identifier tag memory (ITM)을 전송한다.

기존의 system으로 Alien Technology Corporation의 ALR-9780을 선정하였다. 기존의 시스템에서의 태그 인식 절차는 다음과 같다.

1. 리더는 주변의 태그들을 통신 가능한 상태로 만들기 위하여 ‘0’과 ‘1’로 시작하는 태그들에게 *Talk* 명령을 각각 3번씩 보낸다.
2. 다음으로 리더가 *ScrollAllID* 명령을 전송하면, 활성 상태에 있는 모든 태그들이 자신의 전체 ITM을 보내어 응답한다. 이 때 3가지 상황이 발생 할 수 있다.
   a. 만약, *ScrollAllID*에 대한 응답이 전혀 없다면, 리더는 주변에 태그가 없다고 판단하고, 주기적으로 *ScrollAllID* 명령을 반복하여 전송한다.
   b. 만약, 충돌 없이 하나의 응답만 들어온다면, 리더는 이를 인식하고, 주기적으로 *ScrollAllID* 명령을 반복하여 전송한다.
   c. 만약, 응답에 충돌이 있다면 리더는 주변에 다수의 태그가 존재한다고 판단하고 [PTR]=0, [LEN]=1, [VALUE]=0로 설정하여 MSB가 ‘0’인 태그들에게 *PingID* 명령을 보낸다. 만약, 빈 슬롯에 응답이 있으면, 리더는 응답이 있었던 빈 슬롯들을 bin 0에서 bin 7까지 순차적으로 처리하게 된다.
   d. 그림 1은 이러한 인식 절차의 예를 나타낸 것이다. 응답이 있는 빈 슬롯에는 [PTR]=0, [LEN]=[LEN]+3, [VALUE]=0+(3-비트 빈 슬롯 번호)로 설정하고 *ScrollID* 명령을 전송한다.
      i) 충돌 없이 ScrollID 응답을 전송받게 되면, 리더는 그 태그를 인식하고, Quiet 명령을 보내

* 이 논문은 2006 년도 2 단계 BK21 사업에 의하여 지원되었음
† 본 연구는 과학기술부 21 세기프론티어연구개발사업의 일환으로 추진되고 있는 유비쿼터스컴퓨팅 및네트워크원천기반기술개발사업의 지원에 의한 것임.

인식한 태그를 비활성화 시킨다. 마지막으로 태그의 비활성화 상태를 확인하기 위한 ScrollID를 보낸다.

ii) 만약, ScrollID 응답에 충돌이 있으면 해당 빈 슬롯에 여러 개의 태그가 존재한다고 판단하고, 태그를 인식하기 위한 다음 과정으로 PingID 명령을 보낸다. 이 때, [LEN]=[LEN]+3, [VALUE]는 이전 값의 뒤에 3비트 빈 슬롯 번호를 덧붙인 값이다.

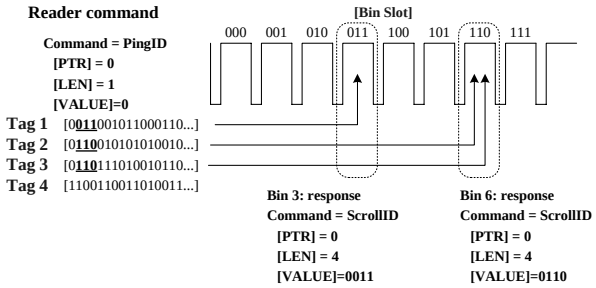

그림 1. EPC Class 1 알고리즘에서의 태그 인식 예제

EPC Class 1 시스템에서 태그 인식 절차는 트리구조를 따른다. [LEN]=1인 '0', '1' 두 노드에서 인식 과정이 시작되며, 각각의 노드마다 8개의 가지가 뻗어져 나오는 구조이다. 각각의 노드가 8개의 가지를 갖고 있는 이유는 *PingID* 응답이 8개의 빈 슬롯을 통해서 들어오기 때문이다. 마찬가지로 빈 슬롯 번호가 3비트이기 때문에 아래쪽 가지의 노드들의 [VALUE] 필드가 위쪽 가지의 노드의 필드보다 3비트 길다. 만약 응답이 있었던 마지막 빈 슬롯에 해당하는 태그들에 대한 인식이 끝날 경우, 리더는 응답이 있었던 빈 슬롯에 대해 각각 *ScrollID* 명령을 보내어 인식한 태그와 같은 [VALUE]를 가진 태그가 남아있는지 확인하고, 다시 트리 구조의 위쪽 가지로 옮겨간다. ITM의 MSB가 '0'인 태그들을 모두 인식한 후에는 '1'로 시작하는 태그들을 인식하기 위해 [PTR]=0, [LEN]=1, [VALUE]=1로 설정하여 *PingID*를 보낸다. 이후의 과정은 동일하며, 모든 태그에 대한 인식이 끝날 경우 다시 주기적으로 *ScollAllID* 명령을 전송하게 된다.

기존의 알고리즘에서 태그의 ITM의 분포가 랜덤한지 순차적인지는 태그 인식에 큰 영향을 미치지 못한다. 그 이유는 총 112비트 ITM의 앞부분이 96비트 태그 ID의 16비트 CRC로 구성되어 있기 때문이다.

III. 제안하는 충돌 방지 알고리즘

본 논문에서는 리더가 *PingID* 응답의 각각의 빈 슬롯에서 충돌 유/무 정보를 알 수 있다고 가정하고 이를 충돌 방지 알고리즘에 사용한다. 제안한 알고리즘에서는 다수의 태그를 고속으로 인식하는 상황을 일반적인 상황으로 고려했기 때문에, 기존 알고리즘에서의 *ScrollAllID*를 전송하는 절차를 생략하였다. 그림 2는 제안한 알고리즘의 태그 인식 절차 과정을 예로 나타낸 것이다. 리더는 응답이 있는 빈 슬롯들을 2가지 다른 방식으로 처리하게 된다. 만약 빈 슬롯에 들어온 8비트 응답 중에 빈 슬롯 번호를 제외한 5비트에 충돌이 없을 경우, 기존의 알고리즘과 마찬가지로 우선, *ScrollID*를 전송하게 된다. *ScrollID* 응답에서 충돌이 발생하지 않을 경우, 리더는 태그를 인식한다. 하지만, ScrollID 응답에서 충돌이 발생할 경우에는, 빈 슬롯에 들어온 8bit 응답까지 [VALUE]가 일치함을 알기 때문에 [LEN]=[LEN]+8로 설정하고 알고 있는 [VALUE]까지 설정하여 *PingID*를 전송하게 된다. 만약, 빈 슬롯에 들어온 응답의 뒤 5비트에서 충돌이 있을 경우, 리더가 빈 슬롯 번호까지 [VALUE]가 동일한 2개 이상의 태그가 존재함을 알기 때문에 바로 [LEN]=[LEN]+3으로 [VALUE]는 빈 슬롯 번호까지 설정하여 *PingID*를 전송하게 된다. 그리고, 제안한 알고리즘에서는 인식 이후의 확인을 위한 *ScrollID*와 *PingID*를 전송하지 않는다. 그 이유는 만약에 어떠한 이유에서 인식이 되지 않은 태그의 경우 반복되는 다음 인식 과정에서 바로 인식될 것이기 때문이다.

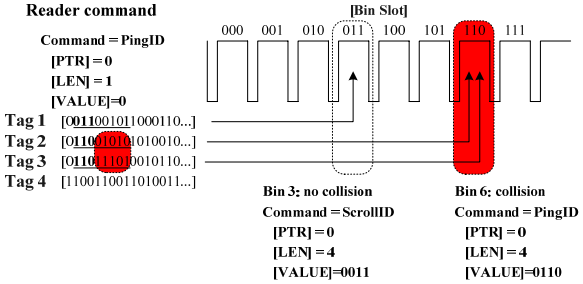

그림 2. 제안하는 알고리즘에서의 태그 인식 예제

IV. 성능분석

이 절에서는 기존의 알고리즘과 제안한 알고리즘의 성능을 분석한다. 이를 위해 전체 태그를 인식하는 동안 전송되는 각각의 명령어 전송 횟수를 계산하고, 실제 국내에서 사용되는 전송률과 EPC Class 1 표준에서 사용되는 파라메터들을 고려하여 태그 인식에 걸리는 시간을 계산한다. 먼저 두 알고리즘을 분석하는데 공통적으로 사용되는 전체 태그 인식 시간 $T_{identification}$은 다음과 같다.

$$\begin{aligned} T_{identification} &= CW \times n_{total} + \frac{b_{reader}}{DR_{reader}} + \frac{b_{tag}}{DR_{tag}} + T_{process} \\ &= CW \times n_{total} + \frac{(8.25+51)\times n_{total} + S_{LEN}}{DR_{reader}} \\ &\quad + \frac{n \times 128}{DR_{tag}} + T_{process}. \end{aligned} \tag{1}$$

위 식은 모든 태그를 인식하는 동안, 전송되는 비트수를 데이터 전송률로 나누어 시간을 계산한 것이다. 이 때, *CW*는 continuous wave를 전송하는데 걸리는 시간이며, DR_{reader}는 리더에서 태그로 전송할 때의 전송률, DR_{tag}는 태그에서 리더로의 전송률, b_{reader}와 b_{tag}는 각각 총 인식 과정 동안 리더와 태그에서 전송하는 비트 수이다. 8.25는 preamble을 비트수로 환산한 것이며, 51은 [VALUE] 필드를 제외한 명령어당 비트 수이다. n_{total}은 전체 명령어 전송 횟수이고, n은 *PingID*와 *ScrollID*, *ScrollAllID* 명령어 전송 횟수이다. 128은 태그의 응답을 비트 수로 나타낸 것이고, S_{LEN}은 [LEN] 필드의 값들의 합 즉, [VALUE] 필드의 비트 수의 총합이다. n_{total}, n, S_{LEN}을 구하기 위해 다음 두 절에서 각각의 [LEN]를 명령어들의 전송 반복 횟수를 구한다. 수학적인 방법은 [5]-[7]과 유사한 방식을 사용한다.

4.1 EPC Class 1 충돌 방지 알고리즘

[LEN]=1+3k인 *Scroll1ID*와 *PingID* 명령 전송 횟수 IS_k와 IP_k를 구하기 위해 각각의 명령어를 전송할 확률을 구한다. 리더는 *PingID* 응답에서 응답이 있는 빈 슬롯에 *Scroll1ID*를 전송한다. 빈 슬롯에 응답이 있을 확률 $P_{response}$는

$$P_{response} = 1 - \left(\frac{r-1}{r}\right)^n \tag{2}$$

이다. 이때 r은 빈 슬롯의 수로 EPC Class 1에서는 8이 되며, n은 전체 태그 수이다. 만약 빈 슬롯에 하나의 태그 만이 응답한다면, *Scroll1ID* 응답에서 충돌이 발생하지 않을 것이다. *Scroll1ID* 응답에서 충돌이 발생하지 않을 확률, P_{no_coll}은

$$P_{no\_coll} = n\left(\frac{r-1}{r}\right)^{n-1} \cdot \frac{1}{r} \tag{3}$$

이다. 따라서, n_{bin}을 [LEN]에 해당하는 전체 빈 슬롯의 수라 하고 m을 리더의 범위 내에 있는 전체 태그의 수라고 할 때, IS_k(k=1,2,3,…)는

$$IS_k = 2r^k \times \left[1 - \left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}} + \frac{m}{2r^{k-1}}\left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}-1} \cdot \frac{1}{r}\right] \tag{4}$$

하나의 빈 슬롯에 2개 이상의 태그가 응답을 한다면 *Scroll1ID* 응답에서 충돌이 발생할 것이다. 따라서 *Scroll1ID*에서 충돌이 발생할 확률, P_{coll}은

$$P_{coll} = P_{response} - P_{no\_coll} \tag{5}$$

이고, IP_k(k=1,2,3,…)는

$$IP_k = 2r^k \times \left[1 - \left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}} - \frac{m}{2r^{k-1}}\left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}-1} \cdot \frac{1}{r}\right] + 2r^{k-1} \times \left[\frac{m}{2r^{k-2}}\left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-2}}-1} \cdot \frac{1}{r}\right] \tag{6}$$

이다. 계산에서 빈 슬롯내의 태그의 수가 1보다 작을 때에는 해당 [LEN]에서 남은 태그가 모두 인식된다고 가정하였다.

4.2 제안하는 충돌 방지 알고리즘

제안한 충돌 방지 알고리즘에서 빈 슬롯내에 충돌이 없을 경우 리더는 *Scroll1ID*를 전송한다. 빈 슬롯에 충돌이 발생하지 않을 경우는 하나의 태그만이 응답했을 경우 또는 PingID 응답으로 보내게 될 8비트가 똑같은 2개 이상의 태그가 응답했을 경우이다. 빈 슬롯에 하나의 태그만이 응답할 확률, P_{tag_1}은

$$P_{tag\_1} = n\left(\frac{r-1}{r}\right)^{n-1} \cdot \frac{1}{r} \tag{7}$$

이고, 2개 이상의 태그가 응답할 확률, $P_{tag\geq 2}$는

$$P_{tag\geq 2} = 1 - \left(\frac{r-1}{r}\right)^n - n\left(\frac{r-1}{r}\right)^{n-1} \cdot \frac{1}{r} \tag{8}$$

이다. 2개 이상의 태그가 응답했지만 3비트 빈 슬롯 번호를 제외한 5비트가 같다면 빈 슬롯 충돌은 발생하지 않기 때문에 리더는 *Scroll1ID*를 전송하게 되며 이때, *Scroll1ID* 응답에서 충돌이 일어나 다시 *PingID*를 전송하게 된다. 이러할 확률 $P_{bin_no_coll}$은

$$P_{bin\_no\_coll} = \left(\frac{1}{2^5}\right)^n \quad , n \geq 2. \tag{9}$$

이다. 하지만, $P_{bin_no_coll}$은 매우 작은 값이기 때문에 계산에서는 0이라고 가정하였다. 따라서, IS_k는

$$IS_k = n_{bin} \times P_{tag\_1} = 2r^k \times \left[\frac{m}{2r^{k-1}}\left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}-1} \cdot \frac{1}{r}\right] \tag{10}$$

이 되고, IS_k는

$$IP_k = 2r^k \times \left[1 - \left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}} - \frac{m}{2r^{k-1}}\left(\frac{r-1}{r}\right)^{\frac{m}{2r^{k-1}}-1} \cdot \frac{1}{r}\right] \tag{11}$$

이 된다.

V. 분석 결과 및 고찰

이 절에서는 제안한 알고리즘의 수학적 분석 결과를 기존 알고리즘의 분석 결과와 비교하고, 이를 다시 시뮬레이션 결과와 비교하여 검증한다. 표 1은 수학적 분석과 시뮬레이션에 사용한 수치들이다[4],[8]. 태그의 수는 500개까지 고려하였고, 시뮬레이션은 50개부터 500개까지 50개 간격으로 실시하였다. 수학적 분석에서는 태그의 ID가 랜덤하게 분포한다고 가정하였지만, 시뮬레이션에서는 랜덤한 분포와 시퀀셜한 분포를 모두 가정하였다.

표 1. 시뮬레이션 파라메터

Parameter	Value
CW	0.064 msec
T_o(master clock interval)	0.025 msec
$DR_{reader}(1/T_o)$	40 kbps
$DR_{tag}(2/T_o)$	80 kbps
Transaction gap(1.25 T_o)	0.03125 msec
Tag setup period(8 T_o)	0.2 msec
Tag response period(64 T_o)	1.6 msec

그림 3과 그림 4는 수학적 분석과 시뮬레이션 결과를 나타낸다. 선으로 표시된 것은 수학적 분석 결과이고, 기호로 나타난 것은 시뮬레이션 결과이다. 수학적 분석 결과가 시뮬레이션 결과와 매우 유사한 것을 확인 할 수 있다. 그림 3은 전체 명령어 전송 횟수를 태그의 수에 따라 나타낸다. 전체 명령어 전송 횟수는 *Scroll1All1D*, *Scroll1ID*, *PingID*, *Quiet* 명령의 전송 횟수를 모두 합한 것이다. 명령어 전송 횟수가 태그 수에 따라 거의 선형적으로 증가하는 형태로 나타난다. 태그가 500개 일 때, 전체 명령어 전송 횟수는 기존 알고리즘에서는 약 2410번, 제안한 알고리즘에서는 약 1200번 이다. 각 태그를 인식한 후에 확인을 위한 *Scroll1ID*와 *PingID*가 줄어들었고, 빈 슬롯의 충돌 유무 정보를 이용함으로써 불필요하게 전송되던 *Scroll1ID* 전송도 줄어든 것이다. 이를 비율로 나타내면 제안한 알고리즘이 기존 알고리즘에 비해 명령어 전송 횟수 측면에서 약 50.21%의 성능 향상이 있음을 알 수 있다.

그림 4는 전체 태그를 인식하는데 걸리는 시간을 나타낸다. 태그 인식 시간 역시 태그 개수에 따라 거의 선형적으로 증가함을 알 수 있다. ID가 랜덤하게 분포된 500개의 태그의 경우, 모든 태그를 인식하는데 걸리는 시간은 그림 4에서 보이는 바와 같이 기존 알고리즘을 사용할 경우 약 8.9초, 제안하는 알고리즘을 사용할 경우 약 4.7초이다. 이를 계산해보면 기존 알고리즘의

경우 초당 약 56개, 제안한 알고리즘을 사용할 경우 초당 약 106개의 태그를 인식할 수 있음을 알 수 있다. 나타난다. 제안한 알고리즘의 태그 인식 시간이 기존 알고리즘의 태그 인식시간보다 약 52.81%로 줄어듦을 알 수 있다.

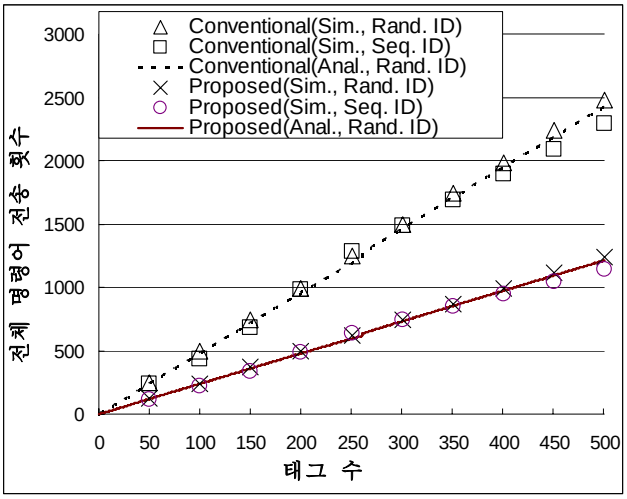

그림 3. 전체 명령어 전송 횟수

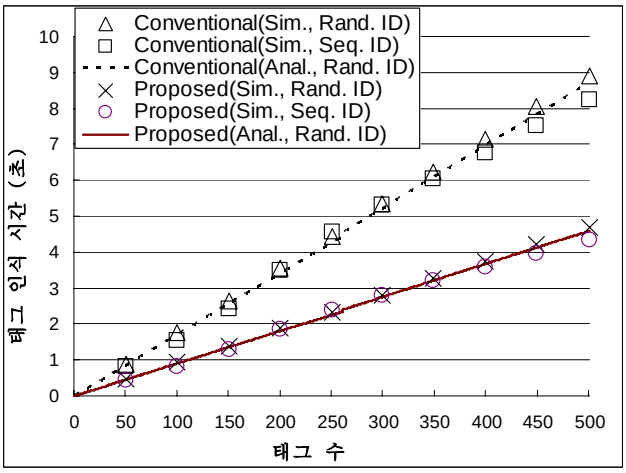

그림 4. 태그 인식 시간

VI. 결 론

본 논문에서는 성능 분석을 위해 기존에 있는 EPC Class 1 RFID 시스템의 태그 인식 절차와 충돌 방지 알고리즘을 분석하고, 이를 향상시키기 위한 새로운 알고리즘을 제안한다. 제안한 알고리즘에서는 *PingID* 응답에서 빈 슬롯 내의 충돌 정보를 이용하여 불필요한 *ScrollID* 전송 횟수를 줄였다. 또, 기존의 알고리즘에서 태그의 인식 이후에 중복적으로 전송하던 불필요한 확인 명령 전송도 줄였다. 마지막으로 제안한 알고리즘과 기존의 알고리즘의 성능을 수학적으로 분석하였으며, 이를 시뮬레이션을 통해 검증하였다. 결과에 따르면, 제안한 알고리즘에서 기존의 알고리즘에 비해 태그 인식속도가 약 52.81%로 줄어듦을 확인할 수 있었다. 제안한 알고리즘이 실제 시스템에 적용된다면, 많은 태그를 적은 시간에 인식 할 수 있을 것이다.

[참 고 문 헌]

[1] H. Vogt, "Efficient Object Identification with Passive RFID tags," *IEEE ICPC*, Zürich, 2002, pp.98-113.

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

[3] S. Sarma, D. Brock, and D. Engels, "Radio frequency identification and electronic product code," *IEEE MICRO*, 2001, pp. 50-54.

[4] Auto-ID Center, *Technical report 860MHz - 930MHz Class I Radio Frequency Identification Tag Radio Frequency & Logical Communication Interface Specification Candidate Recommendation, Version 1.0.1.*

[5] H. S. Choi, J. R. Cha and J. H. Kim, "Fast Wireless Anti-collision Algorithm in Ubiquitous ID System ," in *Proc. IEEE VTC 2004*, L.A., USA, Sep. 26-29, 2004.

[6] H. S. Choi, J. R. Cha and J. H. Kim, "Improved Bit-by-bit Binary Tree Algorithm in Ubiquitous ID System," in *Proc. IEEE PCM 2004*, Tokyo, Japan, Nov. 29-Dec. 3, 2004, pp. 696-703.

[7] H. S. Choi, and J. H. Kim, "A Novel Tag identification algorithm for RFID System using UHF," *Lecture Notes in Computer Science 3824: IEEE EUC*, Dec. 2005, pp. 629-638.

[8] EPCglobal, *EPC™ Tag Data Standards Version 1.1 Rev.1.24*, Apr. 20