RFID 시스템에서 리더기들간의 협업을 이용한 태그 인식 알고리즘

김홍화, 박숙영, 이상규
숙명여자대학교 컴퓨터과학과
e-mail:ruby-q, blue, sanglee@sookmyung.ac.kr

An cooperative tag recognition algorithm in RFID systems using multi-readers

Hong-Hua Jin, Sook-Young Park, Sang-Kyu Lee
Dept of Computer Science, Sookmyung Women's University

요 약

본 논문에서는 단일리더기가 아닌 멀티리더기를 이용하여 리더기들간의 상호 협동작업을 통하여 일정한 속도로 이동하는 태그를 최대한 많이 읽을 수 있는 멀티리더기들간의 협업을 이용한 향상된 태그인식 알고리즘을 제안하였다. 즉, 두 개의 리더기를 사용하여 많은 양의 태그가 서로 다른 이동 속도로 리더기를 지나 갈 때, 첫 번째 리더기가 저장해 놓은 태그정보를 두 번째 리더기가 이어받아 계속하여 나머지 태그를 인식하는 알고리즘을 개발한다. 시뮬레이션 결과 멀티리더기를 이용하면 단일리더기의 단점을 극복하여 보다 향상된 태그 인식 결과를 얻어내어 좀 더 안정적이고 효율적인 알고리즘을 제안하였다.

1. 서론

RFID(Radio Frequence Identification)기술은 최근에 무선 인식 기술의 중요성이 점차 커지면서 물류, 유통 분야 및 금융서비스 등의 분야에서 현재 사용 중인 바코드(Bar code)를 대체할 기술로 주목 받고 있다. 또한, 보안, 안전, 소방/방재, 환경관리, 생산자동화 등 다양한 분야에 적용될 수 있다[6]. 물체에 부착된 태그는 고유한 식별번호와 물체의 정보를 가지고 있으며 리더는 태그와의 무선 통신을 통하여 물체를 식별한다[1].

이러한 물체 식별을 위해 여러 가지 알고리즘이 사용되고 있지만 충돌 발생을 완전히 방지할 수 없어, 특정 태그가 오랫동안 리더에게 인식되지 못하는 태그 기아 문제(tag starvation problem)가 발생하게 된다. 태그 기아 문제는 RFID 시스템이 물체의 식별뿐만 아니라 이에 기반한 감시 및 추적 기능을 성공적으로 수행하지 못하도록 한다. 그러나 트리 생성을 기반으로 하는 결정론적 충돌 방지 방법은 트리를 만들 때 태그의 고유한 식별 ID를 사용하여 트리를 생성하기 때문에 모든 태그를 인식할 수 있고 과정을 예측할 수 있다는 장점이 있다. 태그가 부착된 물체는 이동성을 가질 수 있으며 태그 식별 속도가 물체의 이동 속도보다 느릴 경우 리더는 물체를 인식하지 못할 수 있다.

태그가 부착된 물체가 일정한 속도로 이동할 경우 단일 리더기로는 물체를 인식하지 못하는 문제가 발생하므로, 본 논문에서 제안한 방법에서는 멀티 리더기들 간의 상호 협업 작업을 통한 태그 식별을 이용하여 최대한 많은 태그를 식별하였다. 리더기 범위 내에 정지되어 있는 태그가 아닌 빠른 속도로 이동하는 태그들을 멀티 리더기들간의 협동 작업을 통해 질의 횟수를 줄이고 최대한 많은 양의 태그들을 효과적이고 안정적으로 읽을 수 있는 방법을 제시하였으며 시뮬레이션을 통하여 단일 리더기보다 멀티 리더기를 사용하였을 경우 보다 향상된 결과를 얻어 내었다.

※ 이 논문은 서울시 산학연 협력사업(10544)의 지원에 의하여 연구되었음

2. 관련 연구

리더가 태그를 식별할 때, 한 개의 태그만 리더에 응답을 하면 리더는 그 태그를 식별할 수 있지만 여러개의 태그가 동시에 응답하게 되면 충돌로 인하여 리더는 어떠한 태그가 응답했는지를 알 수 없게 된다. 이러한 충돌이 일어나는 것을 방지하기 위하여 많은 충돌방지 알고리즘(anti-collision algorithm)이 연구되어져왔다[4]. 태그 충돌 방지 알고리즘에는 공간 분할 다중 접속(SDMA), 주파수 분할 다중접속(FDMA), 시간 분할 다중 접속(TDMA), ALOHA, Slotted ALOHA, Dynamic Slotted ALOHA, 이진 트리 검색 알고리즘 등이 있다[1].

2.1 트리 기반 알고리즘(tree based algorithm)

트리 기반 알고리즘은 결정론적(deterministic) 알고리즘으로 이진 트리를 순회하며 모든 태그를 식별한다. 트리 워킹 알고리즘은 태그에 대한 질의 만으로 태그를 구분하는 비기억형 알고리즘으로 bit-by-bit 질의 방식으로 이진트리의 깊이 우선 탐색방법으로 태그를 식별한다[7]. 트리 워킹 알고리즘의 경우 비트단위로 태그를 검출하기 때문에 리더기 영역 내에 태그가 하나만 존재하더라도 마지막 비트까지 질의에 대한 응답을 해야 하고, 영역내의 많은 태그가 있을 경우에는 자주 충돌이 발생하기 때문에 시간이 오래 걸리는 등의 성능이 저하되는 문제점을 가지고 있다.

QT(Query Tree) 알고리즘은 결정론적 충돌 방지 방법 중 대표적인 기술로써 트리 워킹 보다 향상된 방법으로, 태그 응답에 따라 리더가 전송하는 질의가 결정되어

진다. 먼저 리더는 질의가 저장된 큐에서 d-비트 길이의 질의를 가져와 태그에게 브로드캐스트한다. 각각의 태그들은 리더로부터 전송받은 질의를 자신의 태그 ID와 비트 순서대로 비교하여, 자신의 태그 ID와 수신한 질의의 모든 비트가 일치하는 경우에만 자신의 태그 ID를 리더에게 전송하고, 일치하지 않으면 전송하지 않는다.

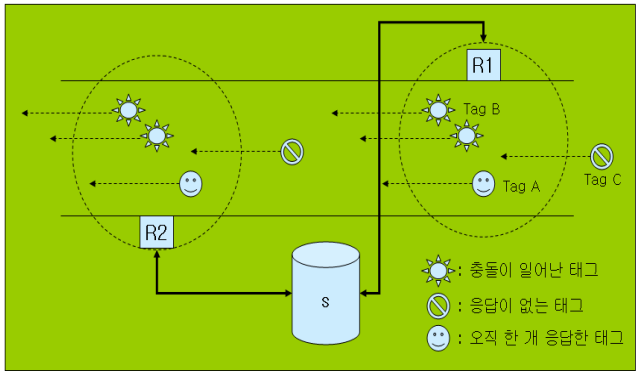
2.2 Adaptive Query Splitting 을 이용한 query 처리 방법

리더는 질의를 전송하고 태그는 자신의 ID로 응답한다. 질의는 하나의 비트 문자열을 포함한다. 리더는 두 개의 큐 Q와 CQ(Candidate Queue)를 갖는다. Q는 현재 식별 프로세스의 질의를 위한 비트 문자열을 저장하며 CQ는 다음 식별 프로세스의 질의를 위한 비트 문자열을 저장한다. 프로세스가 시작될 때, 리더는 CQ의 비트 문자열들로 Q를 초기화한다. 만약 CQ 가 어떠한 비트 문자열도 포함하고 있지 않다면 (예를 들어, 리더가 reset된 경우) Q는 QT 프로토콜에서처럼 두 개의 1비트 문자열인 0과 1로 초기화된다. 리더는 하나의 비트 문자열을 Q에서 꺼내어 질의를 전송한다.

3. 멀티리더기를 이용한 이동 태그 검출 방법

본 논문에서는 두 개의 리더기 R1, R2가 서로 협업하여 질의를 공유하면서 이동 태그를 최대한 많이 검출하는 방법을 제안한다. 즉, 두 개의 리더기를 사용하여 많은 양의 태그가 서로 다른 이동 속도로 리더기를 지나 갈 때, R1이 저장해 놓은 태그정보를 이용하여 R2가 그 정보를 이어받아 계속하여 나머지 태그를 인식하는 알고리즘을 제안한다.

또한, 모든 태그가 각자의 random한 속도를 가지고 R1에서 R2방향으로 [그림 1]에서와 같이 이동한다고(오른쪽 방향에서 => 왼쪽 방향으로) 가정하였다. 즉, 태그는 R1 범위 밖에 있다가 R1 수신 범위 내에 들어오게 되고 일정 시간 이동속도로 R1 수신 범위를 빠져 나가게 되어, R2 수신 범위 에 들어갔다가 빠져나오게 되는 단계를 거치게 될 것이다. 태그의 종류는 충돌이 일어나는 태그, 수신 범위 밖에 위치한 태그, 그리고 하나만 응답한 태그로 구분하여 처리 하였다.



[그림 1] RFID 시스템 모델

검출 방법은 간단하게 다음과 같은 세 단계로 나눌 수 있다.

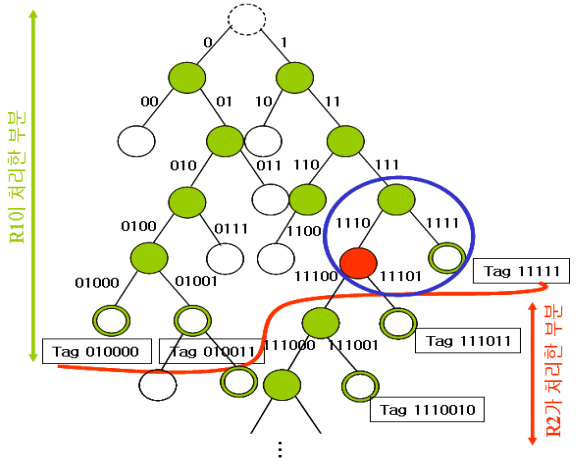
첫 번째 단계에서는 시뮬레이션에 사용할 태그를 생성하고, 두 번째 단계에서는 Reader-1(태그들이 먼저 지나가는 리더기)의 수신 범위와 Reader-2(태그들이 나중에 지나가는 리더기)의 수신 범위를 정하고, 각 질의를 해당 Queue에 저장하여 부분 트리를 완성한다. 마지막 단계에서는 Reader-1에서 저장해둔 Queue를 이용하여 Reader-2에서 나머지 트리를 완성한다.

3.1 리더기들간의 tree 구성 방법

리더의 질의에 수신 범위 내에 있는 모든 태그는 리더에 응답한다고 가정하고 Reader-1과 Reader-2를 R1, R2로 표시한다. 또한, R1이 R2에게 넘겨줄 정보를 저장하는 큐를 Q, MCQ(Max Candidate Queue), COLQ (COLlision Queue)로 정하였고 각각 다음과 같은 일을 한다.

- Q는 현재 단계에서 충돌이 일어난 질의에 대하여 해당 prefix에 '0'과 '1'을 붙여 쌍으로 저장해둔다. Queue에 질의를 저장하였다가 꺼낼 때도 항상 쌍으로 꺼내어 작업을 한다. 만약, Q가 어떠한 비트 문자열도 포함하고 있지 않다면(예를 들어, 리더가 reset된 경우) Q는 두 개의 1비트 문자열인 0과 1로 초기화된다. 리더는 하나의 비트 문자열을 Q에서 꺼내어 질의를 전송한다.
- MCQ는 첫 번째 리더가 어떠한 질의를 전송하였을 때, 오직 한 개의 태그 ID만 응답하였을 경우와 아무런 응답을 하지 않은 prefix를 확인하면서 태그 이동에 따라 충돌이 일어날 수도 있는 prefix에 대한 처리를 한다.
- COLQ는 첫 번째 리더기에서 질의 전송을 통하여 충돌이 일어난 것까지 확인하였지만 태그의 이동으로 인하여 미처 처리 하지 못한 질의가 있을 경우, 해당 질의의 비트 문자열을 저장해 두었다가 그 다음 리더기에 전달한다.

위에서 제시한 상황에 따라 첫 번째 리더기는 조건에 맞는 정보를 각각의 해당하는 Q, MCQ, COLQ에 저장해 두었다가 그 다음 리더기에 전달하게 된다. 자세한 내용은 3.2절의 리더기들간의 데이터 공유방법을 참고하기 바란다.



[그림 2] R1과 R2의 tag-list 구성하는 과정 예

[그림 2]에서는 두 개의 리더가 전체 트리를 구성하면서 tag-list를 구성하는 전반적인 과정을 보여준다. R1 리더 범위내에 각각 010000, 010011, 11111의 3개 태그 ID가 있고, 태그가 이동하면서 R2 리더 범위내에서 111011, 1110010의 2개 태그ID가 발견되었다고 하자. 가운데 가로 곡선을 기준으로 윗부분은 R1이 찾아낸 3개의 태그 ID이고 아랫부분은 R1이 넘겨준 정보를 이용하여 R2가 찾아낸 태그 ID라고 가정하자. 그림에서 동그라미로 표시되어 있는 부분은 [표 1]에서 두 번째 단계에 해당하는 처리 부분이다. 충돌이 일어난 태그가 그 하위 가지로 나누어지면서 이러한 트리 형태를 이루게 된다. 태그는 각자의 이동범위를 갖고 움직이기 때문에 단일리더기 범위 내에서 처리할 수 있는 태그의 개수가 한정되

어 있으므로, 멀티리더기를 이용한 방법을 제안하였다.

3.2 리더기들간의 데이터 공유 방법

태그의 위치에 따른 R1과 R2의 처리방법을 크게 다음과 같은 두 가지 경우로 분류해 볼 수 있다.

해당 질의를 기준으로 하여 태그 충돌이 일어나서 해당 질의에 '0'을 추가한 LQ(Left edge Query), 태그 충돌이 일어나서 해당 질의에 '1'을 추가한 RQ(Right edge Query)이다. 또한, LQ에서 충돌이 일어난 개수를 세는 변수를 LQN(Left edge Query Number)라 하고 RQ에서 충돌이 일어난 변수를 RQN(Right edge Query Number)라고 한다.

Queue / Response edge		Q	MCQ	COLQ
Response Left edge	Response Right edge	Q에 저장하는 query	MCQ에 저장하는 query	COLQ에 저장하는 query
①LQN = 0	RQN = 0			LQ -> COLQ RQ -> COLQ
②LQN = 0	RQN = 1			LQ -> COLQ RQ->COLQ->tag id저장
③LQN = 1	RQN = 0			LQ->COLQ->tag id저장 RQ -> COLQ
④LQN = 1	RQN = 1		LQ->MCQ->tag id저장 RQ->MCQ->tag id저장	
⑤LQN = 0	RQN > 1	RQ0 -> Q RQ1 -> Q	LQ -> MCQ	
⑥LQN > 1	RQN = 0	LQ0 -> Q LQ1 -> Q	RQ -> MCQ	
⑦LQN = 1	RQN > 1	RQ0 -> Q RQ1 -> Q	LQ->MCQ->tag id저장	
⑧LQN > 1	RQN = 1	LQ0 -> Q LQ1 -> Q	RQ->MCQ->tag id저장	
⑨LQN > 1	RQN > 1	LQ0 -> Q LQ1 -> Q RQ0 -> Q RQ1 -> Q		

• LQ: left edge query
• RQ: right edge query
• LQN: 왼쪽 태그 개수 [=0 -> no response, =1 -> 하나 발견, >1 -> 충돌발견]
• RQN: 오른쪽 태그 개수

[표 1] 질의응답에 따른 처리방법

우선, R1의 처리방법은 다음과 같다.

1. 태그가 R1의 수신 범위 밖에 있으면 Idle query상태이며, 어떠한 태그도 응답하지 않는다.

2. 태그가 R1의 수신 범위에 있으면 다음과 같은 두 가지 상태로 나눌 수 있다.

- Readable query: 오직 하나의 태그만 응답하여 리더에게 성공적으로 인식된다. 해당 질의를 MCQ에 저장하고 태그 ID도 함께 저장한다.
- Collision query: 다수의 태그가 응답하여 태그 충돌이 발생하는 경우 해당 질의에 '0'과 '1'을 붙여서 큐에 넣는다.

질의응답에 따른 처리방법을 [표 1]에서 보여주고 있다. R1은 다음 [표 1]에서와 같이 9가지 경우에 따른 여러 정보를 저장하고 있다가 R2에 넘겨주게 된다.

다음, R2의 처리방법은 다음과 같다.

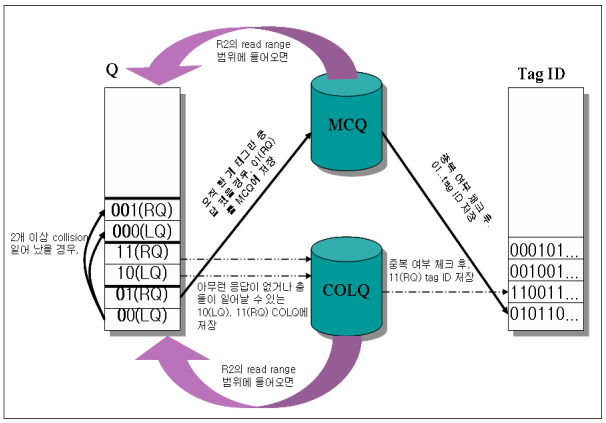
R2는 R1에서 저장한 MCQ와 COLQ를 갖고 다시 질의를 전송한다. 이는 오직 하나의 태그만 있다고 판단되어 저장해 놓은 태그가 이동할 때, 충돌을 일으키는 태그가 나타날 수 있는 경우를 대비하여 한번 더 검증해보기 위함이다. 아래 [그림 3]은 Q, MCQ, COLQ들 간의 관계도이다. 그림에서와 같이 Q에는 항상 prefix 뒤에 '0'과 '1'이 붙은 한 쌍의 질의가 들어있다. 오직 한개의 태그만 응답 했을 경우와 아무런 응답이 없을 경우에는 MCQ에 들어가게 되고 태그 ID를 저장해야 한다. 또한, 충돌이 일어날 가능성이 있는 경우에는 COLQ에 넣는다.

충돌이 일어날 가능성이 있는 경우는 다음과 같은 세 가지 경우로 분류하여 볼 수 있다.

1) LQN = 1 & RQN = 0, R1에서 LQ로 시작하는 태그ID가 한 개 있다고 판단하였는데 그다음에는 충돌이 일어날 가능성이 있고, RQ로 시작하는 태그 ID는 충돌이 일어났다고 판단하였는데 그 다음 순간에는 아무런 응답이 없을 경우, LQ와 RQ모두 COLQ에 넣은 후 LQ의 태그 ID를 저장한다.

2) LQN = 0 & RQN = 1, 위의 경우와 반대일 경우, LQ와 RQ모두 COLQ에 넣은 후 RQ의 태그 ID를 저장한다.

3) LQN = 0 & RQN = 0, 모두 응답이 없을 경우, LQ와 RQ모두 COLQ에 넣는다.



[그림 3] 여러 큐들 사이의 관계도

3.3 알고리즘

Algorithm 1. Reader-1 Operation

```
1.    while Q != NULL do
2.        LQ = pop (Q)
3.        Transmit a query including LQ Receive tag responses and detect collision
4.        if collision occurs then
5.            LQN > 1   //left query number
6.        else if receive only a response then
7.            LQN = 1
8.        else  /* receive no response */
9.            LQN = 0
10.       end if
11.       RQ = pop (Q)
12.       Transmit a query including RQ Receive tag responses and detect collision
13.       if collision occurs then  //RQN > 1
14.           if (LQN > 1) then
15.               LQ0, LQ1, RQ0, RQ1, push in Q
16.           else if (LQN == 1) then
17.               LQ, RQ push in MCQ  -> store the LQ tag ID
18.           else (LQN == 0)
19.               push (MCQ,LQ)  & RQ0, RQ1 push in Q
20.           end if
21.       else if receive only a response then      //RQN = 1
22.           if  (LQN > 1) then
23.               LQ0, LQ1 push in Q & push (MCQ, RQ)  -> store the RQ tag ID
24.           else if (LQN == 1) then
25                LQ, RQ push in MCQ ->  store the LQ, RQ tag ID
26.           else (LQN == 0)
27.               LQ, RQ push in COLQ -> store the RQ tag ID
28.           end if
29.       else /* no response ,RQN = 0 */
30.           if  (LQN > 1) then
31.               LQ0, LQ1 push in Q &  push (MCQ,RQ)
32.           else if (LQN == 1) then
33.               LQ, RQ push in COLQ -> store the LQ tag ID
34.           else (LQN == 0)
35.               LQ, RQ push in COLQ
36.           end if
37.       end if
38.   end while
```

[알고리즘 1] Reader-1 Operation

본 논문에서 제시한 알고리즘은 초기 생성한 태그들의 위치 값이 제일 작은 태그가 R1 수신 범위를 벗어 날 때까지 반복을 하게 된다. [알고리즘 1] 첫째 줄부터 셋째 줄까지 큐가 Null 이 아닌 동안, LQ를 Q로부터 꺼낸다. LQ로 충돌이 일어나는 태그에 대하여 세 가지로 구분하는 것은 넷째 줄부터 열 번째 줄까지 나타내었다. 열세

번째 줄에서 스무 번째 줄까지는 RQ를 Q에서 꺼내서 RQN이 한 개 이상 충돌이 일어나는 경우와 LQN이 한 개 이상충돌이 일어난 경우, 오직 한 개 응답했을 경우, 응답이 없을 경우 각각 처리 방법에 대하여 나타내었다. RQN이 오직 한 개 충돌이 일어났을 경우와 LQN 의 세 가지 경우에 대하여 라인 스물한 번째 줄에서 스물여덟 번째 줄까지 나타내었다. 스물아홉 번째 줄에서 서른일곱 번째 줄까지는 RQN이 응답이 없을 경우와 LQN 의 세 가지 경우에 대하여 R1에서의 처리 방법을 나타내었다.

이와 같은 처리 방법을 통하여 R1은 MCQ와 COLQ에 각각 해당 정보를 저장해두게 된다. R1에서 완성해놓은 이러한 정보를 R2에서 사용하게 됨으로써 태그 식별 속도를 많이 줄일 수 있게 된다. R2의 알고리즘은 R1알고리즘과 비슷한데 단지 MCQ와 COLQ의 내용을 R1의 저장값으로 초기화하여 질의를 보내는 부분만 추가하면 된다.

4. 실험 및 결과

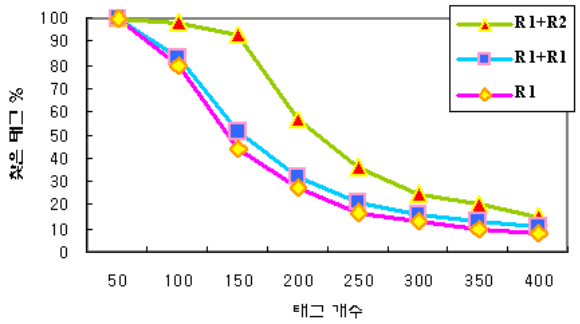
본 시뮬레이션에서는 ISO 15693-3에 의하여 유일한 태그ID를 식별하는 UID(Unique Identifier)를 제외한 그 외의 데이터를 저장하기 위한 128비트의 태그 ID를 생성하여 사용하였다. 응용프로그램 마다 사용하는 비트의 길이가 달라질 수 있다. 태그가 R1의 범위 밖에서부터 R1의 범위를 통과하고, 계속하여 R2의 범위 밖에서부터, R2의 범위를 통과하는 전체과정을 확인하기 위하여 각각 R1 범위의 최대값과 최소값, R2범위의 최대값과 최소값을 정하였다. 즉, 임의로 R1의 최소값을 5, 최대값을 180을 주었고 R2의 최소값을 200, 최대값을 375를 주었으며 R1의 범위와 R2의 범위가 서로 겹치는 부분이 없게 하기 위하여 간격을 20으로 설정하였다. 이동하는 태그를 표현하기 위하여 x축 기준으로 정수의 난수값을 생성하였으며 이 값은 비트별 검사가 한번 끝나는 시점에서 한번 씩 바뀌게 된다.

테스트에 사용한 태그의 개수는 다음과 같다. 단일리더기가 처리할 수 없는 많은 양의 태그를 멀티리더기가 협업작업을 통하여 처리하는 과정을 보여주기 위하여 100개 이상의 태그개수를 정하여 테스트하였다. 100, 150, 200, 250, 300, 350, 400 개의 Tag ID를 생성한 후, 각 태그의 초기 값을 작은 범위내의 정수 난수값을 생성하고 생성한 정수 난수값 중 제일 작은 값이 R1의 수신 범위를 벗어 날 때까지 값을 증가시키면서 테스트 하였다.

정수 난수값은 10에서 200까지의 10 단위로 증가하는 값을 사용하였다. 즉, 각각의 태그 개수를 정수 난수값으로 테스트해서 나온 결과의 평균값을 얻어내었다. [그림 7]은 R1, R1+R1, R1+R2의 테스트 결과화면이다.

R1은 단일리더기로 얻어낸 결과이고 R1+R1은 독립적으로 동작하는 단일리더기 두 개로 테스트해본 결과이고 R1+R2는 상호 협업작업을 하는 두 개의 리더기로 얻어낸 결과이다. 그림에서와 같이 태그개수가 적을 때는 세 가지 경우 모두 모든 태그를 찾아내어 100%의 결과를 보이지만 태그개수가 점점 많아짐에 따라 찾아내는 태그 개수가 점점 줄어들게 되고 결과적으로 그다지 큰 차이가 없음을 확인할 수 있다.

[그림 7]에서처럼 R1이 찾은 태그 개수에 비해 R1+R2는 훨씬 더 많은 태그 개수를 찾아내었고 태그개수가 점점 많아짐에 따라 R1+R2는 적게는 7%, 많게는 49%의 높은 태그인식률을 보였다. 또한, R1+R1가 찾은 태그 개수에 비해 R1+R2가 훨씬 더 많은 태그 개수를 찾아 낸 것을 확인 할 수 있다. 즉, 적게는 4%, 많게는 42%의 높은 태그인식률을 보였다. 결과적으로 리더기들간의 협업작업을 통하여 얻어낸 결과는 R1과 R1+R1보다 평균 적게는 4%, 많게는 49%의 향상된 결과를 얻을 수 있다.



본 논문에서는 단일리더기가 아닌 멀티리더기들간의 협업 작업을 거쳐 이동하는 태그를 더 안정적이고 더 효율적으로 읽어내는 방법을 제안하였다. 결과적으로 제안한 알고리즘에 따라 태그인식을 하면 maximum common prefix 길이만큼 트리를 추적한 후, 해당 태그를 인식하게 된다.

5. 결론

태그가 부착된 물체가 빠른 속도로 이동할 경우, 단일리더기로는 물체를 인식하지 못하는 경우가 많기 때문에 멀티리더기들간의 상호 협업 작업을 통한 태그 식별을 이용하여 최대한 많은 태그를 식별하고자 한다.

본 논문에서 제안한 방법에서는 리더기 범위 내에 빠른 속도로 이동하는 태그들을 멀티리더기들간의 협동 작업을 거쳐 질의 횟수를 줄이고 최대한 많은 양의 태그들을 효과적이고 안정적으로 읽을 수 있는 방법을 구현하였으며 시뮬레이션을 통하여 단일리더기를 사용하였을 때보다 본 논문에서 제안한 방법을 이용하여 4% ~ 49% 더 향상된 결과를 얻어 내었다.

참고문헌

[1] Jihoon Myung & Wonjun Lee. Adaptive Splitting Protocols for RFID Tag Collision Arbitration. MobiHoc'06. May 22-25, 2006, Florence, Italy.

[2] Klaus Finkenzeller. RFID HANDBOOK: Fundamentals and Applications in Contactless Smart Cards and Identification 2nd Edition.

[3] Feng Zhou, Chunhong Chen, Dawei Jin, Chenling Huang, Hao Min. Evaluating and Optimizing Power Consumption of AntiCollision Protocols for Applications in RFID Systems.

[4] Sok-Won Lee, School of Electrical and Electronic Engineering College of Engineering, A Multiple Access Algorithm for Passive RFID tags, June 3 2005

[5] Part1: Active and Passive RFID: Two Distinct, But Complementary, Technologies for Real-Time Supply Chain Visibility

[6] Feng Zhou, Dawei jin, Chenling Huang and Hao Min, "White Paper: optimize the power Consumption of Passive Electronic Tags for Anti-collision Schemes," Auto-ID center Fudan Univ., October, 2003

[7] Ari Juels, Ronald L. ivest and Michael Szydlo, "The Block Tag: Selective blocking of Tags for Consumer Privacy," Proceedings of the 10th ACM conference on Computer and communication security, ISBN: 1-58113-738-9, pp. 103-111,2003