개선된 타임 슬롯 방법을 이용한 효과적인 태그 인식 알고리즘†

(An Efficient Tag Identification Algorithm Using Improved Time Slot Method)

김 태 희*, 김 선 경**
(Tae Hee Kim and Sun Kyung Kim)

요 약 최근 유비쿼터스 환경 구축의 가장 핵심은 센서 네트워크와 RFID 시스템이다. 이 중 RFID 시스템은 태그의 전자정보를 RF 신호를 이용하여 리더에게 전송한다. RFID 시스템은 다중 태그의 존재로 인해 충돌이 발생하고 태그 인식 성능이 저하된다. 그래서 태그 충돌을 중재할 수 있는 방법이 필요하다. 본 논문은 태그 간 충돌을 줄이며 좀 더 빠른 태그 인식이 가능한 하이브리드 방법을 제안한다. 본 논문에서 제안하는 방법은 트리기반 알고리즘의 장점인 확실성을 기반으로 동작하며 충돌을 줄이기 위해 태그 아이디를 이용하여 전송 타임 슬롯을 결정한다. 시뮬레이션을 통한 성능평가에서 다른 트리기반의 알고리즘과 다른 하이브리드 알고리즘에 비하여 충돌 횟수와 쿼리 수에서 높은 성능을 가진다는 것을 보여준다.

핵심주제어 : RFID 시스템, 충돌 방지 알고리즘, 태그 인식, ETIA

Abstract In recent year, the cores of ubiquitous environment are sensor networks and RFID systems. RFID system transmits the electronic information of the tag to the reader by using RF signal. Collision happens in RFID system when there are many matched tags, and it degrades the tag identification performance. Such a system needs algorithm which is able to arbitrate tag collision. This paper suggests a hybrid method which reduces collision between the tags, and can quickly identify the tag. The proposed method operates based on certainty, which takes an advantage of tree based algorithm, and to reduce collision it selects transmission time slot by using tag ID. The simulation results show the suggested method has higher performance in the number of queries and collision compared to other tree based and hybrid algorithms.

Key Words : RFID system, Anti-collision algorithm, Tag Identification, ETIA

1. 서 론

최근 유비쿼터스 환경의 구축으로 RFID (Radio Frequency Identification) 시스템이 널리 이용되고 있다. RFID 시스템은 태그에 부착된 전자 정보를 RF(Radio Frequency) 신호를 이용하여 리더와 송수신한다.

RFID 시스템은 하나의 리더가 자신이 송수신 가능한 리더 범위 내의 Tag(태그)에게 모든 정보를 송신

† 이 논문은 대구대학교 학술연구비 지원에 의한 논문임
* 대구대학교 컴퓨터·IT공학부
** 대구대학교 컴퓨터·IT공학부, 교신저자

하라고 신호를 보내게 된다. 이때 신호를 수신 받은 태그는 자신의 ID(아이디)를 RF 신호를 이용해 리더에게 보내게 된다. 그러나 리더는 다수의 태그가 보낸 신호로 인해 충돌이 발생하고 태그의 인식이 불가능해진다. 이것이 태그 간의 충돌인 것이다. 이와는 반대로 여러 개의 리더가 하나의 태그에게 신호를 송신해서 생기는 리더 간의 충돌도 존재한다. 본 논문에서는 다수의 태그에서 생기는 태그 간 충돌 문제를 효과적으로 해결하는 충돌 중재 방법을 제안한다.

앞선 연구들에서 태그의 충돌을 중재하는 알고리즘은 크게 두 가지로 분류된다. 트리 기반의 방식과 확률적 방식이 있다. 확률적 기반의 방식에는 알로하라고 하는 대표적인 알고리즘이 있다. 이 방법은 리더가 보낸 명령어 내의 파라메타와, 태그가 자신이 가지고 있는 랜덤 제너레이터를 이용하여 특정한 시간을 선택하여 리더에게 아이디를 전송하는 방식이다. 만약 여러 개의 태그가 같은 시간을 선택하여 아이디를 전송하였다면 충돌이 발생하게 되는 것이다. 그리고 이 충돌된 아이디는 아주 긴 시간동안 인식되지 못하는 태그 기아상태에 빠질 수 있는 단점을 가지고 있다. 대표적인 알고리즘은 알로하, 슬롯 알로하, 그리고 프레임 슬롯 알로하가 있다[1,2,3].

트리기반의 알고리즘은 태그의 충돌이 발생하면 태그의 아이디를 트리 형태로 확장시켜 인식하는 방법이다. 이 방법은 다수의 태그가 존재할 경우 많은 충돌로 인해 태그의 재전송이 빈번하게 요구되고 이로 인해 전송지연이 야기된다. 대표적인 트리 기반 알고리즘에는 이진 트리 알고리즘과 쿼리 트리 알고리즘 등이 있다[4,5].

본 논문에서 제안하는 충돌 중재 방법에서 태그 아이디 확장은 쿼리 트리 알고리즘 방식을 태그의 응답 과정은 슬롯 알로하 방식을 변형하여 이용한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구로 확률적 방식과 트리 방식의 충돌 방지 알고리즘을 설명한다. 3장에서는 개선된 타임 슬롯을 이용한 효과적인 태그 인식 알고리즘을 제안하고 동작 방식을 설명한다. 4장에서는 제안 방법의 성능 평가를 하고 마지막으로 5장에서 결론을 기술한다.

2. 관련 연구

2.1 쿼리 트리 알고리즘

Query Tree [5](QT, 쿼리 트리) 알고리즘은 리더의 질의와 태그의 응답을 반복하면서 태그를 식별한다. QT 알고리즘을 이용한 태그의 식별 과정은 다음과 같다.

<표 1> 쿼리 트리 알고리즘의 동작 과정

스텝	1	2	3	4	5	6	7	8	9
쿼리	ε	0	1	00	01	10	11	110	111
태그 응답	X	X	X	001	10	idle	X	110	111
태그1 (001)	001	001		001					
태그2 (010)	010	010			010				
태그3 (110)	101		101				110	110	
태그4 (111)	111		111				111		111
큐	0 1	1 00 01	00 01 10 11	01 10 11	10 11	11	110 111		

리더가 태그에게 k 비트의 Prefix(프리픽스)를 모든 태그에게 전파하면, 태그는 전달된 프리픽스와 자신이 소유한 아이디가 일치하는지 검사한다. 만일 일치한다면 태그는 k+1번째부터 나머지 아이디를 리더에게 응답한다. 다수의 태그가 응답함으로 인해 충돌 사이클이 발생했다면 같은 프리픽스를 가진 태그가 두 개 이상 존재한다는 것을 의미하므로 리더는 큐에 현재 프리픽스에 '0'과 '1'을 접목한 새로운 두 개의 프리픽스를 Queue(큐)에 입력한다. 표 1은 QT 알고리즘의 동작 과정을 나타내고 있다.

QT에서 태그들을 인식하기 위한 리더와 태그간의 질의 응답 과정을 '라운드'라고 할 때, 표 1의 과정에서는 총 9회의 라운드가 필요하다. 스텝 1, 2, 3에서는 여러 개의 태그 응답으로 충돌이 발생하여 전송 받은 프리픽스 뒤에 0과 1을 붙여 새로운 쿼리를 생성하였다. 스텝 4, 5, 8, 9에서는 하나의 태그가 응답하여 태그를 식별할 수 있었다. 그러나 6 라운드에서는 '10'의 프리픽스를 갖는 태그의 부재로 인해 태그 Idle cycle (무응답 상태)인 유휴 사이클이 나타났다.

QT 알고리즘은 구현이 쉽고, 적은 비용으로 생산이 가능하다는 장점을 갖는다. 그러나 태그 아이디의 분포 상황을 고려하지 않은 상태에서 태그 아이디를 확

장하는 것은 유휴 사이클의 원인이 된다. 또한 이러한 환경으로 인해 태그 인식에 많은 질의와 전송 비트 수를 필요로 한다.

2.2 슬롯 알로하 알고리즘

알로하 기반의 알고리즘은 보통 Slotted Aloha[2](슬롯 알로하) 알고리즘을 말하며 이는 태그의 응답 시간을 고정된 몇 개의 슬롯(Slot)으로 나누고 각 태그들은 자신이 사용할 슬롯을 선택하여 자신의 태그 아이디를 전송하는 방식이다. 여기서 슬롯이란 시간 간격을 의미한다. 이 방식에서 리더는 나누어진 각 타임 슬롯에 태그 아이디가 두 개 이상 응답하면 충돌 발생으로 인식한다.

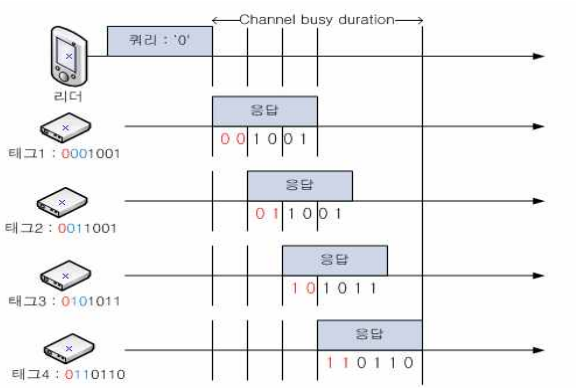
그림 1은 슬롯 알로하 알고리즘을 이용하여 리더가 4개의 태그 아이디를 식별하는 과정이다. 먼저 리더는 태그들에게 요청 메시지를 전파하면서, 각 태그들에게 슬롯 선택에 대한 정보를 전송한다. 이 때 각 태그들은 전송된 정보를 이용하여 자신의 슬롯을 결정한다[6]. 태그2와 태그3은 슬롯1, 태그1은 슬롯3, 태그4는 슬롯2에서 응답하였다. 슬롯2와 슬롯3은 하나의 태그가 응답하여 태그를 인식하였고, 슬롯1은 두 개의 태그가 응답하여 충돌이 발생하였다. 충돌이 발생한 태그들은 이러한 과정을 반복하여 모든 태그가 식별될 때까지 반복한다.

Down-Link (리더->태그)	응답요청	슬롯1	슬롯2	슬롯3	응답요청	슬롯1	슬롯2
UP->Link (태그->리더)		충돌	1111	0010		0110	1110
태그1				0010			
태그2		0110				0110	1110
태그3		1110					
태그4			1111				

(그림 1) 슬롯 알로하 알고리즘의 동작 과정

2.3 하이브리드 쿼리 트리 알고리즘

Hybrid Query Tree algorithm[7](HQT, 하이브리드 쿼리 트리 알고리즘)은 4-ary 검색 트리 기법을 적용하여 프리픽스가 1비트에서 2비트씩 증가하도록 하였다. 이로 인해 태그 간의 충돌 사이클을 줄이고 있으나 유휴 사이클의 증가를 가져왔다. HQT는 슬롯 지연 기법을 이용하고 있는데 슬롯 지연 기법이란 전달된 프리픽스와 일치하는 태그들이 즉시 응답하는 것이 아니라 특정시간 대기한 후 데이터를 전송하는 기법이다. 이러한 기법을 통해 HQT는 특정 비트 패턴의 존재 유무를 확인한다. 그림 2는 HQT의 동작과정을 나타내고 있다.

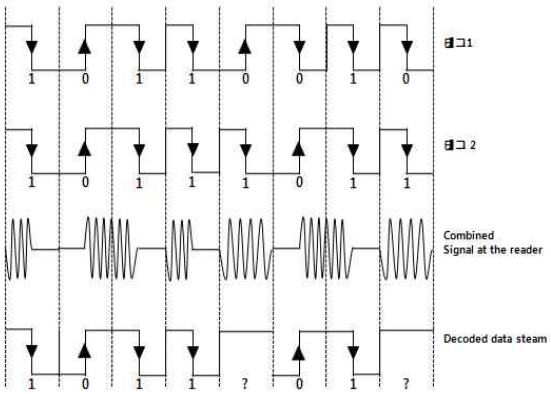


(그림 2) 하이브리드 쿼리 트리 알고리즘

HQT는 슬롯 지연 기법을 이용하여 유휴 사이클의 수를 줄이는 것을 시도하였다. 즉, 시작 위치와 완료 위치를 점검하여 불필요한 프리픽스의 확장을 줄이는 것이다. 그러나 이 방법에서는 busy time(바쁜 시간)만을 점검하기 때문에 중간에 보내지지 않은 슬롯의 경우 탐지가 불가능하다. 또한, 슬롯의 개념을 변형하여 한 슬롯 당 시간을 줄이고 있으나 질의 횟수를 늘려야 하는 문제를 발생시킨다. 예를 들어 “01010000”, “01010101”, “01011100”이 존재 한다고 가정하고, 리더가 “0101”의 프리픽스를 전파하면 태그는 각 슬롯에 응답한다. 이때, 만일 각 슬롯이 남은 아이디를 위한 충분한 시간을 확보하였다면 각 태그는 한 번에 인식할 수 있다. 그러나 시간과 질의 횟수를 단축하기 위한 방법으로 변형된 슬롯 알로하 방법을 고려함으로써, 두 개 이상의 유사한 태그가 존재할 때 서로 다른 슬롯에 할당 됨에도 불구하고 인식하지 못한다. HQT는 태그들의 바쁜 시간만을 점검하기 때문에 슬롯에서 충돌이 일어나면 정확한 태그 아이디를 알 수 있는 방법이 없다. 그로 인해 새로운 유휴 사이클을 생성한다.

2.4 완전 충돌 추적

RFID 시스템의 대표적인 디지털 부호화 방식에는 NRZ(NRZ, Non Return to Zero)방식과 맨체스터 코딩 방식이 있다[8]. NRZ 방식에서는 각 비트 구간에서 전송되어진 디지털 신호로 복호화 했을 때 값은 '0'혹은 '1'의 값을 가진다. NRZ 방식에서 한 비트 구간에서 '1'의 값이 하나라도 존재하게 되면 그 구간은 무조건 '1'을 가지게 된다. 따라서 충돌이 발생했는지 혹은 실제 '1'의 값만 전송되었는지 알 수 없다. 그러나 맨체스터 방식에서는 디지털 비트 구간에서 위상 전위가 일어나게 된다. 이 위상 변화에 따라 충돌 비트의 위치 정보를 알 수 있게 된다. 맨체스터 부호화 방식을 사용하게 되면 충돌 추적 알고리즘[7]중 Complete Collision Tracking(CCT, 완전 충돌 추적)방식을 사용할 수 있다. 이 CCT방식은 한 비트 구간내의 일정 구간동안 위상 변화가 감지되지 않으면 그 비트를 충돌 비트로 인식하게 된다. 그래서 태그 아이디내의 각 비트의 충돌 위치 정보를 사용할 수 있게 된다. 그림 3은 맨체스터 방식을 이용한 비트 충돌 검출의 예를 나타내고 있다.



(그림 3) 맨체스터 코딩 방식

태그1은 '10110011', 태그2는 '11111011'일 때 이 두 태그가 동시에 맨체스터 방식으로 리더에게 아이디전송을 하면 그림 3과 같이 2번째와 4번째 비트에서 위상 변화가 감지되지 않아 충돌로 인식하게 된다.

3. ETIA

본 논문에서 제안하는 Efficient Tag Identification Algorithm (ETIA, 효과적인 태그 인식 알고리즘)은 다수의 비트를 확장하기 위하여 변형된 타임 슬롯을 이용한다. 이 개선된 타임 슬롯에는 각 슬롯마다 충돌한 아이디의 예측이 가능한 형태로 태그를 그룹핑하여 전송한다. 이 방법에서는 기본 확장 비트를 3비트로 하며 3비트가 가질 수 있는 특정 비트 패턴을 이용하고 있다. 표 2는 이 방법에서 이용하고자 하는 충돌 비트 예측이 가능한 비트를 그룹 지어 놓은 것이다. 우선 3비트를 이용한 이유를 언급하고 ETIA에서 사용하는 방법에 대해서 설명한다. 표 2에서 살펴보면 2개의 비트를 확장한다고 했을 때 4개의 충돌 아이디를 검출하기위해 2개의 슬롯이 필요하고 3비트 확장에서는 3개의 슬롯으로 8개의 충돌 아이디 검출이 가능하다. 4개의 비트의 경우는 6개의 슬롯을 이용하여 최대 16개의 전송 아이디 검출이 가능하다. 그러나 4비트 확장을 사용하지 않은 이유는 6개의 슬롯을 사용했을 때 낭비되는 슬롯의 발생 수가 많기 때문이다. 특정 비트들의 부재로 인해 많은 빈 슬롯이 발생하게 되면 전체 인식시간에서 낭비되는 시간이 많이 발생하게 되기 때문이다. 물론 비트 패턴이 일정한 아이디들만 존재한다고 가정했을 때에는 좋은 성능을 나타내지만 실제로 비트가 랜덤하게 분포 되어있을 경우는 낭비되는 슬롯의 수가 많아지게 되고 인식시간이 늘어나게 되는 문제가 발생하게 된다.

표 2와 같이 3비트를 그룹 지어 전송할 경우 태그 아이디내의 '0'과 '1'의 위치에 따라서 충돌 위치가 달라진다. 가장 먼저 첫 번째 슬롯에 '000'과 '111'을 할당한다. 이 두 종류의 비트를 가진 태그가 첫 번째 슬롯에 응답을 하게 되면 슬롯은 충돌이 발생하게 될 것이다. 그러나 이 두 비트가 특정 슬롯에 할당되어 충돌이 발생하고 있기 때문에 이 두 태그의 상위 3비트의 충돌 비트는 예측 가능하게 된다. 또한 '001', '010', '100'의 비트는 비트 내 1의 위치에 따라 충돌이 달라진다. 만약 '001'과 '010'을 가진 두 태그에서 충돌이 발생하게 된다면 '0XX'의 충돌 비트를 가지게 된다. 이때 3비트 내 1을 가질 수 있는 자리 수는 1개이므로 '001'과 '010'의 비트를 예측 할 수 있는 것이다. 이와 마찬가지로 '011'과 '101', '110'은 0의 위치에 따라 충돌 비트가 달라진다. ETIA의 동작 방식은 쿼리 트리와 동일한 방식으로 진행된다. 가장 먼저 리더에서 태그로 'ε' 스트링을 전송하여 모든 태그의 응답

을 받게 된다. 이때 태그는 자신이 전송할 비트의 상위 3비트를 이용하여 표 2와 같은 형태로 '000'과 '111'의 비트는 슬롯1에 응답하게 되고 '001', '010', '100'의 비트는 슬롯2에 그리고 '110', '101', '011'은 슬롯3에 응답하게 된다. 각 슬롯에 응답한 태그는 해당 슬롯의 충돌 여부에 따라 비트를 예측하고 큐에 저장한다. 그리고 큐에 저장된 프리픽스를 가져와 태그에게 다음 질의를 보내게 된다. 프리픽스를 전송받은 태그는 자신의 아이디와 프리픽스가 일치하면 프리픽스 다음 비트부터 마지막 비트까지 리더에게 전송하는데 이때 프리픽스 다음의 3비트를 이용하여 응답슬롯을 선택하게 되는 것이다

<표 2> 충돌 비트 예측이 가능한 비트의 그룹

2비트	3비트	4비트	
			1010
			0101
	000		
	111	0000	1001
00		1111	0110
11	001		
	010		
	100	0001	1100
10		0010	0011
01		0100	
	011	1000	0111
	101		1011
	110		1101
			1110

그림 4는 리더와 태그의 동작 알고리즘을 의사코드로 나타낸 것이다. 태그는 리더로부터 P길이 만큼의 프리픽스를 받아 자신의 아이디와 일치여부를 확인하고 프리픽스가 일치한다면 ID_{P+1}부터 ID_{P+3}까지 비트를 더해 3으로 나눈 나머지 값을 C에 저장한다. 이 C값에 따라 전송 슬롯이 결정되며 C값만큼 딜레이 타임을 가진 후 리더에게 아이디를 전송한다.

리더는 모든 태그의 인식을 위해 태그에게 프리픽스를 전달하고, 태그로부터 전달 받은 데이터를 분석하여 태그의 아이디를 자동으로 인식하는 부분이다. 리더는 질의, 분석, 저장 3단계를 거친다. 질의 단계는 리더가 인식된 프리픽스를 큐로부터 얻어 태그에게

```
Algorithm 1. Tag Operation
 P = prefix received from a reader
 Lp=length of prefix P
 C  = 0
 If(P==ID[1..Lp]){
 C = Sum(Lp+1,Lp+2,Lp+3)%3
 }
 Else
 Return;
   // 아이디를 전송할 스롯의 시간까지 대기
  Delay (C * td)
  // 매칭된 프리픽스 이후 전 데이터를 전송
 Response(TagIDp+1..TagIDLength)

Algorithm 2. Reader Operation
PushQueue("ε")
DoWhile(!isQueueEmpty()){
 Prefix  =  get a prefix from Queue
 Propagate the prefix to tags  /*질의 단계 */
 For intTimeSlot = 1 to 3  {     /*분석 단계 */
  R  = Receive(TagID)
   For i = 0 to length of R {
     If(AtChar(R,i)=='X')
       If(i<3){ //상위 3비트내에 충돌 발생
         If(intTimeSlot==1 or 2 )
           String = Predict TagIDi...i+3 using 1's position
         Else
           String = Predict TagIDi...i+3 using 0's position
         PushQueue(Prefix+String)
         break; }
          }
          Else {
            PushQueue(Prefix+String(R,i))
            break;   /*저장단계 */  }
         Else
             //충돌이 발생 하지 않으면 아이디인식
              Identification (Prefix+R)
        }}}
```

(그림 4) 효과적인 태그 인식 알고리즘

전파하는 과정이다. 자동인식을 시작하기 위해 null을 뜻하는 'ε'문자열을 큐에 저장한다. 리더는 이후 큐가 빌 때까지 반복적으로 작업을 수행한다. 먼저 큐에서 태그로 전달할 프리픽스를 얻고, 이를 태그에게 전파한다.

분석 단계는 전달할 프리픽스와 일치한 값을 갖는 태그들이 자신의 아이디 값을 전달하면, 이 데이터로부터 원하는 값을 얻는 과정이다. 이 과정에서 충돌의 위치를 파악하기 위해 맨체스터 코드를 사용한다. 또한 분석 단계는 저장 단계와 인접하여 발생한다. 전달

된 데이터는 여러 슬롯을 통해 전달된다. 이렇게 전달된 데이터는 충돌이 발생하더라도 태그가 전달한 3비

그림 5는 6-비트의 아이디를 가진 여섯 개의 태그를 인식하는 과정을 나타내고 있다. 알고리즘의 동작

리더->태그	쿼리	슬롯1	슬롯2	슬롯3	쿼리	슬롯1	슬롯2	슬롯3	쿼리	슬롯1	슬롯2	슬롯3
태그->리더	ε	태그1 (000001)	충돌 (X0X01X)	충돌 (XXXX0X)	001	유휴 사이클	태그2 (001010)	유휴 사이클	100	유휴 사이클	태그3 (100011)	유휴 사이클
태그1 (000001)		000001										
태그2 (001010)			001010				001010					
태그3 (100011)			100011								100011	
태그4 (011101)				011101								
태그5 (101000)				101000								
태그6 (110010)				110010								
큐		001 100	001 100 011 101 110	100 011 101 110	100 011 101 110	100 011 101 110	100 011 101 110	100 011 101 110	011 101 110	011 101 110	011 101 110	011 101 110

리더->태그	쿼리	슬롯1	슬롯2	슬롯3	쿼리	슬롯1	슬롯2	슬롯3	쿼리	슬롯1	슬롯2	슬롯3
태그->리더	011	유휴 사이클	유휴 사이클	태그4 (011101)	101	태그5 (101000)	유휴 사이클	유휴 사이클	110	유휴 사이클	태그6 (110010)	유휴 사이클
태그1 (000001)												
태그2 (001010)												
태그3 (100011)												
태그4 (011101)				011101								
태그5 (101000)						101000						
태그6 (110010)											110010	
큐	101 110	101 110	101 110	101 110	110	110	110	110				

(그림 5) ETIA의 동작 과정

트가 무엇인지 쉽게 인지할 수 있다.

만약 상위 3비트가 아닌 다른 비트에서 충돌이 발생하면, 충돌이 발생하기 이전 비트까지를 얻어 이전 태그에 전달한 프리픽스와 접목하여 큐에 저장한다.

과정은 여러 개의 스텝으로 구성된다. 각 스텝은 쿼리와 3개의 슬롯으로 이루어진다. 태그 인식 과정은 가장 먼저 리더에서 태그로 'ε'을 전송한다. 'ε'을 전송받은 모든 태그들은 아이디 상위 3비트를 이용하여 응

답 슬롯을 선택한다. 이때 스텝 1의 슬롯1에서 태그1만이 응답하였기 때문에 태그1은 슬롯1에서 인식된다. 슬롯2에서는 'X0X01X'의 충돌이 발생하였다. 슬롯2에서 충돌 값은 '1'의 위치에 따라 나타나게 된다. 그러므로 상위 3비트 내의 충돌은 '100'와 '001'의 비트로 인해 충돌이 발생했다는 것을 예측할 수 있다. 이 예측 값은 큐에 저장되고 다음 스텝의 프리픽스로 이용된다. 다음 슬롯3에서는 'XXXX0X'의 충돌 형태를 보였다. 이때 상위 3비트에서 모두 충돌이 발생하였다. 슬롯3에서는 0의 위치에 따라 충돌이 발생하는데 3비트 모두에서 충돌이 발생했다는 것은 '011', '101', '110'의 값이 모두 전송되었다는 것을 예측할 수 있다. 이 값 또한 큐에 저장되고 다음 스텝으로 진행된다. 다음 스텝에서는 '001'이 프리픽스로 보내지고 '001'와 같은 아이디를 가지는 태그2가 '001' 다음의 '010'를 이용하여 슬롯2에 자신의 아이디를 전송하게 된다. 이렇게 6개의 스텝을 거치면 6개의 태그가 모두 인식된다. 여섯 개 태그를 모두 인식하기 위해서는 QT와 HQT모두 11개의 쿼리가 요구된다.

ETIA 방식은 충돌 사이클을 줄임으로서 트리 노드의 수를 줄이고 실제 존재하지 않는 아이디를 가지는 유휴 사이클은 생성 되지 않는다. 또한 태그의 아이디가 특정 그룹으로 분류되어 있는 경우 좋은 성능을 발휘하는 것으로 성능평가에서 나타났다.

4. 성능평가

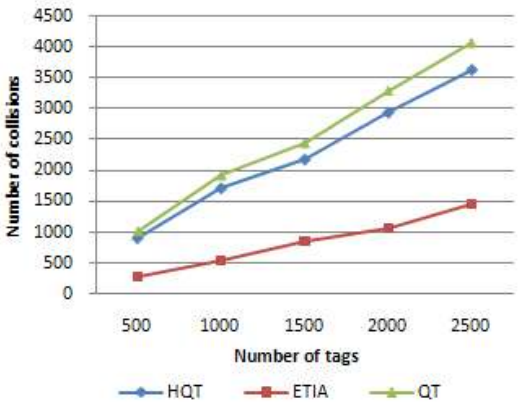
논문에서 제안하는 방법의 성능평가를 위해 C# 언어로 시뮬레이션 프로그램을 설계하였다. 그리고 QT, HQT 알고리즘과 본 논문에서 제안하는 ETIA를 비교해 보았다. 성능평가 항목은 질의 응답 수와 충돌 수이다. 실험에서 사용한 아이디의 길이는 EPC global의 표준 중에 하나인 96bit 태그를 사용하였다.

96bit 태그는 Header(8bit), EPC Manager(28bit), Object class(24bit), Serial number(36bit)로 구성되어 있다. Header는 데이터 유형 및 길이를 정의하고, EPC Manager는 상품에 대한 분류와 일련번호를 관리하는 기관이나 기업을 표시한다. Object Class는 바코드의 상품 품목 코드에 해당하는 것으로 각 품목 또는 단위를 표시하고 Serial Number 는 각 상품들에 대하여 부여되는 고유한 식별번호를 나타낸다.

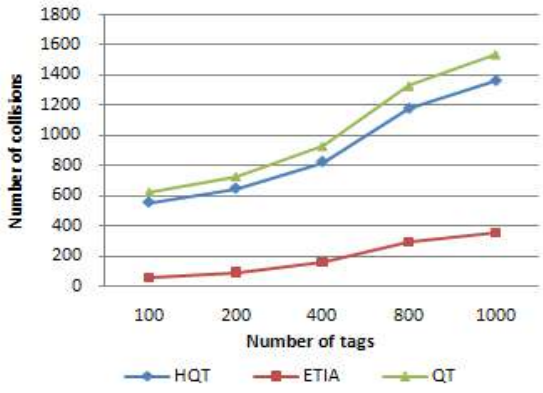
실험에서 실제 환경과 유사성을 고려하기 위해 Header, EPC Manager, Object class 필더를 무작위로 10개를 생성하고 Serial Number 필더는 두 가지 방법으로 생성한 후 접목하였다. 첫 번째 방법은 36 Ob중 랜덤하였을 선택하고 1씩 증가 시키면서 태그를 생성하는 방법이고, 두 번째 방법은 Serial Number 필더를 다시 임의로 생성하는 것이다.

Serial Number 필더를 제외한 나머지 비트가 같다는 것은 동일 제품 코드를 나타내는 것이며 Serial Number가 순차적으로 있는 것은 물류 창고에 동일 제품이 여러 개 쌓여있는 것을 의미한다.

그림 6은 태그 충돌 수를 나타내고 있다. 순차적 할당이나 랜덤할당에 있어 본 논문에서 제안하는 방법의 충돌 수가 다른 알고리즘에 비해 확연히 줄어 든 것을 알 수 있다.



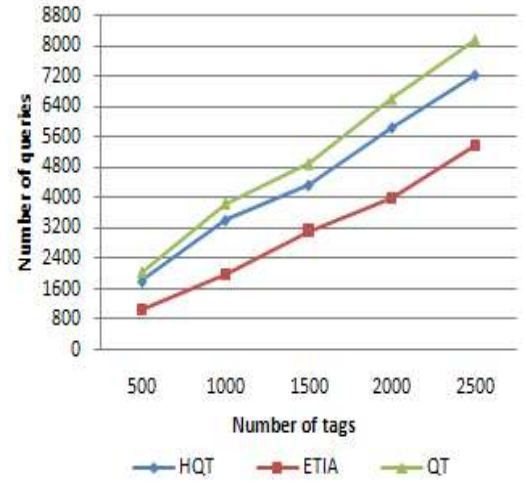
(a) Serial Number 필더의 랜덤 할당



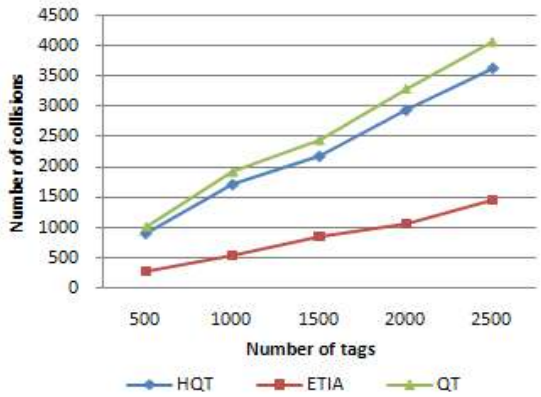
(b) Serial Number 필더의 순차적 할당

(그림 6) 태그 아이디 충돌 수

그림 7은 전체 태그를 인식하기 위해 필요한 쿼리 수를 나타내고 있다. 제안 알고리즘에서는 한 번의 쿼리로 확장할 수 있는 태그의 비트 수가 다른 알고리즘에 비해 더 많기 때문에 결국 태그를 인식하기 위한 쿼리 수 또한 줄어드는 것이다. 전체 성능을 평가해본다면 태그 인식 쿼리 수와 충돌 수에서 최소 2배에서 최대 10배 가까이 향상된 것을 알 수 있다.



(a) Serial Number 필터의 랜덤 할당



(b) Serial Number 필터의 순차적 할당

(그림 7) 태그 인식을 위한 전체 쿼리 수

5. 결 론

RFID 시스템은 다수의 태그를 리더가 빠르게 인식할 수 있는 알고리즘이 필요하다. 기존 트리기반 알고리즘은 아이디 확장 과정을 거치면서 유휴 사이클이 늘어나게 되어 불필요한 질의 응답과정이 생긴다. 본 논문에서는 이러한 문제점들을 개선하여 태그확장을 빠르게 할 수 있는 개선된 방법을 제안하였다.

제안 방법은 태그의 아이디 비트 패턴을 이용하여 특정 응답 슬롯을 사용하도록 하였다. 각 비트 패턴은 충돌 예측이 가능한 상태를 가지고 있다. 따라서 충돌이 슬롯 내에서 발생하더라도 상위 응답 3비트 아이디는 예측이 가능하다. 제안 알고리즘은 다른 알고리즘에 비하여 비트 확장 속도가 빠르고 전체 쿼리 수도 줄여준다. 또한 특정 비트끼리 강제 충돌을 일으킴으로서 전체 충돌 횟수도 줄게 되며 충돌이 일어났다 하더라도 예측 가능한 비트 상태가 존재하게 된다.

제안한 ETIA 방식은 기존의 알고리즘 QT의 간단한 동작 방식과 HQT의 슬롯 방식의 장점을 하이브리드하여 태그인식의 성능을 개선하였다. 특히 태그가 응답 슬롯을 결정하는 과정에서 태그가 전송할 앞부분의 3비트를 이용하여 전송함으로 충돌 비트가 발생하여도 이를 예측 할 수 있도록 하였다. 이로 인해 충돌 비트를 다시 쿼리하지 않아 전체 쿼리의 수가 줄어들게 된다. 시뮬레이션 결과에서도 알 수 있듯이 특히 태그가 특정 그룹으로 분류되어있고 태그의 아이디 부여가 순차적인 경우 쿼리 수와 충돌 수가 다른 알고리즘에 비해 크게는 평균 3배 정도, 작게는 1.5배 정도 쿼리의 수가 줄어들어 태그를 빠르게 인식할 수 있다.

그러나 실제 응답하지 않은 슬롯도 요구 되고 있기 때문에 빈 슬롯이 발생한다. 그래서 향후 낭비되는 빈 슬롯을 줄일 수 있는 동적 슬롯을 가지는 알고리즘의 연구가 필요하다.

참 고 문 헌

[1] EPCTM Radio-Frequency Identification Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz Version 1.0.8, EPCglobal, December 2004.

[2] 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, Final Draft International Standard ISO 18000-6, November 2003.
[3] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. John Wiley & Sons, 2003.
[4] T. Wang, "Enhanced binary search with cut-through operation for anti-collision in RFID systems," IEEE Commun. Lett., Vol. 10, No.4, pp.236-238, April 2006.
[5] J. Chio, D. Lee, and H. Lee, "Query tree-based reservation for efficient RFID tag anti-collision," IEEE Commun. Lett., Vol, 11, No. 1, pp. 85-87, April 2006.
[6] C. Law, K. Lee, and K. Sju, "Efficient Memoryless Protocol for Tag Identification", In Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication, ACM, pp. 75-84, August 2000.
[7] J. Ryu, H. Lee, Y. Seok, T. Kwon and Y.Choi, "A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID system,", IEEE International Conference on Communications, pp 5981-5986, June 2007
[8] C. Quan, "Design and Performance Evaluation of High Performance Anti-collision Algorithm in the RFID System," Ph.D. thesis. Dept. of Computer & Communications Engineering, Daegu Univ., December 2004.
[9] J. Cho, J. Shin, and S. Kim, "RFID Tag Anti-Collision Protocol: Query Tree with Reversed IDs", 10th International Conference on Advanced Communication Technology, IEEE Communications Society, pp 225-230, February 2008.

김 태 희 (Tae Hee Kim)

- 2006년 대구대학교 전자계산학과 학사
- 2009년 경북대학교 대학원 전자전기컴퓨터학부 공학 석사
- 관심분야: 유비쿼터스, RFID, Embedded System

김 선 경 (Sun Kyung Kim)

- 정회원
- 1979년 이화여자대학교 수학과 학사
- 1982년 한국과학기술원 전산학과 석사
- 1991년 미국 Minnesota대학교 전산학과 박사
- 1992년~현재 대구대학교 컴퓨터 IT공학부 교수
- 관심분야: 과학계산, 병렬처리, 알고리즘, RFID

논 문 접 수 일 : 2010년 03월 18일
1차수정완료일 : 2010년 06월 30일
게 재 확 정 일 : 2010년 06월 30일