900Mhz RFID 기반의 충돌방지 알고리즘

정회원 이 준 혁*, 정 원 수**, 오 영 환***

The Anti-Collision Argorithm for the 900MHz RFID

Jun-Hyuk Lee*, Won-soo Jung**, Young-hwan Oh*** *Regular Members*

요 약

RFID(Radio Frequency Identification) 충돌방지 알고리즘이란 RFID 리더 영역 내에 존재하는 다수의 태그를 태그 간의 통신 간섭에 의한 충돌을 회피하며 다수의 태그를 고속으로 인식하는 기술이다. 충돌방지 알고리즘은 RFID 시스템의 성능과 안정성을 결정하는 핵심 기술로 중요성이 높다. 본 논문은 결정적 알고리즘의 대표적인 방식인 쿼리트리(Query Tree) 알고리즘의 성능을 개선한 방식의 충돌 방지 알고리즘을 제안한다. 제안한 충돌방지 알고리즘은 질의 및 응답 비트 수를 줄임으로서 태그 인식 속도를 개선한다.

Key Words : RFID, Query Tree Algorithm, Anti-Collision Algorithm, EPCglobal

ABSTRACT

RFID Anti-Collision algorithm is needed to avoid collision problem caused by radio interference between tags in the RFID reader area. It decide the performance and reliability of the RFID system. This paper propose the Anti-Collision algorithm of the deterministic algorithm that it improve the QT algorithm. The proposed Anti-Collision algorithm increases identification speed of tag since it reduce the number of query and response bit

Ⅰ. 서 론

최근 유비쿼터스 컴퓨팅을 실현하기 위한 핵심 기술로서 RFID(Radio Frequency Identification) 기술이 주목 받고 있으며 RFID 기술을 도입하기 위한 연구가 다각도로 진행 중에 있다. RFID 방식은 접촉하지 않고 태그의 정보를 판독하거나 인식하는 무 접촉 객체인식 기술로서 물류, 국방, 교통, 의료, 환경 분야에서 관련 연구가 다양하게 진행되고 있다[1, 2].

RFID 시스템에서 사용하는 무선 주파수는 활용 분야에 따라 다르며 125KHz, 13.56MHz, 433.92MHz, 860~960MHz, 2.45GHz 등을 사용하고 있다. RFID 시스템에서 태그 식별은 리더가 물품에 부착된 태그에 일정 주파수 신호를 안테나를 통해 방사하면 리더에 인접한 태그는 수신된 신호를 변경하여 반사파의 형태로 재전송하는 응답과정을 통하여 수행된다. 이 때 리더의 식별영역 내에 한 개의 태그만이 존재하는 경우 태그 식별은 간단하게 처리될 수 있으나 다수 개의 태그가 존재할 경우에는 여러 개의 태그가 통시에 리더에 응답하기 때문에 리더에서 태그들 간의 충돌이 발생하게 된다. 이러한 충돌은 리더로 하여금 정확한 태그 식별을 방해하는 원인이 되며, 특히 대량의 물품을 실시간으로 식별해야 하는 대규모 전자물류 시스템 등에 적용하기 위해서는 다중 태그 식별을 효과적으로 처리하기 위한 충돌방지 알고리즘이 필수적으로 요구된다.

RFID 시스템에서 충돌방지 알고리즘은 다수의 태

* 한국정보통신대학 정보통신설비과 (jhlee@icpc.ac.kr), ** 광운대학교 전자통신공학과 (sootan77@dreamwiz.com)
*** 광운대학교 전자통신공학과(yhoh@daisy.kw.ac.kr)
논문번호 : 07078-1018, 접수일자 : 2007년 10월 18일

그 또는 리더간의 동시 접속 문제를 해결한다는 개념적인 측면에서 기존 무선 통신의 다중접속 기술과 유사하다. 그러나 RFID 시스템에서 충돌 방지 알고리즘은 태그의 저비용·저전력·초소형화 정책으로 인하여 기존의 무선 통신에서 모바일 장치에 비해 태그의 데이터 전송능력이 제한된다. 이러한 차이점으로 인해 기존 통신시스템에서 사용하는 다중접속 방법을 RFID 충돌방지 알고리즘에 곧바로 적용시킬 수는 없다[3].

충돌방지 알고리즘은 크게 확률적(Probability) 알고리즘과 결정적(Deterministic) 알고리즘이 있으며, 결정적 알고리즘 방식에 관하여 살펴보도록 한다. 결정적 알고리즘은 기억형(Memory) 알고리즘과 무기억형(Memoryless) 알고리즘 방식이 있다. 무기억형 알고리즘은 이진 트리 검색 알고리즘, 쿼리트리(Query Tree) 알고리즘, 충돌 추적트리 알고리즘 방식이 있으며, 태그의 응답이 리더의 질의에 의해서만 결정되는 특징 때문에 태그의 구현이 비교적 간단하여 태그를 설계할 경우 발생하는 저비용·저전력·초소형화 문제를 해결할 수 있다.

무기억형 알고리즘의 대표적인 방법인 쿼리트리 알고리즘은 태그 간 충돌이 발생하는 경우 단순 충돌 여부만을 판별하기 때문에 리더에서 중복되는 질의를 태그에 전송하는 문제가 발생한다. 또한 태그ID를 설계하는 경우 태그는 일정한 구조의 데이터 타입을 유지한다. 리더에서 큐에 아무런 질의가 없는 경우 리더는 null 에 해당하는 'ϵ'을 브로드캐스트 하고 모든 태그는 'ϵ' 받은 후 자신의 ID를 리더에 전송한다. 이 경우 리더에서 충돌이 발생하는데 태그ID의 데이터 타입을 무시하고 획일 적으로 큐에 0, 1을 추가 하는 것은 불필요한 질의 메지시를 발생시키는 문제점이 있다[4, 5, 6, 7].

본 논문에서는 이러한 문제점을 해결하기 위해서 리더에서 'ϵ' 값을 전송 후 태그ID를 전송 받는 경우 최초 충돌 비트의 위치를 파악하여 큐에 최초 충돌 비트 이전 비트에 0, 1 값을 추가하여 프리픽스를 생성하는 방법과 충돌이 발생한 비트의 개수와 위치등 보다 세밀한 정보를 추출하고 이를 활용하여 인식 성능을 개선하는 방안을 제안한다.

II장에서는 RFID 충돌 방지에 관한 관련 연구에 관하여 알아보고 III장에서는 제안한 알고리즘에 관하여 알아보고 IV장에서는 성능 평가에 관하여 알아본다. 마지막으로 V장에서 결론 및 향후 연구 방향으로 끝을 맺고자 한다.

Ⅱ. 기반 기술

2.1 RFID 기반의 충돌 방지 알고리즘

다중태그 식별을 위한 충돌방지 알고리즘과 관련하여 무선 통신 환경에서의 채널 다중접근과 관련된 많은 연구가 이루어지고 있다. 그러나 RF에서 요구되는 통신 방식은 기존의 무선통신과는 전력공급, 연산능력, 태그 상호간의 존재여부와 영역내의 태그 수를 알 수 없는 등 많은 점에서 다르다. 다중태그 식별을 위한 충돌방지 알고리즘은 크게 트리 기반의 결정적(Deterministic) 알고리즘과 슬롯 알로하 기반의 확률적(Probability) 알고리즘으로 구분할 수 있다.

결정적 알고리즘은 이진비트로 표현된 태그 식별자들로 이진트리를 구성한 후 그 트리의 노드를 순회하며 태그 식별을 수행하는 방법으로 태그 식별 과정이 예측 가능하다는 특징을 갖고 있다. 확률적 알고리즘은 알로하 프로토콜에 기반을 두고 있다. 리더의 식별영역내의 태그들은 주어진 N개의 슬롯에서 태그의 정보를 전송할 슬롯을 임의로 선정하여 해당 식별자를 전송하게 되므로 슬롯간의 시간차에 의해 태그충돌을 피한다. 그러나 식별영역내의 태그개수를 정확히 파악하기 어렵기 때문에 적정한 슬롯개수와 종료시점을 확률적으로 계산해야 한다. 따라서 확률에 근거한 종료시점의 결정으로 태그식별의 완전성을 지원하지 못하며 또한 충돌이 발생한 슬롯의 재전송으로 인하여 태그 식별시간에 있어서 높은 성능을 기대하기 어렵다는 단점을 갖는다. 표 1은 충돌 방지 알고리즘의 분류를 나타낸다.

2.2 쿼리트리 알고리즘

쿼리트리 알고리즘은 k 비트의 크기를 갖는 문자열을 가능한 최적의 방법으로 탐색함으로서 영역내의 모든 태그를 식별하는 방법이다. 각 태그는 프리

표 1. 충돌방지 알고리즘 분류

충돌방지 알고리즘 분류		
결정적 알고리즘	기억형 알고리즘	분할트리 알고리즘 비트중재 알고리즘
	무기억형 알고리즘	트리워킹 알고리즘 쿼리트리 알고리즘 충돌추적트리 알고리즘
확률적 알고리즘	ID-슬롯형 알고리즘	I-Code 알고리즘 STAC(Auto-ID) 알고리즘
	비트-슬롯형 알고리즘	비트-슬롯 알고리즘

픽스 매칭(Prefix Matching)을 위한 회로만이 필요하기 때문에 리더와의 통신에 사용되는 메시지가 매우 간단하다는 장점을 갖는다. 태그 식별과정은 계층적으로 이루어지며 쿼리트리(Query Tree) 형태로 표현되기 때문에 쿼리트리 알고리즘이라고 불린다. 쿼리트리 알고리즘의 동작 과정은 아래와 같다.

먼저 태그의 식별코드 길이를 k 비트로 가정한다. A를 최대 길이가 k 비트인 이진 문자열의 집합이라 하고, w를 태그의 식별코드 문자열이라 하면, 집합 A와 문자열 w는 다음과 같이 정의된다.

$$A = \cup_{i=0}^{k} \{0,1\}^{i} \quad (1)$$

$$w = w_1 w_2 \cdot \cdot \cdot w_k \quad (2)$$

리더의 상태(Q, M)는 A의 문자열로 구성된 일련의 문자열인 Q(Queue)와 A의 원소들로 구성된 문자열의 집합인 M(Memory) 으로 구성된다. Q는 리더가 다음에 질의할 질의 문자열을 저장하는 용도로 사용되고, M은 이미 식별된 태그의 식별코드를 저장하는 용도로 사용된다. 리더가 브로드캐스트 하는 질의는 A에 있는 문자열 $q(q \in A)$로 정의한다. 응답코드는 태그의 식별코드 중에서 질의 문자열을 제외한 나머지 비트들로 구성된 문자열 r이다. r은 다음과 같이 정의 된다.

$$r = r_{|q|+1} r_{|q|+2} \cdots r_k \quad (3)$$

<Reader>

초기 상태에서 리더의 큐 Q는 $<\epsilon>$이며, 여기서 'ϵ' 모든 태그가 응답하는 질의 문자열이다. M은 비어 있는 상태로 초기화 한다.

1) 큐를 $Q = <q_1, q_2, \cdot \cdot \cdot, q_l>$로 둔다.
2) 질의 q_1을 Q에서 꺼내서 브로드캐스트 한다.
3) Q를 $Q = <q_2, \cdot \cdot \cdot, q_l>$로 갱신한다.
4) 태그로부터 수신한 응답에 대하여,
   ① 응답이 r이면 $q_1 \cdot r$ 을 식별한 것으로 표시하고 M에 삽입한다.
   ② 만일 충돌이 발생하는 경우 Q를 $<q_2, \cdot \cdot \cdot, q_l, q_1 0, q_1 1>$로 갱신한다.
5) Q가 비어있을 때까지 위의 과정을 반복한다.

리더는 질의가 저장된 Q에서 질의 문자열 'ϵ' 가져와 태그에게 브로드캐스트 한다. 리더는 질의 후 태그의 응답에 따라 세 가지 상태로 반응한다. 첫째, 태그로부터 아무런 응답을 받지 못하는 경우에는 Q에 저장된 질의 문자열을 가져와 태그에게 브로드캐스트 한다. 둘째, 하나의 태그로부터 응답을 받은 경우, 하나의 태그만 인식된 것이므로 리더는 응답코드와 질의 문자열을 조합하여 식별코드를 만들어 M에 저장한다. 그 후 다른 태그를 인식하기 위해여 알고리즘을 반복 수행 한다. 셋째, 두 개 이상의 태그가 인식되는 경우 리더는 충돌 사실을 인지하고 기존의 질의 문자열에 '0' 과 '1'을 추가하여 각각 Q에 저장한 후 새로운 질의 문자열을 Q에서 가져온 후 알고리즘 을 다시 수행 한다. 이러한 과정은 영역내의 모든 태그가 식별될 때까지 반복된다.

<Tag>

1) 태그의 식별코드 w를 $w = w_1 w_2 \cdot \cdot \cdot w_k$라 한다.
2) 리더로부터 수신한 질의를 q라 하고 q의 길이를 $|q|$라 한다.
3) $q = \epsilon$ 또는 $q = w_1 w_2 \cdot \cdot \cdot w_{|q|}$이면 태그는 $r = r_{|q|+1} r_{|q|+2} \cdots r_k$ 로 응답한다. 즉 질의 q가 자신의 식별코드 처음 비트들과 일치하면 식별코드 중에서 질의 문자열을 제외한 나머지 비트들을 리더로 보낸다.

그림 1은 쿼리트리 알고리즘의 태그상태 천이도이다.

S0 상태에 있는 태그가 리더로부터 질의 문자열을 수신하면 S1 상태로 천이한다. S1 상태에서 수신하는 질의 문자열이 태그의 식별코드와 일치하지 않으면 S0 상태로 천이하며 다음 라운드의 질의를 기다린다. 반면 태그의 식별코드가 질의 문자열과 일치하는 태그는 S2 상태로 천이하여 식별코드 중에서 질의 문자열을 제외한 나머지 비트들을 리더로 보낸 후 S0 상태로 천이하여 다음 라운드의 질의를 기다린다.

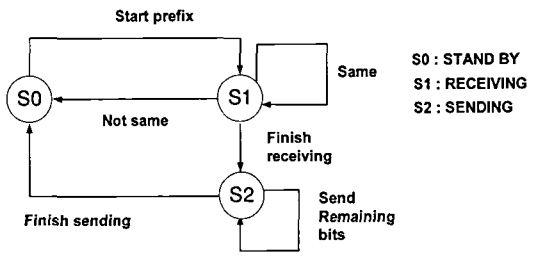


그림 1. 쿼리트리 알고리즘의 태그 상태 천이도

표 2. 쿼리트리 알고리즘의 동작

질의 문자열	태그 응답	응답한 태그	Q	M
ϵ	0XXX (collision)	0001, 0010, 0100, 0110	0, 1	
0	0XXX (collision)	0001, 0010, 0100, 0110	1, 00, 01	
1	no response		00, 01	
00	XX (collision)	0001, 0010	01, 000, 001	
01	X0 (collision)	0100, 0110	000, 001, 010, 011	
000	1	0001	001, 010, 011	0001
001	0	0010	010, 011	0001, 0010
010	0	0100	011	0001, 0010, 0100
011	0	0110		0001, 0010, 0100, 0110

표 2는 식별코드가 <0001, 0010, 0100, 0110> 인 네 개의 태그를 식별하기 위한 쿼리트리 알고리즘의 동작을 나타낸 것이다. 표에서 태그 응답에서 'X'는 충돌이 발생한 비트를 의미한다.

표 2에서 나타낸 바와 같이 네 개의 태그를 모두 식별하기 위하여 총 9번의 질의 라운드가 필요하다. 이때 리더가 방송하는 질의 문자열의 총 비트 수는 16 비트이고, 모든 태그들이 전송하는 총 비트 수는 36 비트이다.

Ⅲ. 제안한 충돌 방지 알고리즘

쿼리트리 알고리즘은 태그 인식 과정에서 1 비트라도 충돌이 발생할 경우 전체 알고리즘에서 충돌이 발생했다고 인식한다. 즉 1 비트에서 충돌이 발생하거나 아니면 전체 비트에서 충돌이 발생하던지 간에 리더는 단순히 이를 충돌로 인식하고 새로운 질의 문자열을 생성하게 된다. 또한 EPC 태그의 데이터 구조를 살펴 볼 때 다수의 태그를 동시에 인식하는 경우 동일한 EPC Manager나 Object Class를 가질 확률이 매우 높다. 그러나 쿼리트리 알고리즘의 경우 EPC 태그의 데이터 구조를 무시하고 충돌이 발생할 경우 질의 문자열의 '0'과 '1'을 추가하여 큐에 저장하게 된다. 이는 리더와 태그의 질의·응답 횟수를 증가 시켜 태그 인식의 효율성을 떨어뜨릴 수 있다.

제안하는 충돌 방지 알고리즘은 리더에서 질의 문자열을 전송한 후 태그를 인식하는 과정에서 충돌이 발생하는 경우 충돌이 발생한 비트의 수와 처음 충돌이 발생한 비트의 위치를 파악하여 리더와 태그의 질의·응답 횟수를 줄여 태그 인식 속도를 개선한 알고리즘이다. 제안한 알고리즘을 수행하기 위한 가정은 다음과 같다.

- 리더에서 충돌이 대한 상황을 인식할 수 있어야 한다. 리더는 태그의 응답에 따른 충돌이 발생한 비트의 위치와 비트 수를 인식 할 수 있어야 한다. 리더는 적절한 코딩 기법을 사용함으로써 태그로부터 전송되는 응답 코드의 비트 구분이 가능하다. 일반적으로 널리 알려진 트리 기반의 결정적 알고리즘은 충돌이 발생하는 비트 위치를 리더에서 인지할 수 있음을 전재로 하고 있다. 예를 들면 EPCglobal에서 발표한 Class0의 대표적 충돌방지 알고리즘인 이진 트리 알고리즘은 두 서브 캐리어 톤(Data '0' for 2.2MHz, Data '1' for 3.3MHz)를 사용함으로써 각 비트를 구분하고 있다.
- 실제 충돌은 태그 응답 코드의 일부에서 발생할 확률이 크다. 다수의 충돌 비트가 넓은 범위에 분포한다면 충돌 비트를 인식할 때 마다 처리하는 알고리즘은 쿼리트리 알고리즘에 비해 성능이 떨어질 수 있다. 태그를 읽는 과정을 줄이기 위해서는 태그 데이터 포맷을 설계할 때 공통 되는 부분을 앞쪽에 배치되도록 설계함으로써 리더와 태그의 질의·응답을 위한 질의 문자열을 최대로 하여 태그의 특정 영역에서 충돌이 발생할 수 있도록 유도하여 태그의 인식률을 높일 수 있어야 한다.

제안하는 충돌 방지 알고리즘은 리더에서 질의 문자열로 'ϵ'을 전송한 후 식별 코드를 수신하는 과정에서 첫 번째 충돌 비트의 위치를 파악하여, 첫 번째 충돌 비트의 전까지의 문자열을 질의 문자열

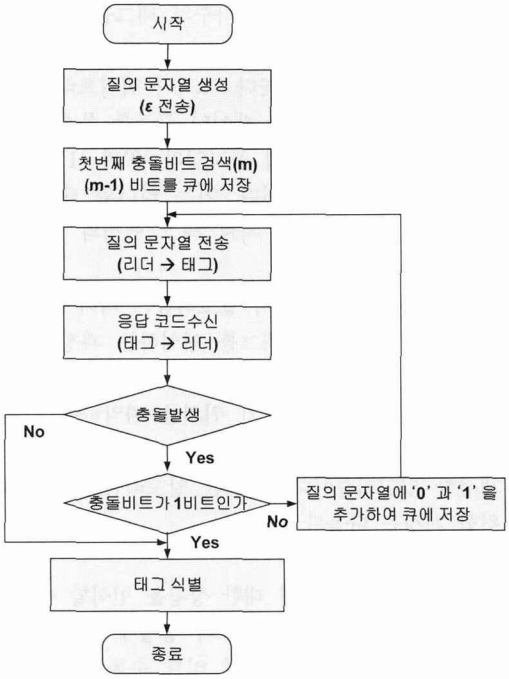


그림 2. 제안한 충돌방지 알고리즘 순서도

로 생성하여 큐에 저장한다. 또한 충돌한 비트의 수가 1개이면 해당 충돌 비트에 각각 '0'과 '1'의 값을 설정한 뒤 메모리에 저장한다. 제안한 충돌방지 알고리즘은 순서도는 그림 2와 같다.

제안한 충돌 방지 알고리즘의 동작 과정은 아래와 같다.

<Reader>

초기 상태에서 리더의 큐 Q는 $<\epsilon>$이며, 여기서 'ϵ' 모든 태그가 응답하는 문자열이다. 메모리 M은 비어 있는 상태로 초기화 한다.

1) 'ϵ' 값을 전송 후 식별 코드의 첫 번째 충돌 비트 위치를 'm'라 하고, 문자열의 길이를 $|m|$라 한다.
2) 큐를 $Q=<q_1 q_2 \cdots q_{|m-1|}, q_{|m|}, \cdot \cdot \cdot, q_l>$로 둔다.
3) 질의 $q_1 q_2 \cdots q_{|m-1|}$ Q에서 꺼내서 브로드캐스트 한다.
4) Q를 $Q=<q_{|m|}, \cdot \cdot \cdot, q_l>$로 갱신한다.
5) 태그로부터 수신한 응답에 대하여,
   ① 응답이 r이면 $q_1 q_2 \cdots q_{|m-1|} \cdot r$을 식별한 것으로 표시하고 M에 삽입한다.
   ② 1 비트 충돌이 발생하는 경우, 충돌 비트의 위치를 z 라고 하면, 문자열의 길이는 $|z|$이다. 즉 1 비트에서 충돌이면 태그의 식별코드 $q_1 q_2 \cdots q_{|m-1|} \cdot r_{|q|+1} r_{|q|+2} \cdots r_k \wedge r_{|z|-1} 0$ 과 $q_1 q_2 \cdots q_{|m-1|} \cdot r_{|q|+1} r_{|q|+2} \cdots r_k \wedge r_{|z|-1} 1$을 식별한 것으로 표시하고 M에 삽입한다.
   ③ 2 비트 이상 충돌이 발생하는 경우 Q를 $<q_{|m|}, q_{|m+1|}, \cdots, q_l, q_1 q_2 \ldots q_{|m-1|} 0, q_1 q_2 \cdots q_{|m-1|} 1>$로 갱신한다.
6) Q가 빌 때까지 위의 과정을 반복한다.

표 3은 제안한 충돌 방지 알고리즘의 슈도코드를 나타낸 것이다.

표 4는 식별코드가 <0001, 0010, 0100, 0110> 인 네 개의 태그를 식별하기 위한 제안한 충돌 방지 알고리즘의 동작을 나타낸 것이다. 표에서 태그 응답에서 'X'는 충돌이 발생한 비트를 의미한다.

표 3. 제안한 충돌 방지 알고리즘의 슈도코드

```
In the Reader

Begin
  Q ← 'ε'  // 질의 문자열 Q 초기화
  M =< >  // 인식된 태그를 저장할 메모리
 While
   질의 문자열 'ε'을 태그로 전송
   태그로부터 응답을 수신
   If(질의 문자열이 'ε'인 경우)
       첫 번째 충돌비트 위치 파악
       Q ← Put() // 큐에 첫 번째 충돌 비트 전 전까지의 문자열을
                    질의 문자열로 저장
   If (응답이 없는 경우)
      If (큐가 비어 있음)
         알고리즘 종료
      Else
         Q ← Get() // 큐에 저장된 질의 문자열을 꺼냄
   Else If (태그가 한 개만 응답하는 경우)
      M ← Put() // 메모리에 태그 ID를 저장
      If (큐가 비어 있음)
         알고리즘 종료
      Else
         Q ← Get() // 큐에 저장된 질의 문자열을 꺼냄
   Else (충돌 발생)
      If (한 비트 충돌 발생 )
         M ← Put() // 메모리에 한 비트 충돌한 태그 ID 저장
         If (큐가 비어 있음)
            알고리즘 종료
         Else
            Q ← Get() // 큐에 저장된 질의 문자열을 꺼냄
      Else (두 비트 이상 충돌 발생)
         Q ← Put() // 현재 질의 문자열에 각각 '0'과 '1'을 추가하여
                      큐에 저장
  End While
 End
```

```
In the Tag

Begin
 리더기로부터 질의 문자열을 수신
 If (질의 문자열이 q = ε   q = w1 w2 ··· wq 인 경우)
   응답 코드 r을 리더로 전송;
 End;
End;
```

표 4. 제안한 충돌방지 알고리즘의 동작

질의 문자열	태그 응답	응답한 태그	Q	M
ϵ	0XXX (collision)	0001, 0010, 0100, 0110	00, 01	
00	XX (collision)	0001, 0010	01, 000, 001	
01	X0	0100, 0110	000, 001	0100, 0110
000	1	0001	001	0100, 0110, 0001
001	0	0010		0100, 0110, 0001, 0010

표 5. 예를 통한 두 알고리즘 비교

알고리즘	질의 라운드 수	질의 비트 수	응답 비트수
쿼리트리 알고리즘	9	16	36
제안한 충돌방지 알고리즘	5	10	20

표 4에서 나타낸 바와 같이 네 개의 태그를 모두 식별하기 위하여 총 5번의 질의 라운드가 필요하다. 이때 리더가 방송하는 질의 문자열의 총 비트 수는 10 비트이고, 모든 태그들이 전송하는 총 비트 수는 20 비트이다.

표 5는 4개의 태그를 식별하기 위하여 표 2와 표 4에서 각각 나태 낸 쿼리트리 알고리즘과 제안한 충돌방지 알고리즘의 동작 과정에서 얻어진 질의 라우드 수, 질의 비트 수, 응답 비트 수를 요약한 것이다. 표에서 나타낸 봐와 같이 본 논문에서 제안한 충돌 방지 알고리즘이 쿼리트리 알고리즘에 비하여 모든 성능 요소에서 우수함을 알 수 있다.

제안한 충돌 방지 알고리즘은 리더에서 질의 문자열로 'ϵ'을 전송한 후 식별 코드를 수신하는 과정에서 첫 번째 충돌 비트의 위치를 파악하여, 첫 번째 충돌 비트의 전까지의 문자열을 질의 문자열로 생성하여 큐에 저장하기 때문에 충돌이 발생 할 경우 태그의 데이터 구조와 상관없이 큐에 '0'과 '1' 값을 저장하는 쿼리트리 알고리즘에 비하여 중복되는 데이터 영역에서의 질의 라운드 수를 최소화 할 수 있다. 또한 태그의 응답 문자열 중에서 충돌이 발생한 비트의 위치와 충돌이 발생한 비트의 수를 수 있기 때문에 충돌한 비트의 수가 1개인 경우 해당 충돌 비트에 각각 '0'과 '1'의 값을 설정한 후 리더는 두 개의 태그를 동시에 식별한다. 따라서 리더의 식별영역 내에 있는 모든 태그들을 식별하는 시간과 질의 라운드 수를 줄일 수 있는 장점이 있다.

Ⅳ. 성능 평가

본 논문에서는 시뮬레이션을 통하여 쿼리트리 알고리즘과 제안한 알고리즘의 성능을 비교 하였다. 충돌방지 알고리즘의 성능을 평가하기 위하여 태그는 EPCglobal에서 제안한 TDS 1.3v의 표준을 따르는 GID(General Identifier)-96 태그의 데이터 타입을 선정하였다.

4.1 성능 평가 구현 환경

제안한 충돌방지 알고리즘의 성능 평가를 위하여 구현한 시뮬레이션 환경은 표 6과 같다.

제안한 충돌방지 알고리즘의 성능을 평가하기 위한 측정 항목은 표 7과 같다.

4.2 성능 평가

4.2.1 태그 식별시간

태그 식별 시간은 충돌방지 알고리즘의 성능을 비교할 수 있는 궁극적인 척도로 리더와 태그의 질의-응답을 통하여 리더의 영역 내에 모든 태그를 식

표 6. 시뮬레이션 환경

환경	기능 및 성능
RFID Tag	· 96 비트 RFID 태그 · GID-96 태그의 데이터 타입
Host Computer	· Intel Pentium 4 CPU 3.0 GHz / 1GB RAM
Host Computer OS	· RedHat 9.0 워크스테이션
Simulation Toll	· NS-2 2.31

표 7. 성능 평가 측정 항목

구분	측정 항목
충돌방지 알고리즘	· 쿼리트리 알고리즘 · 제안한 충돌방지 알고리즘
태그수	· 100~500 개
태그 식별 시간	· sec
질의-응답 횟수	· time
질의-응답에 사용된 평균 전송 비트수	· bit

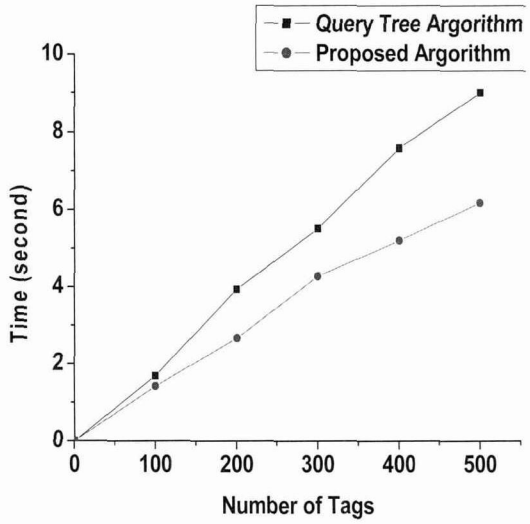


그림 3. 태그 식별시간 비교

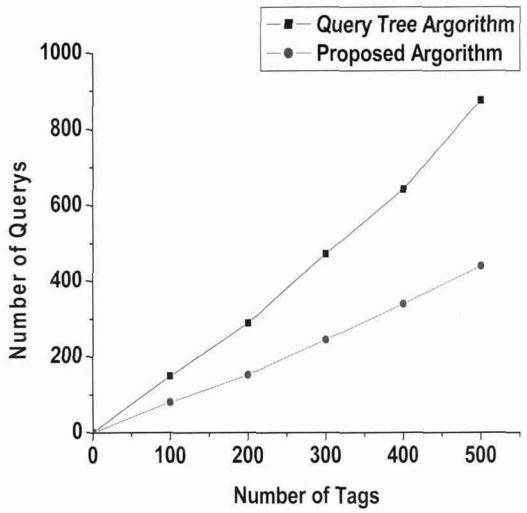


그림 4. 리더의 질의 비트수

별하기 위해 소요되는 시간으로 정의 한다. 알고리즘의 수행 과정 중 중복하여 식별되는 태그에 관해서도 총 태그 식별에 포함 한다. 그림 3은 리더와 태그의 질의-응답을 통하여 리더 영역 내에 존재하는 모든 태그의 식별 시간을 쿼리트리 알고리즘과 제안한 충돌방지 알고리즘으로 비교하였다.

시뮬레이션 결과를 통해 성능 분석을 해보면 태그를 인식하는 데 걸리는 시간은 알고리즘 특성상 태그 개수의 증가에 따라 선형적으로 증가하는 것을 알 수 있다. 쿼리트리 알고리즘의 경우 리더에서 영역 내에 존재하는 100개의 태그를 인식하는데 걸리는 시간은 1.69초 가 소요되지만 제안한 충돌방지 알고리즘의 경우 1.42초가 걸리는 것을 알 수 있다. 리더의 영역 내에 존재하는 태그의 개수가 증가 할수록 모든 태그를 인식하는 데 소요되는 시간의 차이는 점점 벌어져 500개의 태그를 인식하는데 쿼리트리 알고리즘은 9.01초 제안한 충돌방지 알고리즘은 6.18초가 걸린다. 제안한 충돌방지 알고리즘에 비해서 쿼리트리 알고리즘은 불필요하게 발생되는 질의-응답 과정이 늘어나기 때문에 태그의 수가 증가할수록 태그 식별 시간이 증가하게 된다.

4.2.2 질의-응답 횟수

충돌방지 알고리즘은 태그 응답에 따른 충돌을 회피하여 최소한의 시간을 소요하며 태그 ID를 식별하기 위하 것을 목적으로 한다. 리더의 영역내의 태그를 식별하기 위해서 리더와 태그의 질의 응답 횟수를 비교하여 충돌방지 알고리즘에 따른 충돌 회피 능력을 비교 확인 할 수 있다. 그림 4는 리더가 영역 내에 존재하는 태그 개수에 따른 전송한 질의 비트수를 비교한 결과이다.

시뮬레이션 결과를 살펴보면 100개의 태그를 인식할 때 쿼리트리 알고리즘은 총 149번의 단계가 소요되지만 제안한 충돌방지 알고리즘의 경우 총 82번의 단계를 거치는 것을 알 수 있다. 태그의 개수가 늘어날수록 총 태그 인식 단계 수 차이는 점점 멀어져 500개의 태그를 인식하는데 쿼리트리 알고리즘은 875번, 제안한 충돌방지 알고리즘의 경우 445번의 단계를 필요로 하므로 약 1.97배의 성능 향상을 얻을 수 있다. 이것은 태그의 개수가 늘어나면 동일한 Object Class를 갖는 태그가 늘어나므로 일정 범위가 동일한 비트를 갖는 태그가 더 많아지기 때문이다.

그림 3과 4의 시뮬레이션 결과를 살펴보면 100개의 태그를 인식할 경우 제안한 충돌방지 알고리즘이 쿼리트리 알고리즘에 비하여 67단계의 질의가 감소하였으며 이때 시간은 0.27초 감소하였다. 최종적으로 500개의 태그를 인식하였을 경우 제안한 충돌 방지 알고리즘의 쿼리트리 알고리즘보다 430단계의 질의가 감소하였으며 이때 시간은 2.38초가 감소하였다.

V. 결 론

RFID 시스템에서 다중 태그 식별을 효과적으로 처리하기 위해서 RFID 기반의 충돌 방지 알고리즘에 관한 연구가 활발히 진행되고 있다.

본 논문에서는 결정적 알고리즘 중 한 방식인 쿼

리 트리 알고리즘의 성능을 개선한 충돌방지 알고리즘을 제안하였다. 쿼리 트리 알고리즘은 리더에서 태그를 인식하는 경우 한 비트라도 충돌이 발생하면 전체 알고리즘에서 충돌이 발생한 것으로 인식한다. 그러나 제안한 충돌 방지 알고리즘은 리더에서 전송하는 질의 문자열과 태그가 전송하는 응답 비트의 수를 최소화하기 위하여 리더에서 질의 문자열을 전송한 후 태그를 인식하는 과정에서 충돌이 발생하는 경우 충돌이 발생한 비트의 수와 처음 충돌이 발생한 비트의 위치를 파악하여 다음 질의 문자열을 생성함으로서 리더와 태그의 질의·응답 횟수를 줄여 태그 인식 속도를 개선하였다. 또한 태그 인식하는 과정에서 충돌한 비트의 수가 1개이면 해당 충돌 비트에 각각 '0'과 '1'의 값을 설정한 뒤 메모리에 저장함으로서 리더의 질의 횟수를 줄일 수 있다.

성능평가를 통하여 제안한 충돌방지 알고리즘이 쿼리 트리 알고리즘에 비하여 리더의 영역 내에 존재하는 태그를 인식하는데 소요되는 시간이 감소함을 보였으며, 리더에서 전송하는 질의 문자열의 전송횟수가 감소하는 것을 확인하였다.

참 고 문 헌

〔1〕 황태욱, 김영수, 박경환, 900MHz 대역 RFID 시스템의 무선 인터페이스 표준화 동향, 한국전자파학회지, 3-15, 2005년 7월.

〔2〕 박정현, RFID 기술 수준과 도입 사례, ETRI, 전자통신동향분석 제21권 제3호, 137-146, 2006년 6월.

〔3〕 이근호, 한호현, 강병권, 조영빈 역, Klaus Finkenzeller 지음, 유비쿼터스 컴퓨팅의 핵심 RFID HANDBOOK, 영진닷컴, 161-180, 2005년 4월.

〔4〕 이현지, 김종덕, 충돌 비트 위치를 활용한 RFID 다중태그 인식 알고리즘, 한국통신학회 논문지 Vol. 31, No. 4A, pp. 431-439, Apr. 2006.

〔5〕 임인택, 다중 태그 식별을 위한 개선된 질의 트리 충돌방지 알고리즘, Journal of Korea Multimedia society Vol. 9, No. 3, pp. 307-314, March 2006.

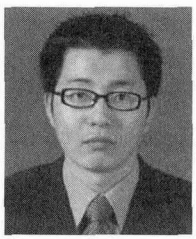
〔6〕 Auto-ID Center, Dratf protocol specification for a 900 MHz Class 0 Radio Frequency Identification Tag, EPCglobal, 23 Feb 2003.

〔7〕 Auto-ID Center, EPC^{TM} Tag Data Standards Version 1.3 Auto-ID Center, 9 Sep, 2005.

이 준 혁 (Jun-Hyuk Lee) 정회원

한국통신학회 논문지 제29권 12A호 참조

정 원 수 (Won-soo Jung) 정회원



2003년 2월 대진대학교 통신공학과 졸업 (공학사)

2005년 2월 광운대학교 대학원 전자통신공학과 졸업 (공학석사)

2007년 8월 광운대학교 대학원 전자통신공학과 (박사수료)

<주관심 분야> 센서 네트워크, RFID 시스템, 임베디드 시스템

오 영 환 (Young-hwan Oh) 정회원

한국통신학회 논문지 제31권 제1B호 참조