RFID 시스템에서 태그 충돌 중재를 위한 하이브리드 기법

(A Hybrid Approach to Arbitrate Tag Collisions in RFID systems)

류 지 호 † 이 호 진 † 석 용 호 †† 권 태 경 ††† 최 양 희 ††††
(Jiho Ryu) (Hojin Lee) (Yongho Seok) (Taekyoung Kwon) (Yanghee Choi)

요 약 본 논문에서는 RFID 시스템에서 쿼리 트리 기반의 태그 충돌 중재를 위한 새로운 프로토콜을 제안한다. 제안한 하이브리드 쿼리 트리(Hybrid Query Tree) 기법은 이진 쿼리 트리 대신에 4-ary 쿼리 트리를 이용하여 태그 충돌 수를 줄였으며 추가적으로 발생하는 유휴 시간을 줄이기 위해 slotted 백오프 기법을 활용하였다. 실험결과 및 수학적 분석은 제안한 하이브리드 쿼리 트리 프로토콜이 기존에 제시된 기법보다 우수한 성능을 보임을 입증한다.

키워드 : RFID 시스템, 태그 혼잡 방지, 패시브 태그, 쿼리 트리

Abstract In this paper, we propose a new hybrid approach based on query tree protocol to arbitrate tag collisions in RFID systems. The hybrid query tree protocol that combines a tree based query protocol with a slotted backoff mechanism. The proposed protocol decreases the average identification delay by reducing collisions and idle time. To reduce collisions, we use a 4-ary query tree instead of a binary query tree. To reduce idle time, we introduce a slotted backoff mechanism to reduce the number of unnecessary Query commands. Simulation and numerical analysis reveal that the proposed protocol achieves lower identification delay than existing tag collision arbitration protocols.

Key words : RFID systems, tag anti collision, passive tag, query tree

1. 서 론

현재 상품에 대한 정보를 쉽게 인식하기 위해서 바-코드가 많이 사용되고 있다. 검은색 선의 두께 및 간격으로 정보를 표현하는 바-코드는 리더기가 인식할 수 있는 범위가 매우 좁으며 표현 가능한 정보 역시 한정되어있다는 단점을 가지고 있다. 이에 따라 최근에 RFID(Radio Frequency Identification) 시스템이 상품에 부착된 RFID 태그를 인식하여 그 정보를 이용하는 것이 제안되었다. RFID 태그는 바-코드 보다 더 많은 정보를 담을 수 있지만 그에 따른 비용도 늘어나게 되어 개개의 상품별로 부착되는 RFID 태그의 가격은 RFID 시스템에서 매우 중요한 요소가 된다. 또한 RFID 리더는 기존의 바-코드 시스템보다 더 빠른 인식률을 요구한다. 실제 RFID 태그는 파워의 공급 유형에 따라서 패시브, 세미-패시브, 액티브 태그로 구분된다. 이중 패시브 태그의 경우 RFID 리더가 발생시키는 전자기장으로부터 파워를 유도하여 구동되기 때문에 가격 면에서 RFID 시스템에 가장 적합한 형태라고 볼 수 있다. 따라서 본 논문에서는 패시브 태그에 초점을 두고 연구를 진행한다.

RFID 시스템에서 리더기의 가장 중요한 역할은 자신의 인식 범위 내의 태그들을 식별하는 것이다. 그 다음에 식별해낸 태그들에게 읽기(Read) 혹은 쓰기(Write)

† 학생회원 : 서울대학교 전기컴퓨터공학부
jhryu@mmlab.snu.ac.kr
lumiere@mmlab.snu.ac.kr
†† 정 회 원 : LG 전자 기술원 정보기술연구소 PAN 그룹 선임연구원
yhseok@mmlab.snu.ac.kr
††† 정 회 원 : 서울대학교 전기컴퓨터공학부 교수
tk@mmlab.snu.ac.kr
†††† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
yhchoi@mmlab.snu.ac.kr
논문접수 : 2007년 1월 22일
심사완료 : 2007년 9월 19일

정보과학회논문지 : 정보통신 제34권 제6호(2007.12)

명령을 수행하도록 지시할 수 있다. 이때 태그는 반송파 감지(carrier sensing) 능력이 없기 때문에 태그들의 응답은 충돌을 일으킬 수 있다. 이에 우리는 RFID 리더의 쿼리에 응답하는 태그의 수에 따라 3 가지의 사이클을 정의 한다.

충돌 사이클: 2개 이상의 태그가 동시에 응답한 경우. RFID 리더는 태그의 아이디를 식별할 수 없다.

유휴 사이클: 태그의 응답이 없는 경우. 쓸데없이 낭비되는 시간이 된다.

성공 사이클: 정확히 하나의 태그만이 응답만 경우. RFID 리더는 태그의 아이디를 식별 할 수 있다.

RFID 시스템의 성능은 태그들을 식별해 내는데 걸리는 전체 시간이 짧을수록 좋다고 할 수 있기에, 태그 응답의 충돌을 줄이면서도 RFID 시스템의 성능을 향상시키려는 많은 논문이 제안되었다[1-7]. 이러한 충돌 방지 프로토콜은 크게 알로하(ALOHA)와 트리(Tree) 기반의 프로토콜로 나누어 진다. 알로하[8,9] 기반 프로토콜은 태그의 충돌을 크게 줄 일 수 있으나 특정 태그의 경우 일정시간 이상 식별되지 않을 수도 있는, 태그 기아 문제(tag starvation problem)를 갖고 있다. 반면에 트리 기반 프로토콜은 태그 기아 문제는 없지만 상대적으로 태그의 식별 시간이 더 오래 걸린다는 약점이 있다. 따라서 본 논문에서는 트리 기반 프로토콜에서 태그 식별 시간을 줄일 수 있는 태그 충돌 방지 프로토콜을 제안한다.

트리 기반의 태그 충돌 방지 프로토콜은 이진 탐색 알고리즘과 흡사하다. 트리 기반의 프로토콜에서 태그 응답이 충돌을 일으킨 경우 태그들을 두 부분집합으로 나누어지고 RFID 리더는 각각의 부분집합에 대해 다시 쿼리를 보내서 식별하게 된다. 대표적인 트리 기반의 태그 충돌 방지 프로토콜로는 이진 트리 분할(Binary Tree Splitting) 기법과 쿼리 트리(Query Tree) 기법이 있다. 이진 트리 분할 기법에서는 각 태그들이 임의의 이진 숫자(0 또는 1)를 생성해 내기 위한 함수가 필요하고 전송 순서를 결정하기 위한 카운터 값도 필요하다. 이를 통해 이진 트리 분할 기법은 쿼리 트리 기법보다 보내어지는 쿼리 메시지 수를 줄일 수 있지만 실제 구현에 있어서는 그 단가가 더 높고 복잡하다는 단점을 갖는다. 따라서 우리는 쿼리 트리 기법을 확장하여 유휴 사이클과 충돌 사이클을 줄이는 기법을 제안한다.

쿼리 트리 기법은 태그 응답 충돌 방지 프로토콜 중에서 가장 대표적인 기법이라 할 수 있다. 쿼리 트리 기법에서 각각의 태그들을 정 이진 트리(full binary tree)의 말단 노드(leaf node)에 해당되고, 각 트리의 노드들은 하나의 쿼리 명령에 상응한다. RFID 리더는 0 또는 1 의 초기 쿼리 프리픽스와 함께 쿼리 메시지를 태그들에게 보낸다. 만약 RFID 리더가 p라는 쿼리 프리픽스로 쿼리를 보냈는데, 이에 대한 태그의 응답이 충돌을 일으키게 되면 RFID 리더는 다음 번 쿼리 메시지에는 원래 쿼리 프리픽스 p보다 1-bit가 늘어난 p0 쿼리 프리픽스를 보내게 된다. p0로 시작하는 ID를 갖는 태그들을 모두 식별한 다음에 RFID 리더는 p1의 쿼리 프리픽스를 보내서 계속 태그들을 식별한다. 따라서 정 이진 트리의 중간 노드들은 충돌 사이클에 해당하게 되고, 말단 노드들은 유휴 사이클이나 성공 사이클에 해당한다. 쿼리 트리 프로토콜은 이진 트리 분할 프로토콜과 달리 태그에 별도의 저장소를 필요로 하지 않고 동작이 간단하다는 장점이 있다.

본 논문에서 저자들은 쿼리트리를 확장한 하이브리드 태그 충돌 방지 프로토콜을 제안한다. 제안하는 하이브리드 쿼리 트리(Hybrid Query Tree, HQT) 프로토콜은 크게 두 가지 기법으로 구성된다. 첫째로 4-ary 탐색 트리 기법인데, 이는 쿼리 트리 프로토콜에서 충돌이 발생했을 때 쿼리 프리픽스를 1-bit 늘이는 것이 아니라 2-bits를 늘이는 것을 의미한다. 이렇게 되면 트리 구조의 특성상 충돌 사이클을 줄일 수 있지만[10] 유휴 사이클은 늘어나게 된다. 사실 쿼리 프리픽스를 2-bits가 아니라 3-bits 혹은 4-bits로 늘이게 되면 더 많은 충돌 사이클을 줄일 수 있다. 하지만[11] 논문에서 저자들이 보였듯이 q-ary 트리 알고리즘에서 최적의 성능을 보이는 경우는 3-ary 트리이며 q가 3 이상인 경우에는 성능이 감소한다. 하지만 쿼리 트리 프로토콜에서 3-ary 트리의 구현이 불가능하기 때문에 본 논문에서는 4-ary 트리를 선택하였다.

둘째로 태그는 RFID 리더에게 응답을 할 때 slotted 백오프(back-off) 기법을 적용한다. 4-ary 탐색 트리 알고리즘이 사용되면 충돌 사이클은 많이 줄어들게 되지만, 그 여파로 유휴 사이클은 많이 늘어나게 된다. 따라서 유휴사이클을 줄이기 위한 방법으로 slotted 백오프 기법을 제안한다. 태그가 RFID 리더의 쿼리 메시지에 응답할 때, 각 태그들은 자신의 ID부분 중 일부를 사용하여 백오프 시간을 설정하게 된다. 태그 응답이 충돌을 일으키게 되면 RFID 리더는 태그의 백오프 시간을 이용하여 특정 태그 ID의 존재 유무를 일부 알 수 있게 되고 필요 없는 쿼리(유휴 사이클)를 줄일 수 있게 된다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 저자들이 제안하는 하이브리드 쿼리 트리 프로토콜에 대해서 설명한다. 3장에서는 수학적 분석을 통하여 하이브리드 쿼리 트리의 성능 향상 정도를 알아본다. 그리고 4장에서는 모의 실험을 통한 제안된 기법의 성능 평가를 수행하고, 5장에서는 제안한 기법의 실제 구현과 관련된 이슈에 대한 고찰 및 기존의 다른 알고리즘과의 비교를 통해 정리하고 마지막으로 6장에서 결론을 맺는다.

2. 하이브리드 쿼리 트리 프로토콜

이 장에서는 하이브리드 쿼리 트리 프로토콜을 소개한다. 하이브리드 쿼리트리에서는 4-ary 탐색 트리 알고리즘을 사용하여 충동 사이클을 크게 줄이지만 4-ary 탐색 트리 알고리즘의 역효과로 유휴 사이클이 많이 늘어나게 된다. 이를 보완하기 위해 slotted 백오프 기법을 제안하여 유휴 사이클 역시 크게 줄였다.

2.1 4-ary 탐색 트리 알고리즘

쿼리 트리 프로토콜은 그림 1의 (a)에서 보여주듯이 쿼리 메시지를 보낸 후 충돌이 발생하면, 쿼리 프리픽스 끝에 0과 1을 이어 붙여서 새로운 쿼리 프리픽스를 만들어 사용한다. 따라서 이 기법은 위에서 설명한 것과 같이 이진 탐색 트리 방식이라고 할 수 있다. 그러나 이 경우 전체 트리의 깊이(depth)가 증가해서 태그를 식별하는데 많은 쿼리 메시지가 필요할 수 있다는 단점이 있다. 따라서 본 논문에서는 4-ary 탐색 트리 알고리즘을 이용한다. 4-ary 탐색 트리 알고리즘은 충돌이 발생한 경우 RFID 리더가 쿼리 프리픽스를 1-bit 늘이는 것이 아니라 2-bits를 늘인다. 즉 충돌이 발생하면 쿼리 프리픽스 끝에 00, 01, 10, 그리고 11을 이어 붙여서 새로운 쿼리 프리픽스를 2개가 아닌 4개를 만들어서 사용한다. 그러나 이 경우 충돌 사이클이 줄어드는 대신에 유휴 사이클이 늘어나게 된다. 그림 1의 (b)는 4-ary 탐색 트리 알고리즘의 동작을 보여 준다. 그림에서 보여주듯이 중간 노드들은 충돌 사이클이 된다. 만약 n개의 태그가 있다면 중간 노드의 수, 즉 충돌 사이클의 수는 아래와 같이 계산 된다[10].

- **이진 탐색 트리 알고리즘: $n + k_2 - 1$** (1)
- **4-ary 탐색 트리 알고리즘: $(n + k_4 - 1)/3$** (2)

여기에서 k_m은 full m-ary 트리에서 유휴 사이클의 수를 의미한다. 따라서 4-ary 탐색 트리 알고리즘을 사용하면 많게는 이진 탐색 트리 알고리즘의 1/3 수준으로 충돌 사이클을 줄일 수 있게 된다. 하지만 이는 전적으로 태그 ID의 분포에 달려있다.

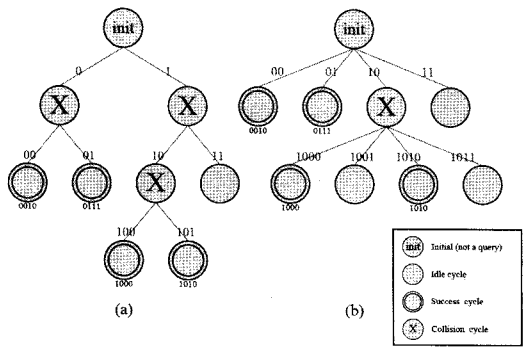

그림 1 (a) 이진 탐색 트리 알고리즘과 (b) 4-ary 탐색 트리 알고리즘. 4-ary 탐색 트리를 이용하게 되면 충돌 사이클은 줄어들지만 유휴 사이클이 많이 늘어나게 된다.

2.2 Slotted 백오프 태그 응답 알고리즘

쿼리 트리 프로토콜에서 태그들은 RFID 리더의 쿼리 메시지를 받으면 안에 담겨 있는 쿼리 프리픽스와 자신의 ID를 비교 한 후 쿼리 프리픽스가 자신의 ID 프리픽스와 매칭이 되면 그 즉시 응답을 하게 된다. 그러나 본 논문에서 제안하는 방식에서 태그는 즉시 응답을 하지 않고 짧은 slot 단위시간 동안 지연하였다가 응답하게 된다. 태그는 자신의 ID의 프리픽스 중에서 쿼리 메시지의 쿼리 프리픽스와 일치하는 바로 다음의 두 bits를 RFID 리더에게 응답할 때 이용한다. 만약 그 두 bits가 00인 경우에 태그는 RFID 리더의 쿼리 메시지를 받은 후 백오프 없이(지연 없이) 바로 응답을 하게 된다. 만약 두 bits가 01인 경우에는 1 slot 단위시간이 지난 후에 응답을 하게 된다. 즉, 쿼리 프리픽스와 일치하는 태그 ID의 프리픽스 바로 다음 두 bits의 값으로 백오프를 한 후 응답을 하는 것이다. 만약 010100, 010101, 010110의 ID를 가진 3개의 태그가 존재한다고 가정하자. 이때 RFID 리더가 0101의 쿼리 프리픽스를 이용하여 쿼리 메시지를 보냈다면 각 태그들은 자신들의 응답을 각각 0 slot 단위시간, 1 slot 단위시간, 2 slot 단위시간 기다린 후에 보내게 된다. 그림 2가 위에서 예로 든 slotted 백오프 태그 응답 알고리즘의 동작 과정을 보여준다.

물론 slot 단위시간 보다 태그의 응답 시간이 더 길기 때문에 RFID 리더는 쿼리를 보낸 후에 반송파 감지(carrier sensing)를 통하여 태그의 응답이 충돌이 발생했는지 여부를 파악할 수 있다.

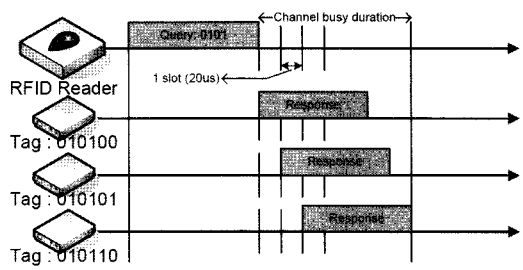

그림 2 Slotted 백오프 태그 응답 알고리즘. 태그들은 0부터 3사이의 slot중 하나를 선택하여 백오프 한 뒤에 응답한다.

2.3 쿼리 축소 알고리즘

위와 같이 태그가 slotted 백오프 방식으로 응답을 하게하여 얻을 수 있는 효과는 무엇일까? 그것은 바로 불

필요한 유휴 사이클을 줄이고자 하는 것이다. 만약 태그의 응답이 충돌을 일으키지 않았다면 slot 단위시간 정도의 오버헤드가 덧붙여질 것이다. 그렇지만 태그의 응답이 충돌을 일으킨 경우에는 RFID 리더는 쿼리 축소 알고리즘을 통하여 유휴 사이클을 생략시킬 수가 있다. 충돌이 발생한 경우, 태그는 쿼리 프리픽스와 일치하는 자신의 ID 프리픽스의 ***다음 두 bits***를 사용하여 백오프를 한 후 응답하였기 때문에 RFID 리더는 충돌이 발생했을 경우 응답을 한 태그들의 ID 프리픽스 ***다음 두 bits***의 범위를 알 수가 있다. 다시 위의 그림 2의 예를 보면 3개의 태그들은 각각 0 slot 단위시간, 1 slot 단위시간, 2 slot 단위시간 후에 자신의 ID로 응답한다. RFID 리더 입장에서는 자신의 쿼리 메시지를 전송한 후에 바로 채널이 busy하기 시작하여 *0101110*의 *ID*를 가진 태그가 전송을 끝마칠 때까지 채널이 busy함을 알게 되고 이때 충돌이 발생했음을 알 수 있다. 그런데 태그 ID의 길이를 알고 전송 속도 역시 정해져 있기 때문에 가장 처음에 응답을 한 태그와 가장 나중에 응답을 한 태그가 백오프 할 때 사용한 bits를 알게 되는 것이다. 위의 그림 2의 예의 경우 우선 쿼리 메시지 전송 직후 바로 태그의 응답이 시작되었으므로 RFID 리더는 응답한 태그들 중에 *00-bits*를 가진 *tag*가 존재함을 알게 되고, 그 뒤에 채널이 **2 slot 시간단위 + (태그의 응답의 길이)/전송속도** 만큼 busy한 것을 알게 된다. 여기서 전파(propagation) 시간은 매우 짧은 시간이므로 무시하면, 결국 RFID 리더는 가장 마지막에 전송을 시도한 태그들이 10-bits를 사용하여 백오프 했음을 알아낼 수 있는 것이다. 따라서 RFID 리더는 0101 쿼리 프리픽스 다음으로는 010100, 010101, 010110 만을 새롭게 쿼리할 쿼리 프리픽스로 선택하고 다음에 쿼리할 목록에 포함하게 된다. 따라서 010111으로는 쿼리할 필요가 없음을 알게 될 뿐만 아니라(불필요한 유휴 사이클을 제거하였다.) 한 bit씩 늘여가며 쿼리 메시지를 보낼 때 보다 적은 수의 쿼리 메시지를 보내게 됨은 자명한 다. 실제로 한 bit씩 늘여가면서 위의 3개의 태그를 쿼리하는 데에는 0101 쿼리 프리픽스 이후 4번의 쿼리 메시지를 더 보내야 하지만 저자들이 제안하는 방법을 사용하면 3번만으로 가능하다.

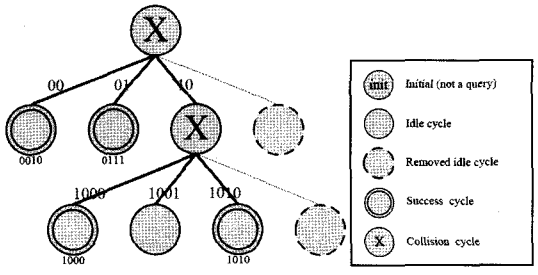

그림 3 하이브리드 쿼리 트리 프로토콜의 동작. 충돌 사이클 및 유휴 사이클을 동시에 감소할 수 있다.

표 1 쿼리 트리 프로토콜의 동작

단계	쿼리 프리픽스	쿼리 결과	쿼리 큐
1	0	충돌	1,00,01
2	1	충돌	00,01,10,11
3	00	성공	01,10,11
4	01	성공	10,11
5	10	충돌	11,100,101
6	11	유휴	100,101
7	100	성공	101
8	101	성공	Empty

표 2 하이브리드 쿼리 트리 프로토콜의 동작

단계	쿼리 프리픽스	쿼리 결과	쿼리 큐
1	empty string	충돌	00,01,10
2	00	성공	01,10
3	01	성공	10
4	10	충돌	1000,1001,1010
5	1000	성공	1001,1010
6	1001	유휴	1010
7	1010	성공	Empty

표 3 하이브리드 쿼리 트리 프로토콜에서 태그의 동작

P가 태그의 ID라고 하면 다음과 같이 표현이 가능하다
P = $p_1p_2p_3p_4\cdots p_k$ (ID의 길이는 k이고 각 pi는 0 또는 1 ($1 \le i \le k$)

RFID리더로부터 쿼리 q를 받았다면,

IF q = ε (empty) or q = $p_1p_2p_3\cdots p_{|q|}$ THEN
  리더에게 자신의 ID P로 응답한다.
  단, 이때 slotted 백오프 응답 기법을 사용한다
  ($|q| \le k$, 백오프 시에는 $p_{|q+1|}p_{|q+2|}$의 두 bits를 이용한다.
  만약 $|q| = k$ 라면, 백오프 없이 바로 P로 응답한다.)
ELSE
  아무 것도 하지 않는다. (응답하지 않는다.)
END IF

표 1과 표 2는 그림 1에서 제시한 예제에서 쿼리 트리 프로토콜과 제안한 하이브리드 쿼리 트리 프로토콜의 동작 과정을 보여 준다. 또한 그림 3을 보면 제안한 하이브리드 쿼리 트리 프로토콜의 경우 그림 1의 (b)에서와는 달리 유휴 사이클이 제거되었음을 알 수 있다. 따라서 제안한 프로토콜은 더 적은 수의 충돌을 발생시킬 뿐만 아니라 유휴 사이클도 줄었기 때문에 전체 쿼리 메시지 수가 줄었음을 보여 준다. 하이브리드 쿼리 트리 프로토콜에서 태그와 RFID 리더의 동작은 아래 표 3과 표 4에 정리하였다.

표 4 쿼리 트리 프로토콜에서 RFID 리더의 동작

```
쿼리 q1: q1 쿼리 프리픽스를 갖고 있는 RFID 리더의 쿼리 메시지
Q: 쿼리 큐. 큐에 담긴 프리픽스 순서대로 쿼리 메시지가 전해진다.
  Q = (q1, q2, q3, ···), 가장 처음에 Q = ε, 여기서 ε는 empty 스트링

WHILE Q != empty (여기에서 empty는 Q가 비었음을 의미한다.)
    쿼리 q1 (Q에서 가장 앞에 있는 쿼리 프리픽스)을 태그들에게 보낸다.
      Q에서 q1를 지운다. Q = (q2, q3, q4, ···)
      태그의 응답을 기다린다.
    IF 성공 사이클 THEN
      태그 ID를 성공적으로 인식하였다. 나중에 사용할 수 있도록 ID를 저장.
    ELSE IF 유휴 사이클 THEN
      아무것도 하지 않는다.
    ELSE (충돌 사이클)
      쿼리 축소 알고리즘을 통하여 줄일 수 있는 유휴 사이클을 파악한다.
      Q에 q100 or q101 or q110 or q111를 추가 한다.
    END IF
END WHILE
```

3. 수학적 분석

본 장에서는 쿼리 트리 프로토콜(2-ary 쿼리 트리 프로토콜), 4-ary 쿼리 트리 프로토콜 그리고 하이브리드 쿼리 트리 프로토콜을 사용하여 태그들을 인식할 때 걸리는 총 지연시간을 분석한다. 4-ary 쿼리 트리 프로토콜은 기존의 이진 트리 검색 알고리즘이 적용된 쿼리 트리 프로토콜 대신 4-ary 트리 검색 알고리즘을 적용한 쿼리 트리 프로토콜을 의미한다. 분석의 편의를 위해 worst case인 상황을 고려하였으며, 위의 프로토콜들을 사용할 때 채널 상황에 따른 태그 인식 실패는 없는 것으로 가정한다. 즉 태그 응답의 충돌에 의해서만 태그 인식 성공 실패 여부가 결정 된다.

3.1 기본 정리

우선 하이브리드 쿼리 트리 프로토콜의 지연시간을 분석하기 전에 필요한 것들을 정리하려고 한다. 먼저 쿼리 트리 프로토콜의 지연 시간을 알아보자. 만약 RFID 네트워크에 현재 N개의 태그가 있다고 가정할 때 쿼리 트리 프로토콜을 사용하여 전체 태그를 인식할 때 걸리는 총 지연시간을 D_{QT}라 하면, D_{QT}는 다음과 같이 표현할 수 있다.

$$D_{QT} = N_C \cdot D_C + N_I \cdot D_I + N_S \cdot D_S \quad (3)$$

위 식에서 N_C, N_I, N_S는 각각 N개의 태그들을 인식하는 중에 발생하는 충돌 사이클의 수, 유휴 사이클의 수 그리고 성공 사이클의 수를 의미한다. 그리고 D_C, D_I, D_S는 각각 이들 충돌 사이클, 유휴 사이클, 성공 사이클의 평균 지연시간을 의미한다. 그런데 만약 충돌 사이클, 유휴 사이클 그리고 성공 사이클의 평균 지연시간이 거의 동일하다고 가정하면 위의 식 (1)은 다음과 같이 쓸 수 있다.

$$D_{QT} = (N_C + N_I + N_S) \cdot D \quad (4)$$

위 식에서 D는 한 사이클 (충돌, 성공, 유휴)의 평균 지연시간이 된다.

Full k-ary 트리에서 중간 노드의 수를 I라고 할 때 full k-ary 트리의 전체 노드 수 N은

$$N = k \cdot I + 1 \quad (5)$$

같이 쓸 수 있다. 그런데 쿼리 트리 프로토콜은 이미 앞에서 언급하였듯이 정 이진 트리(full binary tree)를 만들게 되므로 위 식 (5)에 의해 정 이진 트리에서 전체 노드(쿼리 트리 프로토콜에서는 사이클에 해당)의 수 N_{QT}는 아래와 같이 쓸 수 있다.

$$N_{QT} = N_C + N_I + N_S \quad (6)$$

$$N_{QT} = 2 \cdot N_C + 1 \quad (7)$$

위에서 설명하였듯이 정 이진 트리에서의 중간 노드는 쿼리 트리 프로토콜에서의 충돌 사이클에 해당하기 때문이다. 따라서 식 (7)에 의해 D_{QT}는 아래와 같이 새로 쓸 수 있다.

$$D_{QT} = (2 \cdot N_C + 1) \cdot D \quad (8)$$

따라서 이제 우리는 충돌 사이클의 수 즉, N_C를 구하면 쿼리 트리 프로토콜의 전체 태그를 인식할 때 걸리는 총 지연시간을 구할 수 있다.

3.2 쿼리 트리 프로토콜 및 4-ary 쿼리 트리 프로토콜

우리는 논문 [3]에서 사용된 방법으로 worst case의 쿼리 트리 프로토콜의 태그 인식 지연 시간을 계산한다. Worst case인 경우 논문 [3]에 의하면 태그 ID 길이가 k이고 n 개의 태그가 존재하는 경우 N_{QT}는 아래와 같이 표현할 수 있다.

$$N_{QT} = n \cdot (k + 2 - \log_2 n) - 3 \quad (9)$$

따라서 D_{QT}는 식 (7), (8) 그리고 (9)를 이용해 구할 수 있다.

식 (5)에 의해 4-ary 쿼리 트리 프로토콜에서도 같은 방식으로, 정사 진 트리에서(full 4-ary tree) 전체 노드의 수 N_{4QT}를 아래와 같이 쓸 수 있다.

$$N_{4QT} = 4 \cdot N_C + 1 \quad (10)$$

또한, (10) 식에 의해 4-ary 쿼리 트리 프로토콜을 사용하여 전체 태그를 인식할 때 걸리는 총 지연시간

D_{4QT}는 아래와 같이 새로 쓸 수 있다.

$$D4_{QT} = (4 \cdot N_C + 1) \cdot D \quad (11)$$

N_{4QT}를 4-ary 쿼리 트리 프로토콜을 사용할 경우의 모든 사이클의 수라고 한다면 N_{4QT}는 아래와 같이 계산할 수 있다.

$$N_{4QT} = n \cdot \left(k + \frac{1}{3} - \log_4 n\right) - \frac{13}{3} \quad (12)$$

증명. 4-ary 쿼리 트리 프로토콜을 사용할 경우, 태그들은 각각 유일한 k bit의 ID를 가지고 있으므로 worst case를 가정하면 서로 다른 두 태그가 (k-2)개의 bit까지 동일 한 경우 (k-2)/2 번의 충돌을 일으킬 것이다. 따라서 태그의 총 수가 n개이고 4-ary 쿼리 트리 프로토콜을 사용할 경우 트리에서 높이(depth)가 l인 위치일 때의 최대 충돌 사이클의 수 $N_{4C,l}$는 다음과 같이 표현할 수 있다[3].

$$N_{4C,l} = \begin{cases} 4^l & (0 < l < \log_4 n - 1) \\ n/2 & (\log_4 n - 1 < l < k - 1) \end{cases} \quad (13)$$

식 (13)과 식 (10)을 이용해 계산해 보면 아래와 같다.

$$\begin{aligned} N_{4C} &\le \sum_{l=1}^{b-2} N_{4C,l} \\ &= \sum_{l=1}^{\lfloor \log_4 n - 1 \rfloor} 4^l + \frac{n}{2}(k - 2 - \lfloor \log_4 n \rfloor + 1)/2 \\ &\le \frac{n}{4}\left(k + \frac{1}{3} - \log_4 n\right) - \frac{4}{3} \end{aligned} \quad (14)$$

따라서 식 (12)가 성립한다. □

(9)번 식과 (12)번 식을 보면 쿼리 트리 프로토콜을 사용하거나 4-ary 쿼리 트리 프로토콜을 사용할 경우 전체 태그 인식에 걸리는 지연시간이 비슷함을 알 수 있다. 이는 4-ary 쿼리 트리 프로토콜의 경우 충돌 사이클의 수를 많이 줄였으나 유휴 사이클이 수도 그 만큼 늘어 났기 때문이다.

3.3 하이브리드 쿼리 트리 프로토콜

하이브리드 쿼리 트리 프로토콜의 경우도 식 (14)번에서와 같이 충돌 사이클의 수를 계산할 수 있을 것이다. 즉 worst case인 경우 충돌 사이클은 두 개의 태그 응답에 의해서 발생하게 된다. 그런데 이때 쿼리 축소 알고리즘이 적용 된다면 하이브리드 쿼리 트리 프로토콜을 사용할 경우의 모든 사이클 수 N_{HQT}는 아래와 같이 쓸 수 있다.

$$N_{HQT} = \frac{9}{16} n \cdot \left(k + \frac{1}{3} - \log_4 n\right) - 2 \quad (15)$$

증명. 하이브리드 쿼리 트리 프로토콜을 사용할 경우에도 충돌 사이클의 수는 N_{4C}와 같다. 그런데 worst case의 경우 충돌 사이클은 두 개의 태그 응답에 의해서 발생하므로 이 경우를 살펴보면 그림 4의 6가지 경우가 된다. 즉, 축소 가능한 쿼리 메시지 수를 구해 보면

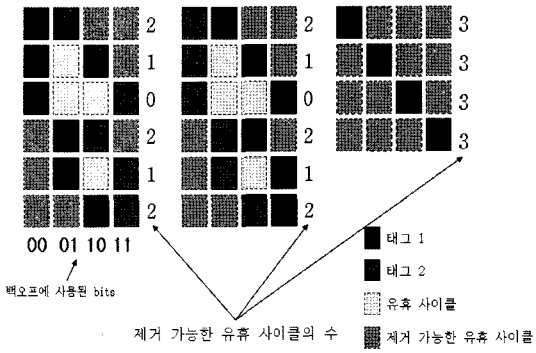

그림 4 하이브리드 쿼리 트리 프로토콜 적용시 worst case에서의 태그 ID 분포에 따른 제거 가능한 유휴 사이클

$$\frac{1}{16}(2+1+0+2+1+2+2+1+0+2+ 1+2+3+3+3+3) = \frac{28}{16} = \frac{7}{4} \quad (16)$$

즉, 평균 4/3개의 쿼리 메시지를 줄일 수 있다. 본래 4-ary 쿼리 트리 프로토콜에서 충돌이 발생한 경우 그 다음 번에는 4개의 새로운 쿼리 메시지가 생성되었으나, 하이브리드 쿼리 트리 프로토콜에서는 쿼리 축소 알고리즘에 의해 4-7/4 즉 9/4개의 쿼리 메시지만이 새로 생성 되는 것이다. 따라서 식 (12)는 하이브리드 쿼리 트리 프로토콜의 경우 아래와 같이 적용 된다.

$$N_{HQT} = \frac{9}{4} \cdot N_{4C} + 1 \quad (17)$$

따라서 식 (14)와 (17)에 의해 식 (15)가 성립한다. □

식 (15)에 의하면 하이브리드 쿼리 트리 프로토콜을 사용하게 될 경우 worst case에서는 쿼리 트리 프로토콜의 약 2/3만큼의 전체 태그 인식 지연시간이 걸리게 됨을 알 수 있다. 아래 그래프는 위의 식 (9), (12) 그리고 (15)를 n, 즉 전체 태그 수를 변화 시켜 가면서 그린 것이다. Tag ID의 길이는 128bit로 설정하였으며, 그래프에서 x축은 tag의 수가 되고 y축은 전체 태그를 인식하는데 필요한 총 사이클의 수를 나타낸다. 그래프에서 볼 수 있듯이 본래의 쿼리 트리 프로토콜과 4-ary 쿼리 트리 프로토콜의 경우 그 값이 별로 차이가 나지 않음을 알 수 있으며 제안한 하이브리드 쿼리 트리 프로토콜의 경우 전체적으로 33%정도 감소하였다.

4. 성능 평가

이 장에서는 본 논문에서 제안한 알고리즘의 성능 평

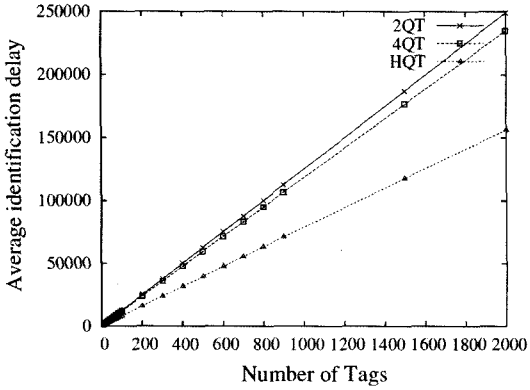

그림 5 Worst case에서 태그 수에 따른 모든 태그 인식에 필요한 총 사이클의 수

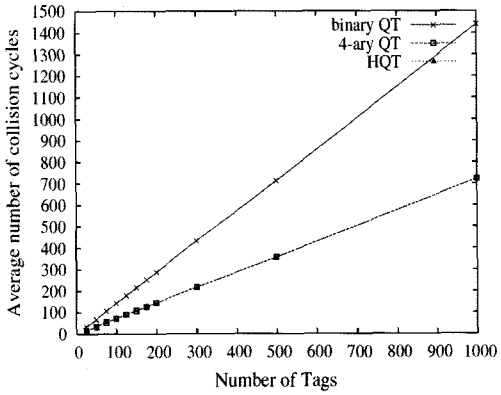

그림 6 전체 태그 수의 증가에 따른 충돌 사이클의 변화

가를 위하여 수행한 모의 실험 결과를 설명한다. 저자들은 NS2[12] 시뮬레이터에 제안한 하이브리드 쿼리 트리 프로토콜 및 비교 대상으로 쿼리 트리 프로토콜과 4-ary 쿼리 트리 프로토콜을 구현하였다. 실험에 사용된 RFID 시스템 네트워크는 하나의 RFID 리더와 n개의 태그들로 구성된다. 각각의 태그들은 128bits의 각기 다른 ID를 가지고 있으며 모두 다 RFID 리더의 인식 범위 내에 있다고 가정하였다. 또한 수학적 분석에서와 같이 채널 상태에 따른 태그 인식 실패는 없다고 가정하였다. 쿼리 메시지의 크기와 태그의 응답의 길이는 128bits로 하였고, 전송 속도는 128Kbps로[13] 설정하였다. 백오프 slot 단위시간의 길이는 20μs로 설정하였고 유휴 사이클의 길이는 하이브리드 쿼리 트리의 경우 '80μs(20μs + 최대 백오프 slot(3)) + 쿼리 메시지 전송시간'으로, 다른 프로토콜들의 경우 '20μs + 쿼리 메시지 전송시간'으로 설정하였다. 저자들은 각각의 프로토콜에 대해 충돌 사이클의 수, 유휴 사이클의 수, 전체 태그를 인식하는데 걸리는 총 지연시간의 3가지 척도에 대한 비교를 하였다. 각 시뮬레이션은 모두 20회를 실시한 결과를 평균 내었다.

4.1 전체 태그 수에 따른 충돌 사이클의 수

그림 6은 각 프로토콜 별로 충돌 사이클의 수를 보여주고 있다. 4-ary 쿼리 트리 프로토콜과 하이브리드 쿼리 트리 프로토콜은 이진 탐색 트리 알고리즘 대신에 4-ary 탐색 트리 알고리즘을 사용하기 때문에 이진 탐색 알고리즘인 쿼리 트리 프로토콜보다 충돌 사이클이 많이 줄어들었음을 알 수 있다. 실험에 사용한 태그 ID는 임의로 생성하였기 때문에 균등하게 분포하고 있다고 가정할 수 있고, 그림 6을 보면 태그 수가 증가함에 따라 충돌 사이클의 수도 선형으로 증가함을 볼 수 있다. 따라서 일반적으로 4-ary 탐색 트리 알고리즘을 사용하면 이진 탐색 트리 알고리즘의 절반 정도의 충돌 사이클이 발생한다고 할 수 있겠다. 표 5에서 보면 하이브리드 쿼리 트리 프로토콜이 4-ary 쿼리 트리 프로토콜보다 항상 충돌 사이클이 1회 더 많은 것을 볼 수 있는데, 이는 가장 처음에 쿼리 메시지를 보낼 때 기존의 쿼리 트리 프로토콜에서는 0, 1 그리고 4-ary 쿼리 트리 프로토콜에서는 00, 01, 10, 11을 보내지만 하이브리드 쿼리 트리 프로토콜의 경우 empty string을 보내기 때문에 첫 번째 쿼리 시 무조건 충돌 사이클이 발생 하게 된다. (단 태그가 하나만 존재할 경우 충돌 사이클은 발생하지 않는다) 하이브리드 쿼리 트리 프로토콜에서 첫 쿼리 메시지가 무조건 충돌을 일으킬 수도 있지만 쿼리 축소 알고리즘을 적용 시킬 수 있으므로 첫 쿼리 메시지를 empty string로 결정하였다.

4.2 전체 태그 수에 따른 유휴 사이클의 수

그림 7은 제안한 기법이 4-ary 쿼리 트리 프로토콜 및 쿼리 트리 프로토콜에 비해 더 적은 수의 유휴사이클을 발생함을 보여 준다. 쿼리 축소 알고리즘에 의해 하이브리드 쿼리 트리 프로토콜이 4-ary 쿼리 트리 프로토콜보다 적은 수의 유휴 사이클을 발생시킬 것임은 자명하였다. 또한 쿼리 축소 알고리즘을 적용하여 기존의 쿼리 트리 프로토콜보다도 유휴 사이클이 줄어 들

표 5 충돌 사이클의 수

	25	50	75	100	125	150	175	200	300	500	1000
2QT	32.9	67.1	105.6	141.9	178.3	215.9	251.9	288.1	432.8	712	1439.5
4QT	16.3	33.5	52.7	70.8	88.9	107.7	125.7	143.9	217.6	356.3	722.2
HQT	17.3	34.5	53.7	71.8	898.9	108.7	126.7	144.9	218.6	357.3	723.2

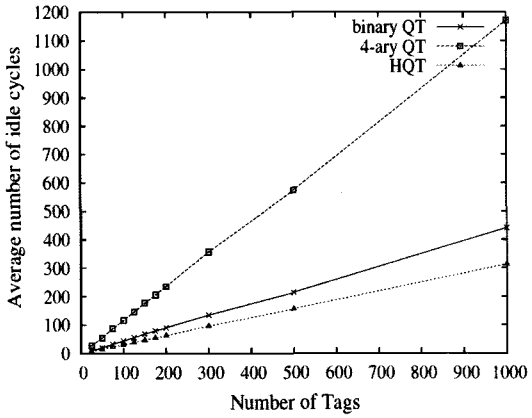

그림 7 전체 태그 수의 증가에 따른 유휴 사이클의 변화

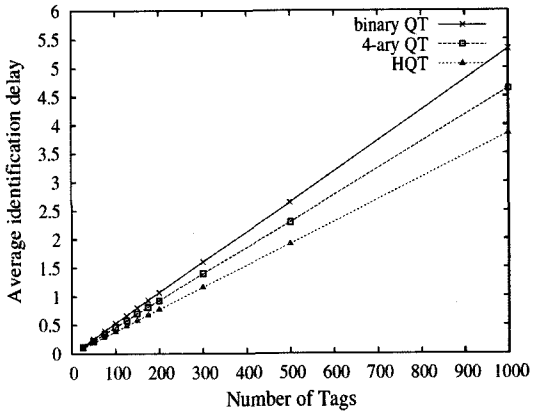

그림 8 전체 태그 수의 증가에 따른 태그 인식 시간의 변화

수 있었기 때문에 전체 성능 향상을 기대해 볼 수 있을 것이다. 그림 7에서 볼 수 있듯이 유휴 사이클의 경우에도 태그수가 증가할수록 유휴 사이클의 수도 선형으로 증가하게 되어 결과적으로 줄어드는 유휴 사이클의 비율이 일정함을 알 수 있다. 4-ary 쿼리 트리 프로토콜과 비교할 경우 약 27% 정도로 크게 유휴 사이클이 줄어들었고, 쿼리 트리 프로토콜과 비교할 경우 약 70% 정도로 줄어들었다. 더 자세한 실험 결과는 표 6에 정리였다.

4.3 전체 태그 수에 따른 모든 태그를 인식하는데 걸리는 지연시간

그림 8은 각 프로토콜들을 사용하였을 경우에 전체 태그 수에 따라 모든 태그를 인식하는 게 걸리는 총 지연시간을 보여 준다. 4-ary 쿼리 트리 프로토콜은 앞서 살펴보았듯이 쿼리 트리 프로토콜보다 충돌 사이클은 약 50% 정도 감소시킨 반면 유휴 사이클은 260%로 증가하였다. 하지만 유휴 사이클은 실제로 충돌 사이클에 비해서 태그 응답 전송 시간 만큼이 더 짧기 때문에 충돌 사이클이 줄어드는 것이 더 효과가 크게 된다. 따라서 지연시간에서 4-ary 쿼리 트리 프로토콜이 쿼리 트리 프로토콜보다 더 좋은 성능을 보이고 있다. 여기에쿼리 축소 알고리즘을 적용한 하이브리드 쿼리 트리 프로토콜은 유휴 사이클의 수를 훨씬 더 많이 줄일 수 있었기 때문에 가장 좋은 성능 보임을 알 수 있다. 보다 자세한 수치들은 표 7에 정리하였다.

5. 구현 이슈 및 비교분석

본 장에서는 논문에서 제안한 하이브리드 쿼리 트리 프로토콜을 실제로 구현을 하게 될 경우 생각해 보아야 할 문제들 및 어려운 점에 대해서 살펴보고, 제안한 기법과 다른 기법들을 비교하여 설명하도록 한다.

우선 RFID 리더에게는 반송파 감지(carrier sensing) 기술을 필요로 한다. 쿼리 축소 알고리즘의 적용을 위해서는 RFID 리더는 쿼리 메시지를 태그들에게 보내고 태그들의 응답을 감지하여 채널이 얼마 동안이나 사용중인 지에 대한 정보를 파악해야 하기에, 반송파 감지 기술은 RFID 리더에 반드시 적용되어야 하는 기술이다. 그러나 본래 RFID 리더기는 태그의 응답이 없는 경우와 충돌이 난 경우를 구분하기 위해 기본적으로 반송파 감지 기술이 필요하다. 이런 반송파 감지 기술의 경우 기존의 IEEE 802.11 무선랜 기술에서 이미 널리 사용되고 있기 때문에 적용에 있어서는 큰 어려움이 없다고

표 6 유휴 사이클의 수

	25	50	75	100	125	150	175	200	300	500	1000
2QT	9.9	19.2	32.6	44.0	55.4	67.9	79.0	90.1	134.9	214.0	441.5
4QT	27.9	54.5	87.3	116.4	145.9	177.1	206.1	235.7	357.0	573.1	1170.8
HQT	8.0	15.2	23.4	30.8	38.5	46.7	54.4	62.8	95.9	156.2	313.0

표 7 전체 태그 인식 시간

	25	50	75	100	125	150	175	200	300	500	1000
2QT	0.13	0.25	0.39	0.53	0.66	0.80	0.93	1.07	1.60	2.64	5.33
4QT	0.11	0.22	0.34	0.46	0.58	0.70	0.81	0.93	1.40	2.30	4.64
HQT	0.09	0.19	0.29	0.38	0.48	0.58	0.67	0.77	1.16	1.92	3.85

판단된다. 또한 태그들의 응답이 거리에 따라서 약간의 시간차를 보일 수 있다는 약점을 지적할 수 있다. 그러나 이러한 전파 딜레이는 본 논문에서 설정한 백오프 slot 단위 시간인 20μs 보다 매우 작기 때문에 무시할 수 있다. 예를 들면, 만약 태그가 RFID 리더로부터 10m만큼 떨어져 있다고 가정했을 때 전파 딜레이는 10(m)/300000000(m/s)(약 0.03μs)이고, 태그의 응답이 오는 시간까지 고려하면 RTT가 약 0.06μs 된다. 이 값은 백오프 slot 단위시간 길이인 20μs 보다 무시할 수 있을 만큼 작음을 알 수 있기에 전파 딜레이에 의한 시간차 문제는 무시할 수 있다고 판단된다.

위와 같이 RFID 리더의 경우에는 큰 문제가 보이지 않지만, 태그의 경우 구현에 추가적인 오버헤드가 따르는 것으로 판단된다. 우선 논문에서 제안한 기법 중 가장 중요한 백오프 태그 응답 알고리즘을 위해 태그는 RFID 리더가 쿼리 메시지를 보냈을 때 응답을 지연시켜 전송을 해야 하므로 얼마만큼 지연 시킬 것인지, 즉 백오프 타이머를 저장할 counter 등의 저장소가 필요하다. 본래 쿼리 트리 프로토콜이 태그에게 어떠한 저장소도 요구하지 않았던 것에 비하면 본 논문에서 제안하는 기법은 이런 약점을 갖는다고 볼 수 있다. 물론 현재 EPC Global[13]에서 정의하는 패시브 태그의 경우 EEPROM을 가지고 있기 때문에 실제 구현에서는 저장소에 의한 가격의 불리함은 없다고 본다. 태그가 응답을 지연시켜 전송을 하는 기술에 대해서는, RFID 시스템에서 태그의 응답은 산란(backscattering) 방식을 따르기 때문에 RFID 리더의 연속된 파동(continuous wave) 신호만 있다면 지연 전송은 에너지 저장과 무관하게 사용 될 수 있다. 산란 방식이란 거울이 빛을 반사하는 것과 비슷하게 태그가 RFID 리더의 파동 신호를 다시 이용하여 응답을 하는 것이다. 따라서 이 부분에서는 추가적인 오버헤드는 없다. 결론적으로 저자들이 제안한 기법은 태그에 추가적인 저장소를 필요로 하기에 그 만큼의 오버헤드가 늘겠지만 실제 구현에 있어서의 가격측면에서는 불리함을 갖지는 않는다.

다음 표 8은 저자들이 제안한 하이브리드 쿼리 트리 프로토콜을 기존의 태그 충돌 방지 프로토콜과 비교하여 정리한 것이다.

첫 째로 태그에서 필요한 저장소에 관해서는 본 논문에서 제안한 기법의 기본 알고리즘인 쿼리 트리 기법을 제외하고 다른 기법에서는 모두 필요로 한다. 모두가 저장할 카운터 값 등이 존재하기 때문이다. 태그에서의 타이머 기술 역시 마찬가지 인데, 쿼리 트리 기법은 패시브 태그의 동작이 RFID 리더에 종속적으로 동작이 가능하지만 이진 트리 분할 기법 및 제안 기법에서는 저장소에 저장한 값의 감소가 필요하다. 알로하 기법의 경우도 쿼리 트리와 마찬가지로 타이머를 필요로 하지는 않다. 무작위 선출 부분에서는 이진 트리 기법과 알로하의 경우 RFID 리더의 쿼리에 응답을 할 것인가 말 것인가를 결정할 때 필요하기 때문에 태그에 필요하게 되고 이는 복잡도를 높이는 주요한 요인이 된다. 또한 도입 부분에서 언급하였듯이 태그 기아 문제는 알로하 방식에서만 존재한다. 반송파 감지 기술은 이미 설명하였듯이 태그가 응답하지 않은 경우와 두 개 이상의 태그가 응답하여 충돌이 발생한 경우를 구분하기 위해서 RFID 리더기에 반드시 필요한 기술이다. 이진 트리 분할 기법 및 알로하 기법은 무작위 선출 기술을 필요로 하기 때문에 복잡도가 높은 반면, 태그에 단순한 응답만 요구하는 쿼리 트리 기법이 가장 복잡도가 낮을 것이고, 그에 비해 태그에서 지연 전송을 요구하는 본 논문이 제안하는 기법의 복잡도는 중간 정도로 표시할 수 있다.

표 8 기존 프로토콜과의 비교

	쿼리 트리	이진 트리 분할	알로하	제안 기법
저장소 (태그)	X	O	O	O
타이머 (태그)	X	O	X	O
무작위 선출	X	O	O	X
태그 기아 문제	X	X	O	X
반송파 감지	O	O	O	O
복잡도	낮음	복잡	복잡	중간

6. 결 론

본 논문에서는 RFID 시스템에서 4-ary 탐색 트리 알고리즘과 slotted 백오프 태그 응답 기법을 활용한 하이브리드 쿼리 트리 프로토콜을 제안하여 보다 효과적으로 RFID 태그들의 충돌을 중재할 수 있었다. 제안한 알고리즘은 RFID 리더에게는 쿼리 축소 알고리즘을 위해 반송파 감지(carrier sensing) 기술이 태그에게는 백오프 타이머를 저장하기 위한 저장소(counter)와 응답할 때 전송을 지연하기 위한 타이머를 필요로 한다.

제안한 기법에서는 태그 구현의 복잡도가 기본적인 쿼리 트리 기법보다는 상승한다는 단점이 있으나, 모의실험 결과 및 수학적 분석은 제안된 기법의 성능이 기존의 다른 기법들에 비해 더 뛰어남을 보여준다. 현재는 RFID 시스템에 리더가 하나만 존재하는 것이 아니라 다수가 존재하는 경우, 또 태그들이 고정 또는 이동하는 시나리오를 고려한 프로토콜 설계를 위한 연구가 진행 중이다.

참 고 문 헌

[1] Capetanakis, J., "Tree algorithms for packet broadcast channels," in the IEEE Transactions on Information Theory, Vol.25, No.5, pp. 505-515, 1979.

[2] Law, C., Lee, K. and Siu, K.Y., "Efficient Memoryless Protocol for Tag Identification," in Proc. of the 4th ACM International workshop on Discrete algorithms and methods for mobile computing and communications, pp. 75-84, 2000.
[3] Myung, J. and LEE, W., "Adaptive splitting protocols for RFID tag collision arbitration," in Proc. of the 7th ACM international symposium on Mobile ad hoc networking and computing, pp 202-213, 2006.
[4] Vogt, H., "Efficient object identification with passive RFID tags," in the Lecture Notes In Computer Science, Vol.2414, archive Proc. of the 1st International Conference on Pervasive Computing, pp. 98-113, 2002.
[5] Lee, S., Joo, S.D. and Lee, C.W., "An enhanced dynamic framed slotted aloha algorithm for RFID tag identification," in Proc. of the 2nd ACM Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 166-174, 2005.
[6] Bonuccelli, M.A., Lonetti, F. and Martelli, F., "Tree Slotted Aloha: a New Protocol for Tag Identification in RFID Networks," in Proc. of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 26-29, 2006.
[7] Zhou, F., Chen, C., Jin, D., Huang, C. and Min, H., "Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems," in Proc. of the ACM International Symposium on Low Power Electronics and Design, pp. 357-362, 2004.
[8] Abramson, N., "The Aloha system - another alternative for computer communications," in Proc. of the AFIPS, Vol.36, pp. 295-298, 1970.
[9] Roberts, L., "Aloha packet system with and without slots and capture," in the ACM Computer Communication Review, Vol.5, No.2, pp. 28-42, 1975.
[10] Rosen, K., Discrete Mathematics and Its Applications, 5th Edition, McGraw-Hill, New York, 2003.
[11] Mathys, P. and Flajolet, P., "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access," in the IEEE Transactions on Information Theory, Vol.31, No.4, pp. 217-243, 1985.
[12] NS-2, URL http://www.isi.edu/nsnam/ns.
[13] EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.0.9, EPCglobal, January 2005.

류 지 호
서울대학교 전기컴퓨터공학부 대학원생 박사과정. 2005년 2월 한국과학기술원 전자전산학부 졸업. 2005년 3월~현재 서울대학교 전기컴퓨터공학부 박사과정 관심분야는 무선랜 QoS, 게임이론, 네트워크 이동성, RFID systems

이 호 진
서울대학교 전기컴퓨터공학부 대학원생 박사과정. 2003년 2월 서울대학교 전기컴퓨터공학부 졸업. 2003년 3월~현재 서울대학교 전기컴퓨터공학부 박사과정 관심분야는 무선 네트워크, WiBro, RFID systems

석 용 호
LG 전자 기술원 정보기술연구소 PAN 그룹 선임연구원. 2000년 2월 한동대학교 전산전자공학부 졸업. 2002년 2월 서울대학교 전기컴퓨터공학부 석사. 2005년 8월 서울대학교 전기컴퓨터공학부 박사 관심분야는 인터넷 트래픽 측정, 트래픽 엔지니어링, 무선랜 QoS, 멀티홉 무선 네트워크, RFID systems

권 태 경
서울대학교 전기컴퓨터공학부 조교수. 1993년 서울대학교 컴퓨터공학과 공학사 1995년 서울대학교 컴퓨터공학과 석사 2000년 서울대학교 컴퓨터공학과 박사 2004년~현재 서울대학교 전기컴퓨터공학부 조교수. 관심분야는 센서 네트워크, 무선 네트워크, IP 이동성, 유비쿼터스 컴퓨팅, 미래 인터넷

최 양 희
서울대학교 전기컴퓨터공학부 교수. 1975년 서울대학교 전자공학과 공학사. 1977년 한국과학기술원 전자공학과 석사. 1984년 프랑스 국립전기통신대학 컴퓨터 공학 박사. 1991년~현재 서울대학교 전기컴퓨터공학부 교수. 2005년~현재 한국과학기술한림원 회원. 2006년~현재 미래인터넷 포럼 회장. 관심분야는 무선 모바일 네트워크, 미래 인터넷, IP기반 멀티미디어 트래픽