FQTR : RFID 시스템을 위한 새로운 하이브리드 태그 충돌 방지 프로토콜

(FQTR : Novel Hybrid Tag Anti-Collision Protocols in RFID System)

정 승 민 † 조 정 식 † 김 성 권 ††
(Seung-Min Jung) (Jung-Sik Cho) (Sung-Kwon Kim)

요 약 RFID 기술은 바코드를 대체할 자동인식 기술로서 인식 속도가 빠르고 비접촉 방식으로 오염에 강한 장점을 가지고 있다. 그러나 RFID 기술의 대중화를 위해서는 선결되어야할 난제들이 있다. 이 중 본 논문에서는 다중 태그 식별 문제를 다룬다. 다중 태그를 인식하고 충돌을 방지하는 문제는 RFID 시스템의 성능에 직접적인 영향을 준다. 본 논문에서는 기존의 알고리즘들을 분석하고 개선하된 새로운 태그 충돌 방지 알고리즘을 제안한다. 제안된 알고리즘은 ALOHA 기반 알고리즘과 QT 기반 알고리즘의 하이브리드 형태로 분배와 인식 과정에 적합한 특성들을 혼합하여 구성하였다. 분배 과정에서는 ALOHA 방식을 이용하여 인식과정에서 한 프레임에 인식해야할 태그 수를 줄였다. 이는 Tree 의 깊이가 깊어져 지연을 일으키는 문제를 해결하였다. 인식 과정에서는 QT 기반 알고리즘을 이용하여, ALOHA 방식에서 모든 태그를 인식하지 못하는 문제를 해결하였다. 또한, 실제 RFID 환경을 분석하여 더 좋은 성능을 보이도록 태그ID 의 역순으로 인식하는 QTR 알고리즘을 적용하였다. 본 논문에서 제안하는 FQTR 알고리즘은 시뮬레이션 결과 기존의 알고리즘에 비해 뛰어난 성능을 보여주었다.

키워드 : RFID 시스템, RFID 태그, 태그 충돌 문제, 태그 충돌 방지 프로토콜, 알로하 알고리즘, 쿼리 트리 알고리즘

Abstract RFID, Radio Frequency Identification, is a technology of automated identification replacing bar-code. RFID technology has advantages that it recognizes fast and it is strong to contamination using wireless communication. However, there are difficult problems that should be solved for popularization of RFID. Among of these, tag anti-collision problem is dealed in this paper. It affected the performance of RFID system directly. This paper analyzes conventional algorithms and proposes new algorithms of tag anti-collision. The algorithm proposed was composed of appropriate properties to each phase of distribution and recognition as hybrid between ALOHA-based algorithm and QT-based algorithm. At phase of distribution, the number of tags recognizing at a frame was reduced using ALOHA-based algorithm. It addressed the delay problem because of deep depth of tree. At phase of recognition, it solved ALOHA-based chronic problem that couldn't recognize all the tags sometimes. Moreover, QTR algorithm that recognize by reversed tag IDs was adopted for the performance. The FQTR algorithm proposed in this paper showed brilliant performance as compared with convention algorithms by simulation.

Key words : RFID system, RFID tag, RFID tag collision problem, Tag anti-collision protocol, ALOHA algorithm, QT algorithm

· 이 논문은 2009년도 중앙대학교 우수연구자연구비 지원에 의한 것임

† 학생회원 : 중앙대학교 컴퓨터공학과
smjung@alg.cse.cau.ac.kr
mfg@alg.cse.cau.ac.kr
†† 종신회원 : 중앙대학교 컴퓨터공학과 교수
skkim@cau.ac.kr

논문접수 : 2009년 1월 14일
심사완료 : 2009년 5월 16일

정보과학회논문지: 소프트웨어 및 응용 제36권 제7호(2009.7)

1. 서 론

RFID 기술이란 소형 IC 칩(chip)과 무선 통신을 위한 안테나(antenna)를 내장한 태그(tag)를 사물에 부착하여 다른 기기와 라디오 주파수 대역을 사용하여 통신을 하는 기술을 말한다. RFID 기술은 바코드 방식에 비하여 다양한 장점을 가지고 있다. 첫째, 무선 방식으로 통신하므로 바코드와 같이 제품 표면에 코드를 표시할 필요가 없어 오염의 우려가 적다. 둘째로 바코드 스캐너처럼 하나씩 근접시키지 않아도 되기 때문에 인식 작업이 편리하다. 셋째, RFID 기술은 짧은 시간 안에 다수의 태그를 인식할 수 있어 속도가 빠르며, 넷째로 바코드에 비하여 더 많은 정보를 입력할 수 있는 장점도 가지고 있다. 마지막으로 바코드에서는 제품 각각에 고유한 바코드를 인쇄하지 않고 같은 종류의 제품에는 같은 식별 코드를 사용하지만 RFID에서는 모든 제품에 고유한 식별 코드를 넣음으로서 판매나 재고 관리 측면에서 훨씬 정확하고 실제적으로 제품을 제어할 수 있다. 이런 RFID 기술은 기존 바코드를 대체할 차세대 유비쿼터스 사회의 핵심 기술이라 할 수 있으며 기존의 바코드를 대체하여 물품 관리를 네트워크화 및 지능화함으로써 유통 및 물품 관리뿐만 아니라 보안, 안전, 환경 관리 등에 혁신을 선도할 것으로 기대된다[1].

그러나, RFID 기술의 대중화를 위해서는 우선 해결되어야할 난제들이 있다. 전자태그의 높은 가격 문제, 보안 및 프라이버시 문제, 태그 식별자의 코드 표준화 문제, 그리고 다중 태그 식별 문제 등이 그 대표적 예이다[1]. 이 중 본 논문에서는 다중 태그 식별 문제를 다루도록 한다.

리더가 다수의 태그에게 질의할 때는 브로드캐스트(broadcast) 방식으로 전파를 전송하며, 이를 포워드 링크(forward link)라고 한다. 이 질의는 리더의 전파 범위 안에 있는 태그만 들을 수 있으며, 리더의 전파 범위 안에 있는 리더의 질의를 수신한 태그는 이 질의에 대하여 응답을 하게 된다. 이것을 리턴 링크(return link)라고 한다. 이때 태그들은 리더의 질의에 응답하도록 설계되어 있으므로 동시에 응답을 하고 전파 간 충돌(collision)이 발생하게 되며 태그 충돌이라는 문제가 생긴다. 이러한 충돌은 RFID 시스템의 인식률에 큰 영향을 미치므로, 효과적인 태그 충돌 조정 과정이 요구된다.

태그 충돌 방지 프로토콜은 하나의 리더와 다수의 태그가 통신 하는 과정에서 생기는 충돌을 방지하는 데 목적을 둔다. 태그 충돌 방지 프로토콜은 크게 2가지, 알로하(ALOHA) 기반 프로토콜[2-4]과 트리(tree) 기반 프로토콜[2,5,6]로 나누어진다. 알로하 기반 프로토콜이란 보통 슬롯티드 알로하(slotted ALOHA)를 말하며 이는 시간을 슬롯(slot) 단위로 나누어 태그를 무작위로 나누어 한 슬롯에 하나만 응답하게 하여 리더가 인식하는 프로토콜이다. 하지만 이 방식은 무작위라는 확률의 불확실성에 기초를 두고 있기 때문에 리더 식별 영역 내의 모든 태그를 인식하지 못할 수도 있고 모든 태그를 인식하는 데 걸리는 시간을 예측하기 어렵다.

한편 트리 기반 프로토콜은 태그의 고유한 ID를 사용하여 태그 인식 과정을 진행하면서 트리를 만든다. 이것은 모든 태그를 인식할 수 있고 그 과정을 예측할 수 있는 장점이 있지만 비슷한 ID를 갖는 태그들이 많을 때는 트리를 만드는 중 충돌이 많이 발생하여 트리가 깊어지고 결국 태그 인식 시간이 오래 걸리게 되는 단점을 가지고 있다.

1장에서는 RFID 시스템과 태그 충돌 방지 알고리즘에 대해 알아보았다. 2장에서는 태그 충돌 방지 알고리즘과 관련된 기존 연구들에 대해서 살펴보고, 3장에서는 새로운 충돌방지 알고리즘을 제안한다. 4장에서 본 논문에서 제안하는 알고리즘의 시뮬레이션을 통하여 다른 충돌 방지 알고리즘과 성능 비교를 하고, 5장을 마지막으로 결론을 맺는다.

2. 관련 연구

기존의 충돌 방지 알고리즘은 크게 확률적 충돌 방지(*stochastic collision resolution*) 방법과 결정적 충돌 방지(deterministic collision resolution) 방법으로 나눌 수 있다. 먼저 확률적 충돌 방지 방법은 보통 알로하(ALOHA) 알고리즘을 기반으로 하고 있다. 한편 결정적 충돌 방지 방법은 QT(Query Tree) 알고리즘을 기반으로 하고 있다.

또한 최근에는 위의 알고리즘을 혼합한 형태의 알고리즘들과 특수한 RFID 환경에서 좋은 성능을 보이는 알고리즘들이 제안되고 있다.

2.1 ALOHA 기반 알고리즘

ALOHA는 시분할 다중접속(TDMA) 기술을 사용해 무선전송을 하는 프로토콜이다. 만약 충돌이 일어나면 일정시간 후에 재전송된다.

ALOHA 계열의 알고리즘에서 태그가 응답할 슬롯(slot)은 무작위로 선택된다[2]. 따라서 두개 이상의 태그가 지속적으로 같은 슬럿을 선택한다면, 계속 태그를 인식하지 못하게 되는 태그 부족 문제(Tag Starvation Problem)이 발생할 수 있으며, 이는 ALOHA 기반의 모든 알고리즘에서 공통적으로 나타날 수 있는 문제이다.

ALOHA 알고리즘에서 좋은 성능을 내기 위해서는 인식해야 하는 태그 수의 예측이 중요하다. 슬롯에 비하여 태그 수가 너무 많으면, 많은 충돌이 발생하고, 반대의 경우에는 무응답 횟수가 늘어나게 된다. 따라서 좋은 성능을 내기 위한 태그 수의 추정 알고리즘에 대한 연구 또한 중요한 이슈이다.

2.1.1 FS-ALOHA 알고리즘

프레임드 슬롯티드 알로하[4]란 슬롯티드 알로하의 변형이다. 여기서 프레임(frame)이란 리더가 명령을 전송하고 다음 명령 전송까지의 시간을 의미한다. 하나의 프레임은 여러 개의 슬롯으로 구성되고 태그는 리더로부터 전송 받은 프레임 안에서 슬롯을 임의로 선택하고 각자 선택한 슬롯에 ID를 전송하는 방식이다. 이 방식에서 리더는 태그가 슬롯 내에 ID가 충돌 없이 완벽하게 전송될 때만 태그를 인식한다. 인식된 태그는 리더로부터 ACK 명령을 받고 다음 프레임에 참여하지 않는데, 인식된 태그가 다른 태그들의 인식에 방해가 안 되게 하기 위해서다. 한 프레임 안에서 태그를 인식하지 못하면 리더는 인식 못한 태그들을 위해 다시 인식 과정을 시작한다. 하지만 여기서 중요한 점은 기본 프레임드 슬롯티드 알로하는 프레임의 크기가 고정이라는 것이다.

표 1은 프레임드 슬롯티드 알로하 동작의 예이다. 리더는 질의 시 프레임의 크기를 2로 했으며 이것은 고정되어 있다고 가정한다. 이 질의를 받은 태그는 프레임 크기 2, 즉 슬롯 1 또는 2 중 임의로 선택을 해서 대답하게 된다. 이 예제에서는 Tag1과 Tag3은 슬롯 1을 선택했고 충돌이 발생했다. 한편 Tag2는 홀로 대답했기 때문에 인식 되었고 리더로부터 인식되었다는 명령을 받고 다음 프레임에는 참여하지 않는다.

표 1 프레임드 슬롯티드 알로하의 동작과정

Forward Link	Request [2]	Slot 1	Slot 2	Ack [1011]	Request [2]	Slot 1	Slot 2
Return Link		Collision	Identification			No Response	Collision
Tag1		0010					0010
Tag2			1011				
Tag3		0011					0011

2.1.2 DFS-ALOHA 알고리즘

다이나믹 프레임드 슬롯티드 알로하[4]는 RFID 시스템에서 태그의 충돌을 해결하기 위해 사용되는 충돌 방지 프로토콜 중 대표적인 것이다. 이것은 현재 ISO/IEC 18000-6 Type A와 Type C가 된 EPCglobal Class 1 Generation 2에서 실제 사용되고 있기도 하다.

다이나믹 프레임드 슬롯티드 알로하는 프레임드 슬롯티드 알로하와 달리 각 프레임의 크기가 변할 수 있다. 여기서는 리더가 태그에게 ID 전송 요구를 할 때마다 전체 슬롯 수인 프레임의 크기(frame size: FS)를 함께 전송한다. 태그는 리더로부터 ID 전송 요구를 수신하면 전송받은 프레임 크기 안에서 자신이 전송할 슬롯을 랜덤하게 결정하고 자기 차례까지 기다렸다가 전송을 시도한다.

표 2는 다이나믹 프레임드 슬롯티드 알로하 동작의 예이다. 리더는 질의 시 프레임의 크기를 2로 했다고 가정한다. 이 질의를 받은 태그는 프레임 크기 2, 즉 슬롯 1 또는 2 중 임의로 선택을 해서 대답하게 된다. 이 예제에서는 Tag1과 Tag3은 슬롯 1을 선택했고, 나머지 태그는 모두 슬롯 2를 선택하여 충돌이 발생했다. 충돌이 많이 발생하여 태그 추정 알고리즘을 통하여 다음 프레임에서는 슬롯 수를 전 프레임의 2배인 4개로 조정하였다. 다음 프레임에서 Tag2, Tag5가 슬롯 1을 선택하여 또 충돌이 발생하였지만, 슬롯 2에서는 Tag1이 홀로 대답하여 인식되었다. 이와 같이 충돌과 무응답의 수에 따라 슬롯 수를 조정해가며 프레임을 반복한다.

표 2 다이나믹 프레임드 슬롯티드 알로하의 동작 과정

Forward Link	Request [2]	Slot 1	Slot 2	Request [4]	Slot 1	Slot 2	Ack [1010]	Slot 3	Slot 4
Return Link		C	C		C	I		N	C
Tag1		0010				0010			
Tag2			1011		1011				
Tag3		0011							0011
Tag4			0101						0101
Tag5			1010		1010				

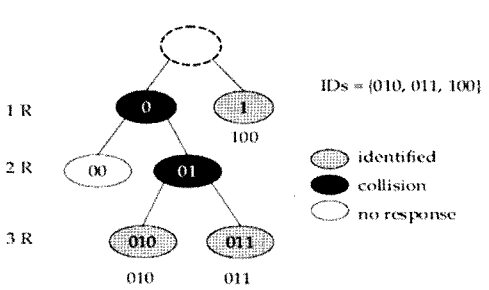
C : Collision, I : Identification, N : No response

2.2 QT 기반 프로토콜[5]

쿼리 트리 프로토콜은 충돌 방지 프로토콜에서 트리 기반 프로토콜 중 대표적인 것이다. 리더는 태그에게 태그의 ID를 요청할 때 프리픽스(prefix)를 함께 전송한다. 태그는 자신의 ID 앞부분과 프리픽스가 같은 지 확인하고 동일하면 자신의 ID를 리더에게 응답한다.

이 때 리더의 질의에 대해서는 다이나믹 프레임드 슬롯티드 알로하와 마찬가지로 어떤 태그도 대답하지 않으면 무응답이, 하나의 태그가 응답하면 인식이, 둘 이상의 태그가 동시에 응답하면 충돌이 발생한다. 리더는 충돌이 일어났을 때 충돌이 발생한 걸 알고, 결국 같은 프리픽스의 태그가 여러 개 있다는 것을 알게 된다. 이 경우 방금 전송한 프리픽스 뒤에 '0'과 '1'을 붙인 2개의 새로운 k+1 비트의 프리픽스 Pk+1을 만들어 큐(queue)에 넣는다. 큐에 넣은 프리픽스는 나중에 다시 질의함으로써 태그의 식별 ID를 가지고 트리를 만든다. 큐의 초기 값은 '0'과 '1'이다.

그림 1은 식별 ID가 각각 '010', '011', '100'인 3개의 태그가 있다고 가정하고 쿼리 트리 프로토콜을 실행한 예제이다. 초기 큐에는 프리픽스 {'0', '1'}이 설정되어 있으며 리더는 큐에서 프리픽스를 꺼내 태그에게 질의한다. 먼저 '0'을 질의 했을 때 태그 '010'과 '011'이 자신의 ID와 프리픽스가 일치하므로 동시에 응답하게 되어 충돌이 발생한다. 이것을 통해 리더는 '0'으로 시작하는 태그가 2개 이상 있다고 보고 큐에 '00'과 '01'을 넣



Query	Response
0	Collision
1	Readable
00	Idle
01	Collision
010	Readable
011	readable

그림 1 쿼리 트리 프로토콜 동작의 예

는다. 다음에는 큐에서 프리픽스 '1'을 꺼내와 질의하게 된다. 여기에 맞는 태그는 '100', 1개이므로 이것만 응답하게 되고 리더에게 정상적으로 인식된다. 이렇게 해서 1라운드가 끝나고 전에 큐에 넣은 '00'과 '01'을 대상으로 다시 라운드를 시작하게 된다.

다음 2라운드에서 리더는 '00'과 '01'을 프리픽스로 해서 질의하고 '00'으로 시작하는 태그는 없으므로 무응답이 된다. 후에 '01'을 질의 하고 충돌이 나서 '010'과 '011'을 큐에 넣어 진행한다. 이런 식으로 진행해 나가다 큐가 비었을 때 태그의 ID 트리가 완성된 것으로 보고 종료한다.

2.3 QTR 알고리즘

QTR(Query Tree Algorithm with Reversed tag IDs) 알고리즘[7]은 태그 ID의 특성에 영향을 받지 않는 트리 알고리즘이다. 이 알고리즘은 순차적인 태그 배치환경 상황에서 태그 ID의 프리픽스가 많은 부분 일치한다는 것에 기인해 고안되었다. QT 알고리즘이 앞부터 뒤로 순차적으로 인식하는 것에 비하여, QTR 알고리즘은 태그 ID를 역순으로 인식한다. 태그 ID를 역순으로 인식할 시, 태그 배치 특성에 관계없이 모두 랜덤 성향을 갖으며, 평균적인 성능을 보일 수 있게 된다. 이것은 마치 한쪽으로 쏠려있는 트리를 균형있게 양쪽으로 분배한 효과이다.

QTR 알고리즘은 쿼리 트리 알고리즘의 치명적인 단점에 대한 해결책을 제시하였지만, 실제 성능에서 큰 효율은 보여주지 못하였다. 더욱이, 인식하려는 태그 ID가 많을 경우(일치하는 상위 비트 수가 적을 경우)일수록 좋지 않은 성능을 보여주었다.

그러나, QTR 알고리즘은 QT과정으로 인식하는 FQT 알고리즘의 문제까지 해결함으로써, 더 좋은 성능의 알고리즘의 발전 가능성을 보여주었다는 데에 의의가 있다.

3. FQTR 알고리즘

기존의 태그 충돌 방지 프로토콜은 일반적으로 알로하 기반 충돌 방지 프로토콜과 트리 기반 충돌 방지 프로토콜, 2 부류로만 나눠왔다. 앞서 설명한 다이나믹 프레임드 슬롯티드 알로하는 알로하 기반 충돌 방지 프로토콜 중 대표적인 것이며, 쿼리 트리 프로토콜은 트리 기반 충돌 방지 프로토콜의 대표라 할 수 있다. 하지만 둘 다 리더가 인식하려는 태그들이 리더 주위에 많이 밀집되어 있는 경우 충돌이 많아 태그 인식 시간을 증가시키는 공통된 문제를 가지고 있다.

본 논문에서 제안하는 프레임드 쿼리 트리 리버스드 (framed query tree with reversed tag IDs : FQTR)은 다이나믹 프레임드 슬롯티드 알로하와 쿼리 트리 프로토콜의 혼합으로 인식하려는 태그들을 그룹별로 분산시켜 충돌을 줄임으로서 인식 시간을 줄이는데 목적을 둔다.

3.1 프레임드 쿼리 트리 리버스드 프로토콜의 발생

위 절에서 설명한 제안된 모든 프로토콜은 분배와 인식 과정으로 나누어 생각할 수 있다.

분배 과정(phase)은 밀집 분포해 있는 태그들을 그룹화하는 과정이다. 분배 과정은 수행하려는 알고리즘에 따라 무작위 혹은, 같은 성향을 가지는 태그들끼리 묶어서 수행속도를 빠르게 하려는 목적을 가지고 있다.

인식 과정(phase)은 분배 과정에서 작은 그룹으로 나누어진 태그들을 인식하는 과정을 의미한다.

다이나믹 프레임드 슬롯티드 알로하 프로토콜은 분배 과정을 반복해서 수행하여 하나의 그룹에 하나의 태그가 속할 때까지 나누고, 인식 과정은 그룹(슬롯)과 태그의 관계가 1-1인 상태에서 인식을 수행한다. 1-N인 관계에서는 충돌이 발생하고, 이는 다음 차례(프레임)에 다시 분배 과정을 거치게 된다. 이 프로토콜은 무작위 성질에 의존하여 분배에는 우수한 성능을 보이지만, 인식 과정에서는 좋은 성능을 보이지 못하였다. 특히, 모든 태그를 인식하지 못하는 문제가 발생하였다.

이에 반해 쿼리 트리 프로토콜은 인식 - 분배 과정을 반복적으로 수행한다. 인식을 위하여 프리픽스를 전송하고, 인식이 실패하면 분기하여 분배 과정을 수행한다. 쿼리 트리 프로토콜은 선인식 후분배로 인식 과정에 우선을 둔 프로토콜로 리더의 인식 범위 안의 모든 태그를 인식할 수 있다는 장점을 가지고 있다.

위에서 살펴보듯이 다이나믹 프레임드 슬롯티드 알로하 프로토콜과 쿼리 트리 프로토콜은 각각 분배와 인식 과정에서 우수한 성능을 보인다. 이는 프로토콜의 태생적 성질이다. 따라서 개선된 프로토콜은 두 프로토콜의 합성을 통하여 만들어질 수 있다.

이에 본 논문에서는 분배 과정에서는 다이나믹 프레임드 슬롯티드 알로하 프로토콜을 인식 과정에서는 쿼리 트리 프로토콜을 차용한 하이브리드 프로토콜을 제안한다. 이 프로토콜은 태그 인식의 전체 성능 과정에서 우수한 성능을 보일 것이다.

또한, 실제 물류 환경에서 쿼리 트리 프로토콜은 프리픽스부터 인식하는 것보다 서픽스부터 인식하는 것이 더 유리하다[7]. 랜덤한 태그 ID 환경에서 두 방식의 성능은 동일하지만, 순차적인 태그 ID 환경에서 프리픽스부터 인식시에는 많은 오버헤드가 발생하게 된다. 그러나 서픽스부터 인식하는 쿼리트리 프로토콜은 랜덤 태그 ID 환경 하에서와 동일한 성능을 유지하게 된다. 따라서 쿼리 트리 프로토콜은 서픽스부터 인식하는 것이 성능 개선에 유리하다.

본 논문은 쿼리 트리 리버스드[7] 개념을 참고하여, 제안하는 하이브리드 알고리즘에 반영하였다. 따라서, 본 논문에서 제안하는 알고리즘은 분배 과정에서는 다이나믹 프레임드 슬롯티드 알로하 프로토콜을, 인식 과정에서는 쿼리 트리 리버스드 프로토콜을 차용한다. 제안하는 알고리즘의 개념은 그림 2와 같다.

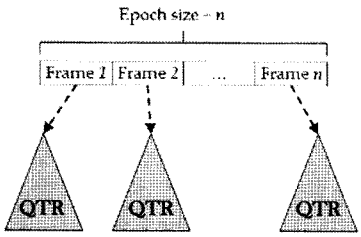


그림 2 FQTR 알고리즘의 동작 원리

3.2 프레임드 쿼리 트리 리버스드 프로토콜의 동작

실제 동작 흐름은 리더는 그림 3과, 태그는 그림 4와 같다. 먼저 리더는 태그에게 ID 전송 요구를 할 때 전체 프레임 개수인 에폭 크기(epoch size)를 함께 전송한다. 이때 전송하는 에폭 크기는 작은 크기로 시작하며 첫 프레임 테스트를 통해 태그 수를 추정하면서 점차 알맞은 크기로 커진다. 알맞은 에폭 크기까지 도달하면 태그 인식을 진행 한다.

태그 인식 과정은 에폭 크기를 받은 태그들이 랜덤하게 자신이 참가할 프레임을 결정하면서 시작한다. 그 후 태그들은 리더가 자신의 프레임에 대해서 질의할 때만

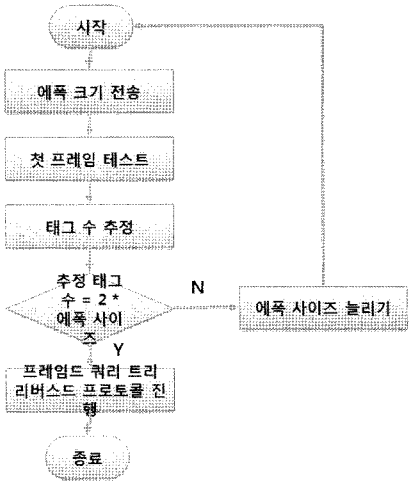


그림 3 프레임드 쿼리 트리 리버스드 프로토콜의 리더 흐름

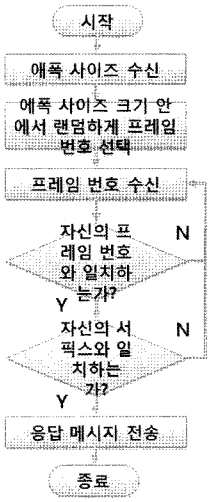


그림 4 프레임드 쿼리 트리 리버스드 프로토콜의 태그 흐름

응답하고 자신의 프레임이 아니면 기다린다. 리더는 각 프레임 안에서 쿼리 트리 프로토콜을 이용하여 인식 과정을 시행한다. 이 때, 리더는 태그에게 ID의 서픽스뿐만 아니라 프레임의 번호도 같이 전송한다.

태그는 프레임의 번호가 자신이 선택한 프레임의 번호와 같은 지 확인한 후 기존의 쿼리 트리 프로토콜과 동일한 방식으로 전송된 서픽스를 보고 일치할 때만 자신의 ID를 전송한다. 리더는 한 프레임에서 쿼리 트리 리버스드 프로토콜 인식 과정을 거쳐 그 프레임에 속한 모든 태그를 인식한 후에 다음 프레임으로 넘어가게 된다. 이 과정을 에폭 크기 안에 있는 모든 프레임에 대해서 반복하여 수행한다.

한편 기존 알로하 기반 프로토콜과 큰 차이점은 프레

임드 쿼리 트리 리버스드 프로토콜에서는 다이나믹 프레임드 슬롯티드 알로하를 큰 그림으로 사용하지만 실제 태그 인식은 쿼리 트리 리버스드 프로토콜로 진행하므로 알로하 기반 프로토콜처럼 리더가 태그에게 인식되었음을 알리는 명령인 ACK를 보내지 않는다.

그림 5의 예제에서는 리더가 인식하려는 태그가 8개이고 에폭 크기가 4일 때, 프레임드 쿼리 트리 리버스드 프로토콜의 인식 과정을 보여준다. 먼저 리더는 태그들에게 에폭 크기를 전송해서 랜덤하게 자신의 프레임을 선택한다. 예제에서 Frame1에 3개의 태그, Frame2에는 태그가 없고, Frame3에는 2개의 태그, Frame4에는 3개의 태그들이 선택하였다.

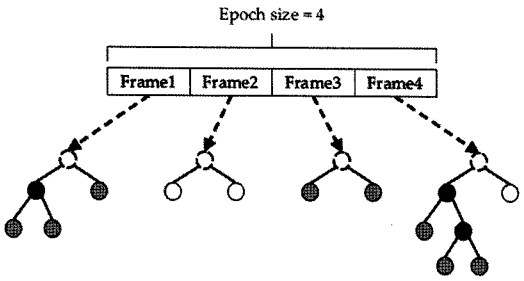


그림 5 프레임드 쿼리 트리 리버스드 프로토콜 인식 과정 예

한편 제안하는 프레임드 쿼리 트리 리버스드 프로토콜에서는 적절한 에폭 크기를 정하는 것이 성능 향상에 있어서 매우 중요하다. 직관적으로 쿼리 트리 리버스드 프로토콜을 시행 했을 때 트리의 깊이가 2 이상이 안 되게 하는 에폭 크기가 가장 좋은 성능을 나타낸다[8]. 쿼리 트리 리버스드 프로토콜의 초기 큐에 있는 {'0', '1'}을 시행 시 '0'으로 끝나는 태그 하나와 '1'로 끝나는 태그 한 개, 총 두 개의 태그가 인식 되는 것이 이상적이기 때문이다. 그림 5에서 Frame3이 이런 경우에 해당한다.

표 3, 표 4는 각각 리더와 태그에서의 프레임드 쿼리 트리 리버스드 프로토콜 의사 코드이다.

4. 시뮬레이션

4.1 시뮬레이션 시스템

4.1.1 태그의 특성

시뮬레이션에 사용되는 태그 ID는 EPCGlobal의 GID-96에 따라 생성하였다. 헤더(8비트), 회사(28비트), 제품군(24비트), 시리얼넘버(36비트)의 총 96비트로 구성된 GID-96은 물류분야에서의 사실상 표준(de facto standard)이며, 헤더는 '00110101'으로 고정되어 있다[9]. 본 논문의 시뮬레이션에서는 태그가 부착된 물류 시스템의 흐름을 고려하여 표 5의 4가지로 분류하였다.

물류 시스템의 흐름은 공장에서 시작하여 도매, 소매,

표 3 FQTR의 의사코드(리더)

```
epoch size = initialize epoch size by user
has new epoch size = True

while( has new epoch size is True ):
    has new epoch size = False
    frame number = 0
    send INIT command with epoch size

    while( frame number < epoch size ):
        collision number = 0
        Q = {'0', '1'}

        while( Q is not empty ):
            suffix = pop suffix from Q
            send QUERY command to tags
                with frame number and suffix
            reply = receive reply from tags

                        if(reply is no response) :
                            # no tag is identified
            else if( reply is identified ):
                # a tag is identified
            else if( reply is collision ):
                append ('0' || suffix) to Q
                append ('1' || suffix) to Q
                collision number += 1
            end if

            # First Frame Test
            if( frame number is 0 and
            collision number > collision threshold
        ):
                epoch size *= 2
                has new epoch size = True
                break
            end if
        end while

        if( has new epoch size is True ):
            break
        end if
        frame number += 1
    end while
end while
```

표 4 FQTR의 의사코드(태그)

```
epoch size = receive epoch size from reader
my frame number = randomly select in epoch size
frame number, suffix = receive suffix and the frame
        number from reader

if( frame number is my frame number ):
    if( ID ends with suffix ):
        return ID
    end if
end if
```

표 5 RFID 물류 시스템에서 GID-96 기반 태그 ID 구성

	헤더	회사	제품군	시리얼넘버	활용
①	00110101	동일 (랜덤)	동일 (랜덤)	1000 (순차)	원자재부품, 입하/공장
②	00110101	동일 (랜덤)	3 (랜덤)	200 (순차)	공장 출하, 도매/총판 입하
③	00110101	3 (랜덤)	30 (랜덤)	100 (순차)	도매/총판 출하, 소매 입하
④	00110101	10 (랜덤)	15 (랜덤)	20 (랜덤)	소매 판매

소비자로 향하는 것으로 가정하고, ①~④번은 각각의 지점에서 입하시 인식하여야 하는 태그 구성이다. ①번은 공장에서 사용되는 태그 ID 구성을 나타낸다. 헤더는 GID-96 코드임을 인식하기 위하여 동일하게 '00110101'로 구성하였고, 회사는 모든 태그가 동일하다. 제품군을 나타내는 비트 역시 동일하지만, 시리얼 넘버는 순차적으로 구성하여 100개와 1000개의 태그 ID를 생성하였다.

같은 구성으로 ②번은 도매에서 이용되는 태그 ID, ③번은 소매에서 이용되는 태그 ID, ④번은 소비자가 이용하는 태그 ID를 가정하여 각각 100개와 1000개의 태그 ID를 생성하였다.

그리고 시뮬레이션의 신뢰도를 높이기 위하여 같은 태그로 충돌 방지 프로토콜을 변경해가며 100회의 시뮬레이션을 수행하였다. 왜냐하면 알로하 기반 프로토콜들은 난수 생성기를 통해 프로토콜을 진행하기 때문이다.

4.1.2 시뮬레이션 비교 항목

태그 충돌 방지 알고리즘의 목적은 빠른 시간 안에 리더의 인식 범위 안에 있는 모든 태그를 인식하는 것이다. 따라서, 태그 충돌 방지 알고리즘의 성능을 비교하기 위해서는 첫째로, 리더의 인식 범위 안의 모든 태그를 인식했는가를 살펴보아야 한다. 이는 태그 충돌 방지 알고리즘의 기본 조건이 된다.

다음으로, 리더와 태그 사이에 주고받는 전송 비트 수를 적게 하는 것이 중요하다. 태그 충돌 방지 알고리즘은 ISO/IEC 18000-6 이 표준인 물류/유통 환경에서 사용된다[10]. 이 표준은 패시브 태그를 사용하여 태그의 자원관리가 중요하다. 따라서, 우수한 태그 충돌 방지 알고리즘은 태그의 송신, 수신 비트를 줄이는 것을 고려하여야 한다.

마지막으로, 태그 충돌 방지 알고리즘들이 얼마나 빠르게 동작하는가를 비교하여야 한다. 그러나, 각 알고리즘들이 서로 다른 동작 방법과 표준으로 구성되어, 비교 기준이 모호하다. 절대적인 시간 단위는 표준마다, 나라마다 다르기 때문에 비교하기가 어렵다. 따라서 상대적인 기준이 필요하다.

그림 6은 QT, DFSA, FQTR의 동작을 시간의 흐름에 따라 선형적으로 나타내었다. (a)는 리더가 태그에게 프리픽스를 보내고, 응답을 기다리는 과정을 반복하는 쿼리 트리 알고리즘을 나타내었다. (b)는 리더가 프레임의 시작을 알리면(FS) 태그는 프레임 내에서 응답할 슬

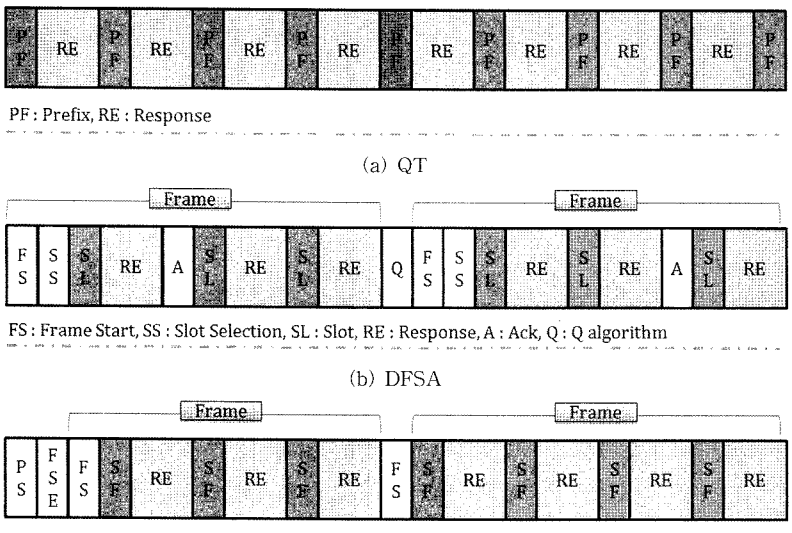


그림 6 쿼리트리(QT), 다이나믹 프레임드 슬롯티드 알로하(DFSA), 프레임드 쿼리 트리 리버스드(FQTR) 프로토콜 동작의 시간 흐름

롯을 선택하고(SS), 리더가 슬롯을 외칠 때(SS) 응답하는(RE) 다이나믹 프레임드 슬롯티드 알로하의 알고리즘을 나타내었다. (c)는 본 논문에서 제안하는 프레임드 쿼리 트리 리버스드 알고리즘으로 리더가 프로토콜의 시작을 알리면(PS) 태그들은 응답할 프레임을 무작위로 선택하게 된다(FSE). 그 후, 매 프레임마다 리더가 프레임 시작을 알리고(FS) 서픽스를 보내면(SF) 태그가 응답하는(RE) 알고리즘의 흐름을 보여준다.

그림 6을 보면 각 알고리즘은 다른 방식으로 동작하지만, 모두 (리더 요청 - 태그 응답) 과정이 전체 알고리즘의 대부분을 차지하는 것을 확인할 수 있다. 좀 더 쉽게 확인할 수 있도록 (리더 질의 - 태그 응답)에 해당하는 과정은 배경색으로 표현하였다. (b), (c) 프로토콜의 실제 동작에서는 1프레임 내부의 (리더 질의 - 태그 응답) 횟수가 훨씬 많이 존재하게 되고, 상대적으로 다른 동작들(흰색으로 표기된 과정들)은 전체 동작 과정에서 차지하는 비중이 미비해지게 된다.

따라서, 태그 충돌 방지 프로토콜의 성능 비교는 (리더 질의 - 태그 응답)횟수로서 비교할 수 있다.

정리하자면, 다음의 3가지를 비교하도록 한다.

① 리더의 인식범위 안에 있는 태그의 인식률

② 리더와 태그간에 주고받은 비트 수

③ 리더의 질의와 태그의 응답을 기다리는 횟수

4.1.3 시뮬레이션 시스템

시뮬레이션 프로그램은 윈도우즈(Windows) XP 환경에서 파이썬(Python) 2.3을 사용하였다.

먼저 시뮬레이션에 사용할 태그 ID 집합을 TagGenerator를 통해 생성한다. TagGenerator 클래스는 앞절에서 설명하였듯이, 실제 RFID 환경과 가장 유사한 상황을 고려하여 생성한다. 생성된 태그 ID들은 Simulator 클래스에서 Type B(바이너리 트리), Type C(다이나믹 프레임드 슬롯티드 알로하), 쿼리 트리, 프레임드 쿼리 트리 리버스드 프로토콜을 각각 100회씩 실행하고 결과를 출력한다.

전체 시뮬레이션 프로그램의 구조는 그림 7과 같다[8].

먼저 실제 시뮬레이션에 들어가기 전에 Tag 클래스는 TagGenerator 클래스가 만든 Tags 집합의 고유 식별 ID를 가지도록 한다. 그리고 Air 클래스는 이런 전체 Tag 클래스들을 가지고 있게 하고 Reader 클래스는 Air 클래스를 가지게 한다. 결국 Reader 클래스는 Tag 클래스와 직접 메시지를 송수신하는 게 아니라 Air 클래스를 통해서만 Tag와 질의를 보내거나 응답을 받을 수 있다. 이렇게 Air 클래스가 Reader 클래스와 Tag 클래스 사이에 들어가는 이유는 Reader 클래스가 Tag 클래스들을 전혀 모르게 하기 위해서다. 시뮬레이션이 아닌 실제 환경에서도 리더기가 태그가 몇 개인지도 모르고 충돌 방지 프로토콜을 하기 때문에 이와 유사하게 한 것이다.

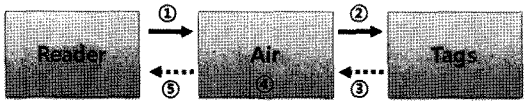


그림 7 태그 충돌 방지 알고리즘 시뮬레이션 프로그램의 구조

시뮬레이션에 들어가면 Reader 클래스는 자신의 프로토콜을 수행하면서 태그에게 질의를 하고 싶으면 Air 클래스에게 질의한다. 그러면 Air 클래스는 Reader에게 받은 질의 그대로를 자신이 가지고 있는 모든 Tag 클래스에게 똑같이 질의 한다. Tag 클래스들은 각각 자신이 응답할 것이 있으면 유효한 응답을 하고 응답할 것이 없으면 무효한 응답을 한다. Air 클래스는 이 Tag 클래스들의 응답 중 유효한 응답들만 모으고 이를 다시 Reader 클래스에게 응답한다. 이러한 일련의 과정은 리더기와 태그 사이의 한 번의 질의-응답으로 간주된다.

본 시뮬레이션 프로그램에서는 전파 방해로 인한 오류는 없다고 가정한다.

4.2 시뮬레이션 결과 및 분석

4.2.1 태그 인식률(Recognition Ratio)

표 6과 표 7은 각각 태그가 100개, 100개 일 때의 시뮬레이션을 통한 태그 인식률 결과이다. 표 안의 숫자는 프로토콜 과정에서 최종적으로 몇 개의 태그가 인식되었는지를 나타낸다. 소수점이 보이는 것은 여러회 반복한 결과값을 평균으로 구하였기 때문이다.

표 6과 표 7 공통적으로 다른 프로토콜들은 모든 태그를 인식하나, Type C 프로토콜은 모든 태그를 인식하지 못하고 종료한다. 더욱이 태그 수가 많은 환경에서는 거의 매번 모든 태그를 인식하지 못하였다. 이는 프레임드 쿼리 트리 알로하 프로토콜의 한계로 랜덤으로 슬롯을 선택하므로, 태그들이 지속적으로 같은 슬롯을 선택하여 모든 태그를 인식하지 못하는 결과를 가져온 것이다.

본 논문에서 제안하는 프레임드 쿼리 트리 리버스드 프로토콜은 모든 실험 결과에서 리더 범위의 모든 태그를 인식하여 태그 충돌 방지 알고리즘의 기본 요건에 적합함을 보였다

표 6 인식한 태그 수(총 태그 ID : 100개)

	공장	도매	소매	소비자
Type B	100	100	100	100
Type C	99.3	100	100	100
QT	100	100	100	100
FQTR	100	100	100	100

표 7 인식한 태그 수(총 태그 ID : 1000개)

	공장	도매	소매	소비자
Type B	1000	1000	1000	1000
Type C	990	993	993	995
QT	1000	1000	1000	1000
FQTR	1000	1000	1000	1000

4.2.2 태그 송수신 비트(Transmitted Bits)

태그가 낮은 송수신 비트를 가지는 것은 저가의 태그 환경에 유리하다. 그림 8에서 볼 수 있듯이 프레임드 쿼리 트리 리버스드 프로토콜은 Type C와 함께 가장 낮은 수준의 태그 송수신 비트를 가진다. Type B 프로토콜은 이보다 조금 많은 비트를 주고받지만, 쿼리 트리 프로토콜의 경우 매우 많은 비트를 송수신하게 된다.

시뮬레이션에 사용된 태그 ID는 실제 RFID 환경에 적용될 GID-96을 기준으로 생성되었기 때문에, 태그 ID들의 프리픽스가 많은 부분 유사하게 구성되어 있다. 그러므로, 프리픽스부터 트리를 따라 내려가는 쿼리 트리 프로토콜은 가장 많은 송수신 비트를 가질 수 밖에 없다. 이에 비하여 프레임드 쿼리 트리 리버스드 프로토콜은 태그를 각 프레임으로 나누었기 때문에 트리의 높이가 작아져 전송해야할 비트 수가 적다. Type C는 프리픽스 혹은 서픽스를 전송하지 않으므로, 가장 작은 송수신 비트를 가진다.

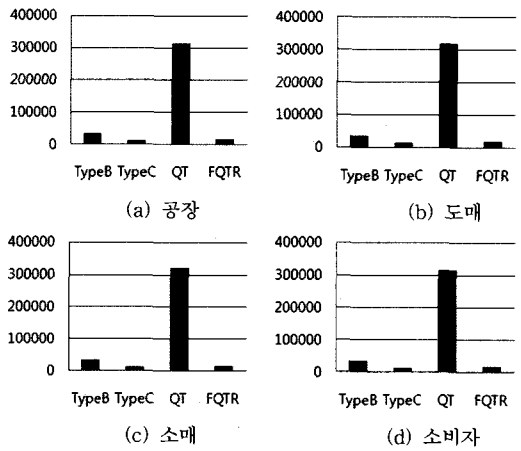


그림 8 태그 송수신 비트 (총 태그 ID : 100개)

그림 9에서 Type B 프로토콜의 성능이 낮아진 것처럼 보이나, Type C 프로토콜과 프레임드 쿼리 트리 리버스드 프로토콜의 2배 수준으로 그림 8과 성능면에서 동일한 결과를 보인다. 그러나, 쿼리 트리 프로토콜이 태그 ID 100개 환경에서 너무 많은 비트를 전송했기 때문에 상대적으로 적게 보이는 것이다.

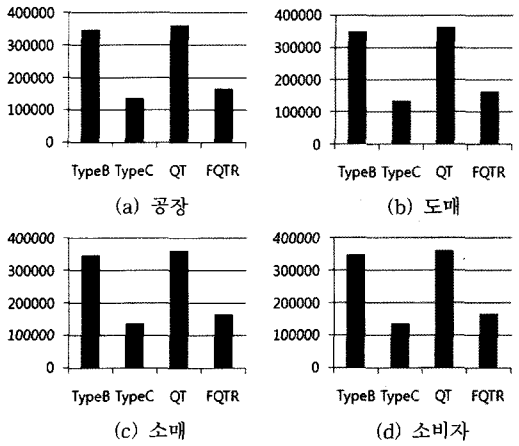


그림 9 태그 송수신 비트(총 태그 ID : 1000개)

4.2.3 리더 질의 - 태그 응답(Request-Response)

태그 충돌 방지 프로토콜의 성능을 결정하는 리더 질의 - 태그 응답 횟수의 시뮬레이션 결과는 표 8과 표 9에서 확인할 수 있다.

태그 ID가 100개일 경우는 프레임드 쿼리 트리 리버스드 프로토콜이 가장 우수한 성능을 보인다. Type B 프로토콜이 약소한 차이로 뒤를 따르며, Type C 프로토콜은 약 1.5배정도 낮은 성능을 보이는데 반해, 쿼리 트리 프로토콜은 뒤떨어지는 성능을 보인다.

태그 ID가 1000개일 경우는 쿼리 트리 프로토콜이 가장 우수한 성능을 보이고, 프레임드 쿼리 트리 리버스드

표 8 리더 질의 - 태그 응답 횟수(총 태그 ID : 100개)

	공장	도매	소매	소비자
Type B	288	288	300	291
Type C	424.7	423.7	428.3	416.3
QT	2836	2886	2916	2860
FQTR	281	276	260	265

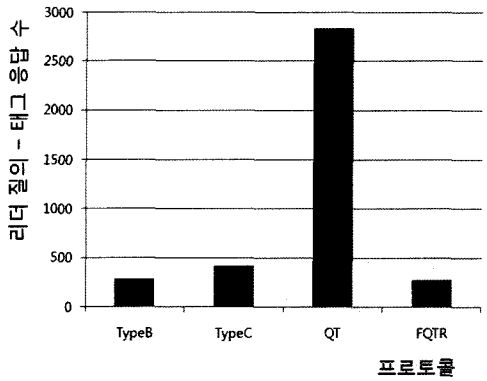


그림 10 리더 질의-태그 응답 횟수(총 태그 ID : 100개)

표 9 리더 질의 - 태그 응답 횟수(총 태그 ID : 1000개)

	공장	도매	소매	소비자
Type B	2873	2892	2886	2901
Type C	3649.3	4069	3921	4074.3
QT	2158	2158	2158	2158
FQTR	2454	2517	2415	2453

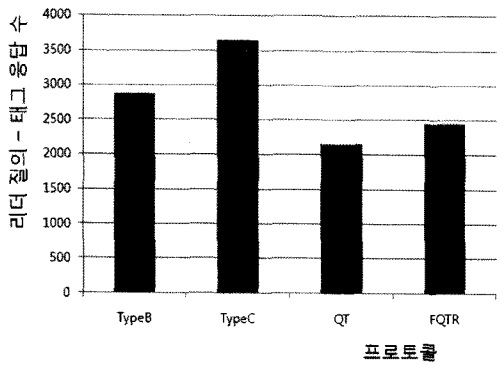


그림 11 리더 질의 - 태그 응답 횟수(총 태그 ID : 1000개)

프로토콜이 10% 정도의 차이로 뒤를 잇는다. 다음으로 Type B 프로토콜과 Type C 프로토콜 순의 성능을 보인다.

각각의 경우에 프레임드 쿼리 트리 리버스드 프로토콜과 쿼리 트리 프로토콜이 가장 우수한 성능을 보였으나, 쿼리 트리 프로토콜의 경우 태그 수에 크게 영향을 받은 결과를 보여주었다. 이에 비하여 프레임드 쿼리 트리 리버스드 프로토콜은 태그 수에 관계없이 꾸준히 우수한 성능을 보여주었다.

특히, 표준으로 사용되고 있는 Type B 프로토콜과 Type C 프로토콜에 비하여 10%~40% 향상된 성능으로 발전 가능성을 보여주었다. 더욱이 Type C 프로토콜은 모든 태그를 인식하지 못하는 문제를, Type B 프로토콜은 전송 비트가 많은 문제를, 쿼리 트리 프로토콜은 태그 수에 영향을 받는 문제를 가진데 반하여, 안정성에서 뛰어난 결과를 확인할 수 있다.

프레임드 쿼리 트리 리버스드 프로토콜의 리더 질의 - 태그 응답 횟수를 분석해 보면 태그 ID의 약 2.5배 수준인 것을 알 수 있다. 이는 하나의 프레임에서 인식하는 이상적인 태그 수가 2개인 점을 고려하여 설계한 부분이 반영된 것으로 볼 수 있다.

즉, 1개의 태그를 인식하는데 2.5 회의 리더 질의 - 태그 응답 횟수를 가진다.

5. 결 론

본 논문에서는 앞으로 발전 하게 될 유비쿼터스 사회에서 각광 받고 있는 RFID 기술에 대해 살펴보고 RFID 기술의 더 많은 사용을 위해 풀어야 할 태그 충돌 방지 프로토콜을 살펴보았다.

현재까지는 태그 충돌 방지 프로토콜을 알로하 기반과 트리 기반으로 크게 나눠왔지만 본 논문에서는 두 가지 부류의 특징을 결합한 새로운 프로토콜을 제안하였다. 이 프로토콜들은 프레임드 슬롯티드 알로하와 쿼리 트리 리버스드 프로토콜을 결합한 형태인 프레임드 쿼리 트리 리버스드 프로토콜로 태그들을 분산시켜 인식하는 것이 핵심이다.

본 논문에서는 새로이 제안하는 프로토콜을 시뮬레이션 프로그램을 통해 기존에 많이 사용하고 있는 프로토콜인, ISO/IEC 18000-6 Type B, Type C, 그리고 쿼리 트리 프로토콜과 비교 해 보았다.

본 논문에서 제안된 프레임드 쿼리 트리 리버스드 프로토콜은 기존의 충돌 방지 프로토콜에 비하여 우수한 성능을 보였다.

더욱이 Type C 프로토콜은 모든 태그를 인식하지 못하는 문제를, Type B 프로토콜은 전송 비트가 많은 문제를, 쿼리 트리 프로토콜은 태그 수에 영향을 받는 문제를 가진데 반하여 제안된 프로토콜은 항상 유사한 결과를 보여주는 뛰어난 안전성을 보여주었다.

또한, 리더와 태그간의 전송 비트 수에서도 낮은 이용률을 보여주는 것으로, 저가의 태그에 적합한 프로토콜임을 증명하였다.

앞으로 프레임드 쿼리 트리 리버스드 프로토콜에 태그 수를 추정하는 알고리즘과 함께 프레임 수를 동적으로 변경할 수 있도록 한다면, 더 나은 성능 향상이 있을 것으로 기대된다.

참 고 문 헌

[1] K. Finkenzeller, "RFID handbook," John wiley & Sons, 1999.

[2] 이수련, 이채우, "RFID 시스템의 다중 인식 기술 현황", *한국전자파학회지*, vol.15, no.2, 2004. 4.

[3] EPCglobal, "Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960MHz, Version 1.0.9," 2005.

[4] Jae-Ryong Cha and Jae-Hyun Kim, "Dynamic Framed Slotted ALOHA Algorithms using Fast Tag Estimation Method for RFID System," Consumer Communications and Networking Conference, IEEE, January 2006.

[5] Ching Law, Kayi Lee and Kai-Yeung Siu, "Efficient Memoryless Protocol for Tag Identification," In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, ACM, August 2000.

[6] Jihoon Myung, Wonjun Lee, "Adaptive splitting protocols for RFID tag collision arbitration," *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, 202-213, 2006.
[7] Jung-Sik Cho, Jae-Dong Shin, Sung Kwon Kim, "RFID Tag Anti-Collision Protocol: Query Tree with Reversed IDs," *ICACT 2008*, vol.1, pp.225-230, 2008.
[8] J Shin, S Yeo, T Kim, SK Kim, "Hybrid Tag Anti-collision Algorithms in RFID Systems," *Lecture Notes in Computer Science*, vol.4490/2007, 693-700, 2007.
[9] D. Brock, "The Electronic Product Code - A Naming Scheme for Physical Objects," Auto-ID White Paper, January 2001. http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-002.pdf
[10] International Organization for Standardization, "ISO/IEC 18000-6:2004/Amd 1:2006," June 2006.

정 승 민

2006년 2월 중앙대학교 컴퓨터공학과 졸업(학사). 2009년 2월 중앙대학교 컴퓨터공학과 졸업(석사). 2009년 3월~현재 ㈜엔텔스 기술연구소 연구원. 관심분야는 RFID, 암호응용 및 정보보호

조 정 식

2003년 2월 강남대학교 전자계산학과 졸업(학사). 2005년 2월 중앙대학교 컴퓨터공학과 졸업(석사). 2005년 3월~현재 중앙대학교 컴퓨터공학과 박사과정. 관심분야는 암호응용 및 정보보호, RFID 보안

김 성 권

1981년 2월 서울대학교 계산통계학과 졸업(학사). 1983년 2월 한국과학기술원 전산학과 졸업(석사). 1983년~1985년 목포대학교 자연과학대학 전산통계학과 전임강사. 1990년 8월 미국 University of Washington Computer Science & Engineering(박사). 1991년 3월~1996년 2월 경성대학교 이과대학 전산통계학과 조교수. 1996년 3월~현재 중앙대학교 컴퓨터공학부 정교수. 관심분야는 생물정보학, 암호응용 및 정보보호, 계산기하학 및 응용