RFID 시스템에서 비트변화감지를 이용한 하이브리드 충돌 방지 프로토콜

A Hybrid Anti-Collision Protocol using Bit Change Sensing Unit in RFID System

김 정 환* Jeong-Hwan Kim　김 영 태** Young-Tae Kim　박 용 수*** Yong-Soo Park　안 광 선**** Kwang-Seon Ahn

요 약

RFID 시스템에서 리더의 식별영역 내에 다수개의 태그가 존재할 경우 태그 충돌 문제가 발생할 수 있으며, 따라서 태그 인식에 많은 시간이 필요하다. 태그 충돌 문제는 RFID 시스템 설계 시 가장 중요한 핵심 이슈중의 하나이며 다양한 프로토콜이 제안되고 있다. 일반적인 트리 기반의 프로토콜들은 적합한 프리픽스를 생성하여 태그 인식을 빠르게 하는 것이 목적이다. 본 논문에서는 리더와 태그의 질의-응답 횟수를 줄일 수 있는 QT-BCS 프로토콜을 제안한다. QT-BCS 프로토콜에서는 타임 슬롯과 비트 변화 감지 유닛을 통하여 프리픽스를 생성한다. 식별영역내의 태그들은 이전 비트와 다른 값이 나올 때까지의 비트값을 리더에 전송하도록 설계되고, 리더는 0-슬롯과 1-슬롯에 첫째 비트 값에 따라 비트값을 저장한다. 이와 같은 방법은 질의 프리픽스를 쉽게 추적하므로 질의 횟수를 효과적으로 줄인다. 시뮬레이션 결과 QT-BCS 프로토콜은 Query Tree, 4-ary Query Tree 프로토콜 보다 질의-응답 횟수를 줄여 태그 인식 속도를 빠르게 개선시켰다.

Abstract

A tag collision problem occurs when many tags are placed in a interrogation zone in RFID system. A tag collision problem is one of core issues and various protocols have been proposed to solve the collision problems. Generally tree-based protocols generate unique prefixes and identify tags with them as quick as possible. In this paper, we propose the QT-BCS protocol which decreases the identification time by reducing the number of query-response. The QT-BCS protocol makes a prefixes using time slot and bit change sensing unit. This protocol compares the current bit of tags until the current bit is differ from the previous one. When this occurs, all of the bits scanned so far are transferred to slot-0 and slot-1 depending on the first bit value in Reader. Consequently, this method can reduce the number of queries by tracing prefixes easily. Simulation result shows QT-BCS is more efficient in identifying tags than Query Tree and 4-ary Query Tree protocol.

☞ Keywords : 수동형 RFID, 쿼리트리 프로토콜, 태그 충돌 프로토콜, 비트감지변화유닛, 프리픽스

1. 서 론

RFID(Radio Frequency IDentification) 시스템은 유비쿼터스 컴퓨팅 환경을 지원하기 위한 차세대 인식기술로 인식되고 있다. RFID는 자동 인식 기술이며, 무선 주파수를 통하여 데이터를 저장하고 검색한다. 최근, RFID 기술은 유통 및 물류관리뿐만 아니라 보안, 소방/방재, 환경관리, 정보유통 등 다양한 응용분야에 적용되고 있다 [1].

RFID 시스템은 리더(Reader)와 태그(Tag)로 구성된다. 리더는 태그에게 정보를 요청하고, 전송 받은 정보를 분석하는 역할을 수행한다. 기존의 환경에서는 대부분 고정형 리더가 사용되었으나, 최근에는 휴대형으로 변환되고 있으며 휴대폰에 리더기

* 정 회 원 : 계명문화대학교 컴퓨터학부 교수
kjh@kmcc.ac.kr
** 준 회 원 : 경북대학교 전자전기컴퓨터학부 석사과정
taenystar@knu.ac.kr
*** 정 회 원 : 경북대학교 BK21 Post-Doc
timpark75@knu.ac.kr
**** 정 회 원 : 경북대학교 컴퓨터공학과 교수
gsahn@knu.ac.kr
[2008/08/11 투고 - 2008/09/04 심사 - 2008/11/12 심사완료]

를 내장하려는 RFID 서비스가 시도되고 있다.

태그는 전원의 공급 유무에 따라 능동형(Active)과 수동형(Passive)으로 분류된다. 능동형 태그는 자체 전원을 가지므로 식별 범위가 넓은 장점을 가지지만, 생존기간(Life Time)이 제한적이며 수동형 태그에 비해 가격이 높고, 크기가 크다는 단점을 가진다. 수동형 태그는 자체 전원이 없기 때문에 식별거리에 제약이 있지만 생산가격이 저렴하여 가장 많이 사용되고 있다.

리더는 태그의 정보를 수집하기 위하여 모든 태그들에게 질의를 보낸다. 하나의 리더의 식별영역 내에 다수개의 태그가 존재할 경우, 동시에 여러 개의 태그가 응답하여 어떠한 태그도 식별 할 수 없는 상태가 발생한다. 이러한 상태를 태그 충돌 문제(Tag Collision Problem)라고 하며, 태그의 고속 식별의 저해 원인이 된다. RFID 시스템에서 가장 핵심이 되는 기술은 더 빠른 시간에 모든 태그를 식별 할 수 있는 충돌 방지 프로토콜을 개발하는 것이다 [6].

태그 충돌 문제를 해결하기 위한 프로토콜은 크게 트리 기반과 슬로 알로하 기반 두 그룹으로 분류된다. 트리 기반 프로토콜은 리더가 프리픽스(Prefix)를 태그에게 전달하고, 동일한 프리픽스를 갖는 태그는 그에 응답하는 과정으로 태그 ID를 식별하는 방법이다. 그리고 이 과정은 트리로 설명된다. 트리 방법은 구현이 쉽고, 태그의 높은 컴퓨팅 능력을 요구하지 않는다. 그러나 유사한 ID 값을 갖는 태그가 많은 경우 질의 횟수가 증가하고, 전송되는 비트의 양이 많아지는 단점이 있다. 대표적인 트리 기반 프로토콜은 쿼리 트리 [3], 4-ary 쿼리 트리 [4], QT-CBP [5] 프로토콜이 있다.

슬롯 알로하 기반의 프로토콜은 리더가 태그에 랜덤 시드(Seed) 값을 전달하고, 태그는 그 값을 기반으로 응답할 시간을 결정하는 슬롯 알로하 방식을 주로 사용한다. 이러한 방식은 하나의 슬롯에 하나의 태그만 응답하게 함으로써 리더가 태그를 인식하는 방법이다. 그러나 이러한 방법은 확률을 이용하기 때문에 식별 영역내의 모든 태그를 인식하지 못할 수 있으며, 인식하는데 걸리는 시간을 예측하기 어려운 단점이 있다 [2][6][7][8].

본 논문에서는 저비용 수동형 RFID 시스템에서 인식속도와 인식률을 개선하기 위해 새로운 프로토콜을 제안한다. 제안된 프로토콜은 리더가 태그에게 랜덤 시드 [2][6][7][8]를 전달하는 대신, 태그가 가진 자신의 비트 값에 따라 전송 슬롯을 자동적으로 결정할 수 있다. 또한 이전 비트와 다른 값을 갖는 비트까지 전송하기 때문에 적은 질의 비트 횟수와 값을 갖는다. 시뮬레이션을 통한 검증 결과 제안 프로토콜은 첫 태그 인식률, 질의-응답 횟수, 전송비트 수에서 쿼리 트리, 4-ary Query Tree 프로토콜보다 높은 성능을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 Anti-Collision 프로토콜을 소개하고, 3장에서는 제안 프로토콜을 설명한다. 4장에서 기존 프로토콜과 본 논문에서 제안하는 프로토콜의 성능을 시뮬레이션 결과와 비교 및 분석하고, 마지막으로 5장에서 향후 과제 및 결론을 맺는다.

2. 관련연구

RFID 시스템에서 리더는 인식 범위내의 모든 태그들을 식별 할 수 있어야 한다. 태그는 단순히 리더의 명령어에 응답하는 능력만 갖고 있기 때문에 충돌 발생 이후 과정은 리더에서 처리한다. 우리는 리더의 질의에 대한 태그 응답을 3가지 Node로 구분하였다.

- No-Response Node – 태그의 응답이 없는 경우. 태그 식별에 시간지연을 가져온다.
- Collision Node – 2개 이상의 태그가 동시에 응답한 경우. 리더는 태그를 인식할 수 없다.
- Success Node – 하나의 태그만 응답한 경우. 리더는 하나의 태그 ID를 식별할 수 있다.

2.1 Slotted Aloha Protocol

슬롯 알로하(Slotted Aloha) 프로토콜 [6]은 리더가 태그들의 응답 시간을 고정된 몇 개의 타임 슬

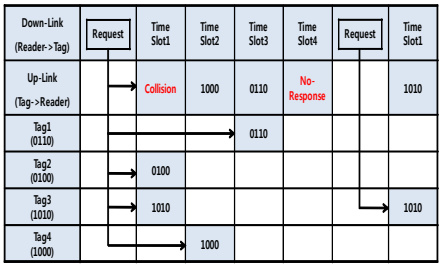
롯(Slot)으로 나눈 후 전송하며, 태그들은 전송 받은 슬롯을 선택하고 태그 ID를 리더에게 전송한 후 태그들을 식별하는 방식이다.

리더가 태그에게 요청 메시지를 전송할 때 슬롯에 대한 정보를 함께 전송하며, 태그들은 전송 받은 슬롯 정보를 이용하여 자신이 사용할 슬롯을 랜덤하게 선택한다 [2][7][8]. 이 때 각 타임 슬롯에 두 개 이상의 태그 ID가 응답하면 충돌이 발생한다. 이후 충돌이 발생한 태그는 리더의 요청메시지를 받고 자신이 사용할 슬롯을 선택한 후 태그 ID를 전송한다. 이러한 과정은 리더가 모든 태그들을 식별할 때까지 반복된다.

슬롯 알로하 프로토콜은 수학적인 확률(Probability)에 기반을 두고 있어 태그 수, 슬롯 수, 종료 시점을 정확히 파악하기 어려운 문제점이 있다.

그림 1은 슬롯 알로하 프로토콜을 이용하여 리더가 4개의 태그 ID를 식별하는 과정이다.

먼저 리더는 태그들에게 요청 메시지를 브로드캐스트(Broadcasting) 하면서, 동시에 각 태그들에게 슬롯 선택에 대한 정보를 전송한다. 각 태그들은 전송된 정보를 이용하여 자신의 슬롯을 랜덤하게 결정한다 [7]. 태그 2와 태그 3은 슬롯 1, 태그 1은 슬롯 3, 태그 4는 슬롯 2에서 응답한다. 슬롯 2와 슬롯 3은 하나의 태그가 응답하여 태그를 인식하였고, 슬롯 4는 응답한 태그가 없기 때문에 No-Response 슬롯이 된다. 반면 슬롯 1은 태그 2와 태그 3이 동시에 응답하여 충돌이 발생한다. 충돌이 발생한 태그 2와 태그 3은 다음 요청메시지 이후 다시 랜덤하게 슬롯을 선택하게 된다. 이러한 과정은 모든 태그가 식별 될 때까지 반복한다.



(그림 1) 슬롯 알로하 프로토콜의 동작과정

2.2 Query Tree Protocol

쿼리 트리 프로토콜 [3]은 대표적인 트리 기반 메모리래스(Memoryless)형 프로토콜이다. 여기서 메모리래스는 충돌 방지 프로토콜 동작 과정에 태그 ID 이외의 메모리는 필요하지 않다는 것을 의미한다.

쿼리 트리 프로토콜의 태그 식별 과정은 아래와 같다.

먼저 리더는 k-bit 길이의 프리픽스 B(B = b1,b2, …, bk)를 모든 태그에게 질의한다. 질의를 받은 태그들은 자신의 태그 ID와 프리픽스를 비교하여 값이 동일하면 자신의 태그 ID(TagIDk+1, Length)를 리더로 전송한다. 여기서 Length는 태그 ID의 길이를 나타낸다. 이때 태그의 응답결과에 따라 리더는 세 가지 노드로 응답한다. 먼저 태그로부터 아무런 응답이 없는 경우 No-Response Node, 이 경우 큐에서 새로운 질의 프리픽스 B'를 가져와 다시 태그에게 질의한다. 리더가 오직 하나의 태그만 응답하였을 경우 Success Node, 리더는 하나의 태그를 식별한다. 다수의 태그가 동시에 응답했을 경우 Collision Node, 충돌이 발생하여 리더는 기존 질의 프리픽스 B'에 1-bit 늘어난 0 과 1을 추가하여 새로운 질의 프리픽스(B'0, B'1)를 만든다. 새로 만들어진 질의 프리픽스는 큐에 저장되며 다음 질의에 사용 된다. 질의-응답 과정은 영역내의 모든 태그가 식별될 때까지 반복한다.

쿼리 트리 프로토콜은 동작 방식이 간단하여 구현하기 쉬운 장점을 가지고 있다. 그러나 전체 Tree Depth가 깊어져 태그를 식별하는데 많은 질의-응답 횟수와 비트수가 필요하며, 많은 Collision Node가 발생하는 문제점이 있다.

2.3 4-ary Query Tree Protocol

쿼리 트리 프로토콜은 충돌 발생시 기존 질의 프리픽스 B 끝에 0과 1을 붙어서 새로운 질의 프리픽스 B'를 만들어 사용한다. 즉 질의 프리픽스를 1-bit 확장한다. 4-ary Query Tree 프로토콜[4]은

쿼리 트리 프로토콜의 1-bit 질의 프리픽스를 2-bit로 확장하는 것이다. 즉, 새로운 질의 프리픽스 B'는 B'00, B'01, B'10, B'11이 된다.

이 경우 Collision Node는 쿼리 트리 프로토콜보다 줄어드는 대신에 No-Response Node가 늘어나는 문제점 있다.

표 1은 4-ary Query Tree 프로토콜의 동작을 보여준다. Step 1, 5는 충돌이 발생하여 태그를 식별할 수 없고 Step 2, 3, 8, 9는 하나의 태그가 응답하여 태그를 식별할 수 있다. Step 4, 6, 7과 같이 3번의 No-Response Node가 발생한다.

(표 1) 4-ary QT 프로토콜의 동작과정

Step	1	2	3	4	5	6	7	8	9
Reader	ε	00	01	10	11	1100	1101	1110	1111
Response	Collision	0010	0110	-	Collision	-	-	1110	1111
Tag1 (0010)	0010	0010							
Tag2 (0110)	0110		0110						
Tag3 (1110)	1110				1110			1110	
Tag4 (1111)	1111				1111				1111
Queue={ε}	00 01 10 11	01 10 11	10 11	11	1100 1101 1110 1111	1101 1110 1111	1110 1111	1111	ε

3. Hybrid Anti-Collision Protocol

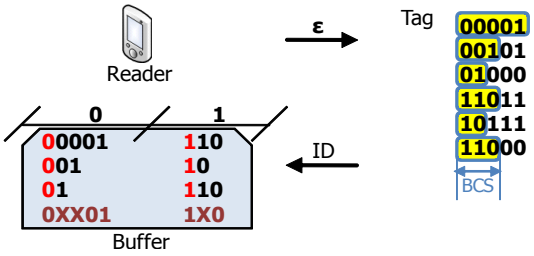
기존의 제안된 프로토콜은 리더의 인식영역 내에 태그의 수에 비례하여 충돌이 증가한다. 그러므로 다수의 태그 인식에 많은 시간을 소모하게 된다. 본 논문에서는 BCS 유닛과 기존의 QT 기법을 혼합하여 충돌을 최소화하고, 인식 시간 및 최초 태그 인식시간을 줄이는 것을 목적으로 한다. 더불어 최초의 태그를 인식하는데 걸리는 시간을 줄인다. 제안된 프로토콜의 핵심 알고리즘은 태그 간의 충돌이 발생하더라도 인지가 가능한 고유한 프리픽스에 관해 개발하는 것이다.

3.1 Bit Change Sensing Unit

트리에 기반한 인식 프로토콜은 유일한 프리픽스를 생성하고, 질의함으로써 태그를 인식하는 것이다. 이 과정에서 프로토콜은 다수의 태그에서 태그의 고유 프리픽스를 찾는 것이 목적이다. 태그 ID 값은 넓게 분포되어 있고, 태그가 자신의 ID를 리더에게 전송할 경우 충돌이 발생한다. 그러므로 빠르게 단일 비트 스트링을 찾는 것이 매우 중요하다.

이 문제를 해결하기 위해 본 논문에서는 태그의 비트 변화 감지(Bit Change Sensing) 유닛을 소개한다. BCS 유닛은 어떠한 비트 값이 이전 비트 값과 다른 경우를 판독하는 기능을 한다. 예를 들어, 태그 ID가 "000101"일 경우 태그의 BCS 유닛은 "0001"과 "01"로 나눌 수 있다. 태그는 ID 전송은 이진 슬롯 쿼리 트리 프로토콜 [9]과 같이 타임 슬롯에 의해 정해지며, 오직 첫 비트의 값에 의해 결정된다. 예를 들어, 태그 ID가 "0001"인 경우 첫 타임 슬롯에 [2][6][7][8] 전송이 이루어 진다. 이 규칙의 장점은 태그 ID 전송 시 마지막 비트에서 충돌이 발생 할 경우, 리더는 프리픽스를 쉽게 추적할 수 있다.

그림 2와 같이 리더가 프리픽스 "ε"을 질의하면 태그들은 자신의 ID 중에서 "00001"과 "001", "01"을 전달한다. 이후 리더는 맨체스터 코드 [5][6]를 이용하여 버퍼 B에 "0xx01"이 존재한다는 것을 인지한다. 버퍼 B는 $b_1, b_2, \cdots, b_k$ $(2 \le k \le T_{len})$ 로 나타내며, Tlen는 태그 ID의 길이를 의미한다.



(그림 2) 프리픽스 검출 과정

버퍼 B에서 프리픽스를 얻기 위해 ExtractPrefix()를 사용하며 x 비트는 1로 변환될 수 있으며 이후 큐에 새로운 프리픽스가 저장된다. 함수는 최하위 비트(LSB)에 도달할 때까지 순차적으로 반복한다. 그래서 “01”, “001”, “00001”이 큐에 저장된다. 충돌이 발생하지 않고 타임 슬롯이 비어 있다면 리더는 프리픽스를 인식한다. 비트가 1로 시작하는 태그 인식과정도 위와 동일하게 진행된다. ExtractPrefix() 함수는 슈도코드를 사용하여 설명하였다. 그림 5는 ExtractPrefix() 함수의 슈도코드를 보여준다.

```
tag T: length T_len , reader receved ID length k
buffer valued B, B = b1,b2, …, bk (2 ≤ k ≤
T_len ) buffer B length B_len B_len , A new prefix
P is saved in the Queue Q, Tag ID is M
1: ExtractPrefix (B)
2: {
3:     For ( c = 0 to B' range )
4:     {
5:       if (b_c  b_c == 'x' )
6:       {
7:         P = P + ( not b1 );
8:         if ( c == T_len T_len )  M ≤ P;
9:         else                     S ≤ P;
10:      }
11:    }
12: }
```

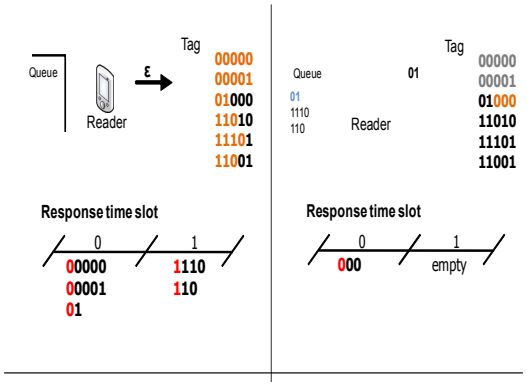
(그림 3) ExrtractPrefix() 함수의 슈도코드

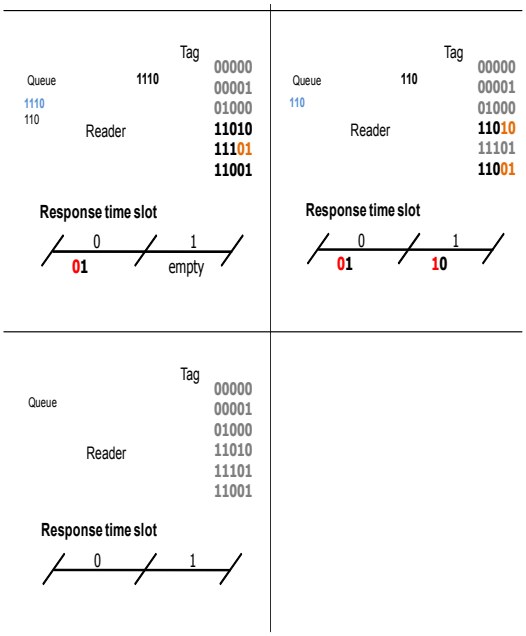
3.2 Example

제안된 프로토콜은 리더가 큐에서 선택한 프리픽스를 모든 태그에게 전파하면, 태그는 자신의 식별 ID와 비교하여 동일한 패턴을 가지고 있는 경우, 태그 ID에서 비트 값이 변화하는 부분까지를 분리하여 리더에게 전달한다. 리더는

프리픽스와 전달 받은 값에서 인식한 값을 접목하여 새로운 프리픽스를 큐에 저장하고, 다시 질의 하는 과정을 반복함으로써 태그 ID 트리를 생성한다.

그림 4은 6개의 랜덤한 태그 ID가 존재하는 상황이라고 가정하고 제안된 프로토콜을 실행한 예제이다. 먼저 리더가 태그에게 ID 전송을 요청한다. 태그는 ID 값에서 비트 값이 변화(BCS)하는 부분까지를 태그에게 전달한다. 이 과정에서 태그는 전달할 ID 비트들 중에서 최상위 비트(MSB) 값을 이용하여 응답할 시점을 결정한다. 태그들은 첫 응답 시점에서 충돌이 발생되고 버퍼에는 “0x00x”가 존재한다. 전달된 프리픽스는 비트 값이 변화하는 부분에서만 1의 값을 가지게 되므로 리더는 버퍼에 전달된 값이 “00000”, “00001”, “01”이 존재한다는 것을 알 수 있다. 인식된 프리픽스는 큐에 저장한다. 태그를 추출하는 과정에서 “00000”과 “00001”은 ID의 길이와 동일한 길이를 가지기 때문에 인식된 것이다. 이와 동일한 방법을 이용하여 전달시점 ‘1’에서도 프리픽스 “1110”과 “110”을 인식한다. 리더는 다시 큐에 저장된 프리픽스 중 “01”을 꺼내 전달하고, 태그는 동일한 프리픽스를 가지고 있는 태그들 중 리더가 전달한 값 이후부터 비트의 값이 변화하는 부분까지 다시 리더에게 전달한다. 이후 리더가 전달한 프리픽스와 전달받은 값을 조합하여 태그 ID를 인식한다. 인식과정은 두 슬롯이 모두 전달 받은 값이 없는 경우 종료하게 된다.





(그림 4) QT-BCS 프로토콜의 동작과정 예제

QT-BCS 프로토콜을 수행한 경우 QT 알고리즘에서 필요한 21번에 비해 현저히 줄어든 7으로 줄일 수 있음을 보여준다. 특히 제안된 방식은 태그 ID의 값만 전송하면 되기 때문에 전체 전달을 위해 사용되는 비트의 수도 줄일 수 있다.

4. 성능평가

논문에서는 제안하는 QT-BCS 프로토콜의 성능 평가를 위해 C#언어로 시뮬레이션 프로그램을 설계하여 기존에 제안되었던 QT, 4-ary QT 프로토콜에 대하여 리더와 태그의 평균 질의-응답 횟수를 비교하였다. 그리고 태그 ID의 길이는 EPCglobal에서 제안하는 EPC code [10]를 감안하여 96비트로 하였다. 그림 5는 EPC-96비트의 구조를 보여준다.

EPC™ TYPE	HEADER SIZE	FIRST BITS	DOMAIN MANAGER	OBJECT CLASS	SERIAL NUMBER	TOTAL
96 BIT	8	00	28	24	36	96

(그림 5) EPC-96bit 태그 구조

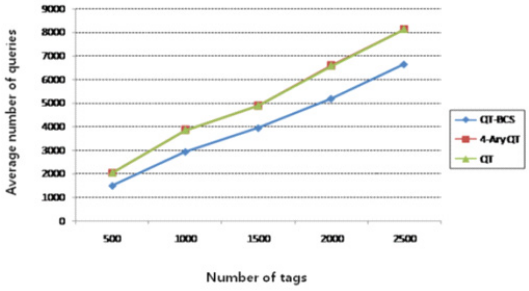
태그 ID의 생성은 96비트 전체가 랜덤, 순차적인 경우와 하위 36비트만 랜덤, 순차적인 경우를 고려하였다. 하위 36비트만 고려하는 이유는 일반적으로 사용되는 EPC 코드에서 데이터 유형 및 길이를 나타내는 헤더, 업체 코드를 나타내는 EPC 관리자, 상품 코드는 거의 변화가 없기 때문이다. 만약 동일한 회사에서 생산된 제품일 경우 위 필드 값은 동일하며 하위 36비트의 제품 번호만 다르게 된다.

본 논문에서는 이러한 상황을 고려하여 96비트 태그 ID 전체 랜덤, 순차적인 경우와 상위 60비트가 산업군에서 사용된다는 경우를 고려하여 5개의 군으로 나누고 하위 36비트 ID를 랜덤, 순차적인 경우에 대해 시뮬레이션 하였다. 그리고 태그 ID의 수는 500개부터 2500개까지 태그를 생성하여 비교하였다. 각 실험 결과는 5회 반복 실시한 결과의 평균값이다.

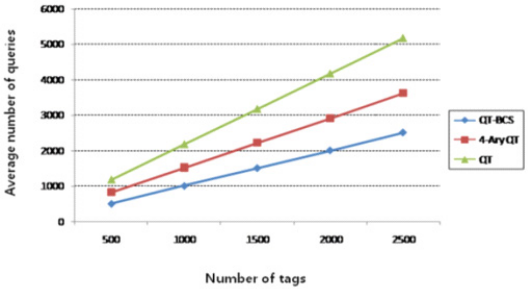
4.1 36bit Random ID, Sequential ID

그림 8은 36비트 태그 ID가 랜덤한 (a) 경우와 순차적인 경우(b), 모든 태그를 식별하기 위한 평균 질의-응답 횟수를 비교한 그래프이다. 그림 8에서 태그의 수가 500개 라는 것은 96비트 중 상위 60비트 값이 동일한 태그가 100개씩 5개군이 있다는 것을 의미한다. 그리고 제품 번호에 해당하는 하위 36비트 값은 랜덤, 순차적인 값을 가진다.

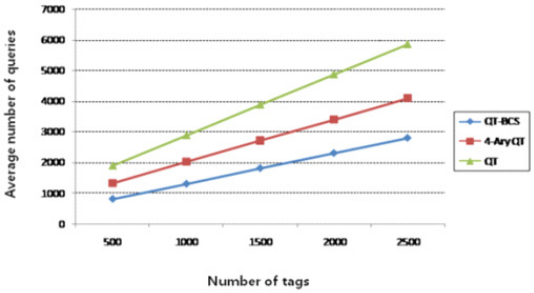
그림 6의 (a)에서 태그가 랜덤 ID일 경우 500개 태그를 인식할 때 QT와 4-ary QT는 2031번 소요되지만 QT-BCS는 1486번 소요된다. 그림 8의 (b)는 태그 ID가 순차적일 경우이다. 그림에서 QT-BCS는 QT보다 평균 2배 성능이 향상 되었는걸 알 수 있다. QT-BCS는 질의 횟수를 줄여 리더가 전체 태그 인식 시간을 단축할 수 있다.



(a) Random assignment



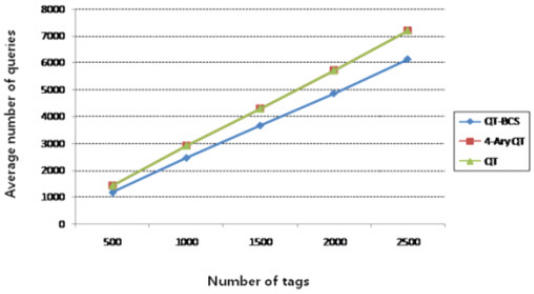
(b) Sequential assignment
(그림 7) 96bit tag ID 실험 결과



(b) Sequential assignment
(그림 6) 36bit tag ID 실험 결과

4.2 96bit Random ID, Sequential ID

그림 9는 96비트 모두를 고려한 실험 결과이다. 태그의 수가 2500개일 경우 QT-BCS는 QT보다 랜덤 ID는 평균 25.4%, 순차적 ID는 51.9%의 성능향상을 보여 주었다. QT-BCS는 태그의 수가 많고 태그 ID가 순차적일 경우 더 높은 성능을 보였다.



(a) Random assignment

5. 결 론

본 논문에서는 RFID 시스템에서 질의 시간을 최소화하는 QT-BCS 프로토콜을 제안하였다. QT-BCS 프로토콜은 기존에 제안되었던 프로토콜과 달리 이웃한 비트의 값(BCS)과 이진 슬롯을 이용하여 프리픽스 생성에 사용함으로써 질의-응답 횟수를 획기적으로 줄였다. QT-BCS 프로토콜의 성능을 측정하기 위하여 태그의 수, 태그 ID의 값이 랜덤한 경우와 순차적인 경우를 고려하여 시뮬레이션 하였다. 시뮬레이션을 통해 기존에 제안되었던 QT와 4-ary QT 프로토콜보다 본 논문에서 제안하는 QT-BCS 프로토콜이 더 우수함을 알 수 있다. 특히 태그 ID가 순차적인 경우와 태그의 수가 많은 경우 뛰어난 성능을 보여 주었다. 이를 활용하여 많은 제품을 생산하는 업체에서 QT-BCS 프로토콜을 활용할 경우 좋은 성능을 보일 것으로 기대한다.

참 고 문 헌

[1] F. Zhou, D. Jin, C. Huang and H. Min, “White Paper: Optimize the Power Consumption of Passive Electronic Tags for Anti-collision Schemes”, In Proceedings of the 5th ASICON, pp. 1213-1217, October 2003.

[2] J. H. Myung, and W. J. Lee, “Adaptive Binary Splitting: A RFID Tag Collision Arbitration Protocol

for Tag Identification", ACM/Springer Mobile networks and Applications (ACM MoNET), Vol. 11, No. 5, pp. 711-722, October 2006.

[3] C. Law, K. Lee, and K. Sju, "Efficient Momoryless 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.

[4] J. H. Ryu, H. J. Lee, Y. H. Seok, T. K. Kwon and Y. H. Choi, "A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID systems", In IEEE International Conference on ICC, pp. 5981-5986, June 2007.

[5] H. J. Lee, J. D. Kim, "QT-CBP : A New RFID Tag Anticollision Algorithm Using Collision Bit Positioning", Emerging Directions in Embedded and Ubiquitous Computing(EUC), LNCS, Springer Vol. 4097, pp. 591-600, August 2006.

[6] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Card and Identification, Second Edition, John Wiley & Sons Ltd, pp, 195-219, March 2003.

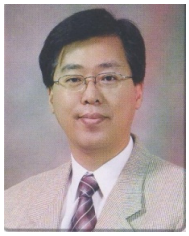
[7] PHILIPS Semiconductor, "I-CODE1 System Design Guide", Technical Report, May 2002.

[8] J. E. Wieselthier, A. Ephremides and L. A. Michaels, "An Exact Analysis and Performance Evaluation of Framed ALOHA with Capture", IEEE Transactions on Communications, COM, Vol. 37, No. 2, pp. 125-137, 1989.

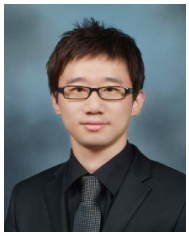
[9] J. H. Choi, D. W. Lee, and H. J, Lee, "Bi-Slotted Tree based Anti-Collision Protocols for Fast Tag Identification in RFID Systems", IEEE Communications Letters, Vol. 10, pp 861-863, December 2006.

[10] Auto-ID Center. "EPCTM Tag Data Standards Version 1.3", Auto-ID Center, September 2005.

◐ 저 자 소 개 ◑



김 정 환
1985년 경북대학교 전자공학과 학사
1987년 경북대학교 산업대학원 전산공학과 석사
1992년 경북대학교 대학원 컴퓨터공학과 박사 수료
1993년~현재 계명문화대학교 컴퓨터학부 교수
관심분야 : 임베디드 시스템 설계, RFID 시스템
E-mail : kjh@kmcc.ac.kr



김 영 태
2005년 대구대학교 정보통신공학부 학사
2009년 경북대학교 대학원 전자전기컴퓨터학부 석사
관심분야 : 임베디드 시스템 설계, RFID 시스템
E-mail : taenystar@knu.ac.kr

박 용 수
1979년 경북대학교 전자공학과 학사
1981년 경북대학교 대학원 전자공학과 석사
2002년 대구가톨릭대학교 대학원 전산통계학과 박사
2007년~현재 경북대학교 BK21 Post-Doc
관심분야 : 임베디드 시스템 설계, ICA
E-mail : timpark75@knu.ac.kr

안 광 선
1972년 연세대학교 전기공학과 학사
1975년 연세대학교 대학원 전자공학과 석사
1980년 연세대학교 대학원 전자공학과 박사
1977년~현재 경북대학교 컴퓨터공학과 교수
관심분야 : 임베디드 시스템 설계
E-mail : gsahn@knu.ac.kr