DOI: 10.3745/KIPSTA.2009.16-A.4.289

RFID 시스템에서 비트 변화 감지를 이용한 가변 슬롯 트리 기반 충돌 방지 알고리즘

김 원 태[†] · 안 광 선[††] · 이 성 준[†††]

요 약

일반적인 RFID 시스템은 하나의 리더와 다수의 태그로 구성되며, 단일 무선 공유 채널을 갖는다. 이로 인해 리더의 요구에 여러 태그들이 동시에 응답할 때 빈번하게 충돌이 발생한다. 태그 충돌을 억제하기 위해 QT[8]와 HCT[10], BSCTTA[2], QT-BCS[9]등과 같은 많은 알고리즘들이 제안되어 왔다. 본 논문에서는 BSCTTA 알고리즘을 기반으로 다중 그룹핑과 비트 변화 감지 센서를 이용한 트리 기반 충돌 방지 알고리즘(TABCS, Tree-based anti-collision algorithm using Bit Change Sensing)을 제안한다. 제안된 알고리즘은 비트 변화 감지 장치를 통해 충돌이 발생하더라도 리더는 태그들이 비트 변화 감지 장치를 이용하여 전달한 비트의 유형을 인식할 수 있게 한다. 또한, TABCS는 유일한 프리픽스를 빠르게 생성하게 하고, 모든 태그의 인식에 필요한 비트 전송량을 줄임으로써 기존 알고리즘에 비해 인식 속도를 빠르게 한다. 시뮬레이션을 통한 성능 평가에서 기존에 제안된 다른 알고리즘에 비해 전체적으로 우수한 성능을 보임을 검증한다.

키워드 : RFID, 충돌방지 알고리즘, 태그 인식, EPC

A Variable-Slotted Tree Based Anti-Collision Algorithm Using Bit Change Sensing in RFID Systems

WonTae Kim[†] · KwangSeon Ahn[††] · SeongJoon Lee[†††]

ABSTRACT

Generally, RFID systems are composed of one reader and several passive tags, and share the single wireless channel. For this reason, collisions occurwhen more than two tags simultaneously respond to the reader's inquiry. To achieve this problem, many papers, such as QT[8], HCT[10], BSCTTA[2], and QT-BCS[9], have been proposed. In this paper, we propose the tree-based anti-collision algorithm using a bit change sensing unit (TABCS) based on BSCTTA algorithm. The proposed algorithm can identify bits returned from tags through bit change sensing unit, even if multi collisions occur. So, it rapidly generates the unique prefix to indentify each tag, and reduce the total of bits. As the result, the cost of identifying all tag IDs is relatively reduced as compared with existing algorithms. It is verified through simulations that the proposed algorithm surpass other existing algorithms.

Keywords : RFID, Anti-Collision Algorithm, Tag Identification, EPC

1. 서 론

RFID(Radio Frequency Identification)는 바코드를 대체할 차세대 인증 기술을 대표할 기술 중 하나로 인식되고 있다. RFID 기술은 주변환경의 모든 사물에 부착되어 무선으로 사물을 인식하는 것이다[1].

RFID 시스템은 태그와 리더 두 가지로 구성된다. 태그와 리더는 무선 채널을 공유하고, 태그는 리더의 요청에 따라 자신의 ID를 RF신호로 리더에게 전송하고, 리더는 태그를 인식하기 위한 작업을 수행한다. 리더의 질의에 대해 다수의 태그가 응답하는 경우 리더는 어떠한 태그도 인식하지 못하게 되는데, 이러한 상태를 충돌 (collision) 이라고 부른다. RFID의 가장 큰 문제점 중 하나인 충돌 문제를 해결하기 위해 많은 충돌 방지 알고리즘 (anti-collision algorithm)이 제안 되어왔다[3].

충돌 방지 알고리즘에 있어서 가장 중요한 것은 태그를 인식하는데 걸리는 시간을 줄이는 것이다. 즉, 하나의 태그

† 종신회원 : 가야대학교 교수
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
††† 정 회 원 : 한국전기연구원 선임연구원(교신저자)
논문접수 : 2009년 2월 25일
수 정 일 : 1차 2009년 5월 19일
심사완료 : 2009년 6월 9일

만이 응답할 수 있는 프리픽스를 빠르게 생성하여 태그에 질의할 수 있는 알고리즘을 필요로 한다.

태그 충돌 방지 알고리즘은 확률적 방식과 트리 기반 방식으로 분류된다[3]. 확률 기반의 태그 충돌 방지 기법은 확률에 근거하여 태그 아이디를 전송할 임의의 시간을 선택하고, 아이디 전송을 반복하는 방법이다. 이러한 방법은 확률에 의존하여 태그가 응답할 시간을 선택하기 때문에 충돌을 완전히 방지하지 못한다. 그러므로, 두 개 이상의 태그가 동시에 같은 응답 시간 슬롯을 선택한다면 아주 긴 시간 동안 인식되지 못하는 기아 현상이 발생할 가능성을 갖는다[3]. 대표적인 확률기반의 태그 충돌 방지 기법은 ALOHA, Slotted ALOHA, Frame slotted ALOHA가 있다[4, 5, 6].

트리 기반의 방법은 충돌이 발생하였을 때, 트리를 기반으로 다음 프리픽스를 선택하고, 질의함으로써 영역 내의 모든 태그를 인식하는 방법이다. 그러나 트리 기반의 방법은 유사 태그가 많은 경우 충돌이 많아져 인식시간이 지연된다. 대표적인 알고리즘으로 이진 검색 트리와 Query Tree (QT) 프로토콜이 있다[7, 8].

최근, 두 가지 방식의 단점을 보안하는 하이브리드 알고리즘이 제안되었다[2, 9, 10]. 즉, 이진 또는 그 이상의 고정된 슬롯을 이용하여 강제적인 충돌을 발생시키고, 이를 트리로 구성함으로써 모든 태그를 인식함과 동시에 질의 횟수 감소시키고 있다. 특히 논문 [2]의 BSCTTA 알고리즘은 리더와 태그 간의 비트 전송 수를 최소화 시킴으로써 빠른 인식 속도를 가지도록 설계되었다.

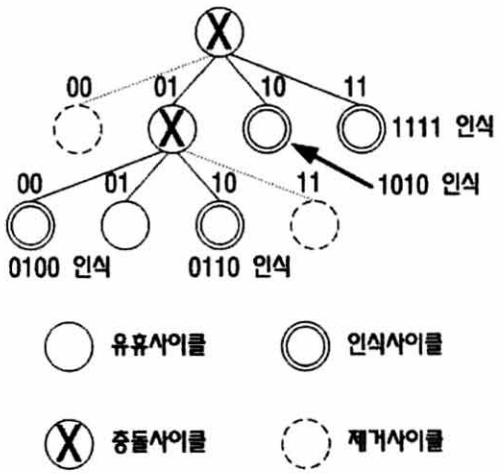
본 논문에서는 BSCTTA 알고리즘을 기반으로 다중 그룹핑과 비트 변화 감지를 이용한 트리 기반 충돌 방지 알고리즘(TABCS, Tree-based collision tracking Algorithm with Bit Change Sensing)을 제안한다. 제안된 방법은 QT-BCS [9]에서 발생하는 문제점을 해결하기 위한 논문이다. 제안된 방법은 BSCTTA 알고리즘[2]과 2단계 그룹핑을 이용하여 각 태그들로부터 순차적으로 비트를 요구하고, 충돌이 발생하면 태그는 충돌 이전 지점을 기준으로 다른 비트의 값이 나타날 때까지 연속적으로 값을 전달한다. 리더는 태그들이 전송한 값에 충돌이 발생하더라고 충돌 위치와 각 태그들이 전송한 값을 파악하는 것이 가능하다. 본 논문에서 제안한 TABCS 알고리즘은 시뮬레이션을 통한 성능 비교에서 기존의 BSCTTA나 QT-BCS보다 나은 성능을 보였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구로 기존에 제안된 하이브리드 알고리즘에 대해 기술한다. 3장에서는 비트 변화 감지를 이용한 트리 기반 충돌 방지 알고리즘을 제안하고 동작 방식을 설명한다. 4장에서는 제안된 알고리즘의 성능 평가를 살펴보고, 마지막으로 5장에서 결론을 기술한다.

2. 관련 연구

2.1 하이브리드 쿼리 트리 알고리즘

QT 알고리즘의 단점은 많은 충돌로 인해 불필요한 질의와 응답에 많은 시간을 소모한다는 것이다. 하이브리드 쿼리 트리 알고리즘(HQT, hybrid query tree algorithm)[10]은 4-ary 검색 트리 기법을 적용하여 프리픽스가 1비트에서 2비트씩 증가하도록 확장하였다. 이로 인해, 태그 들 사이의 충돌 사이클을 줄이고 있으나, 유휴 사이클의 증가를 가져왔다. HQT는 이러한 문제점을 해결하기 위해 슬롯 지연 기법을 적용하였다. 슬롯 지연 기법이란 전달된 프리픽스와 일치하는 태그들이 즉시 응답하는 것이 아니라, 특정시간 대기한 후 데이터를 전송하는 기법이다. 다음 (그림 1)은 HQT를 이용한 태그 인식 과정을 나타낸 것이다.



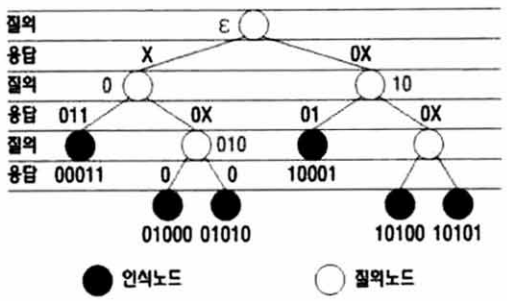
(그림 1) HCT 예제

HQT는 슬롯 지연 기법을 이용하여 유휴 사이클의 수를 줄이는 것을 시도하였다. 즉, 시작 위치와 완료 위치를 점검하여 불필요한 프리픽스의 확장을 줄이는 것이다. 그러나, 제안된 방법에서는 비지(busy) 상태만을 점검하기 때문에, 중간에 전송이 없는 유휴상태에 대한 탐지가 불가능하다. 즉, 00노드와 11노드에대해서만 처리가 가능하고, 01과 10 노드에 대해서는 그 노드가 유휴상태 일지라도 점검이 불가능하다. 또한, 슬롯의 개념을 변형하여 한 슬롯 당 시간을 줄이고 있으나 질의 횟수를 늘려야 하는 문제를 발생시킨다.

예를 들어, 01010000와01010101, 01011100를 ID 값으로 하는 태그들이 존재하고, 리더가 0101의 프리픽스를 전파하였다고 가정하자. 이때, 만일 각 슬롯이 남은 아이디를 위한 충분한 시간을 확보 하였다면 각 태그는 한번에 인식할 수 있다. 그러나, 제안된 논문에서는 시간과 질의 횟수를 단축하기 위한 방법으로 변형된 방법을 고려함으로써, 두 개 이상의 유사한 태그가 존재할 때 서로 다른 슬롯에 할당 됨에도 불구하고 인식하지 못한다. 또한, 상위 예에서 리더가 0101을 전파하고 태그들이 응답한 후, 리더는 충돌을 감지하여 010100, 010101, 010110, 010111을 큐에 저장하게 된다. 다시 말해서, HQT는 010110은 존재하지 않는 프리픽스지만 이를 검출할 능력이 없다.

2.2 이진 슬롯 기반의 충돌 추적 트리 알고리즘

이진 슬롯 기반의 충돌 추적 트리 알고리즘(BSCTTA,



(그림 2) 이진 슬롯 기반의 충돌 추적 트리 알고리즘

bi-slotted collision tracking tree algorithm)[2]은 태그가 전송할 ID의 시작 값에 따라 전송할 슬롯을 결정하고, 비트 값을 순차적으로 전송한다. 또한, BSCTTA에서 리더는 시간 분할하여 충돌이 발생하면 충돌 발생 메시지를 전파하고, 그렇지 않으면 대기시간을 가진 후 자동적으로 다음 비트를 전송한다. 태그들은 충돌 발생 메시지를 받으면 전송을 즉시 중지하고, 리더는 이전까지 인식된 버퍼 값과 프리픽스를 합쳐 다음 프리픽스를 결정한다.

예를 들어, ID의 값이 00011, 01000, 01010, 10001, 10100, 10101인 태그들이 존재 한다고 가정하면 (그림 2)와 같은 인식 과정을 거치게 된다.

BSCTTA 알고리즘은 전체 태그의 수가 N, 태그의 길이를 TAGLEN, 마지막 비트에서 충돌이 발생할 확률을 CPMSB로 표시하면 제안된 알고리즘의 전체 평균 비트 전송량 ATB(N, TAGLEN, CPMSB)은 다음과 같은 식을 만족한다.

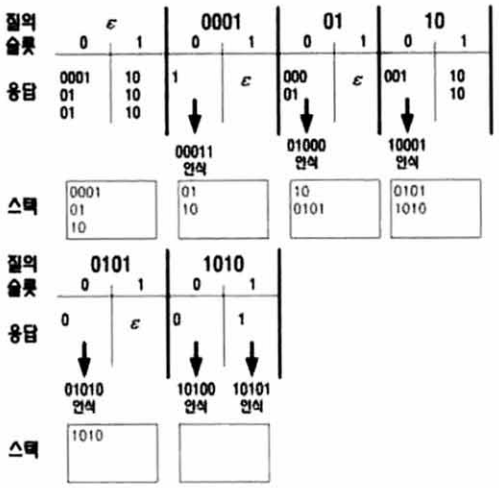
$$ATB(N, Tag_{LEN}, CP_{MSB}) = (Tag_{LEN} - 1) \times N - CP_{MSB}$$

2.3 비트 변화 감지 장치를 이용한 트리 알고리즘

비트 변화 감지 장치를 갖는 트리 알고리즘(QT-BCS, Query Tree algorithm with Bit Change Sensing unit)[9]은 충돌이 발생하더라도 태그들이 전달한 비트 열을 인식하게 함으로써, 빠르게 단 하나의 태그만을 위한 프리픽스를 생성하는 것을 목적으로 한다. 이를 위해, QT-BCS는 비트 변화 감지 (BCS, Bit Change Sensing) 장치를 이용한다. BCS는 전달된 프리픽스와 일치하는 각 태그들이 프리픽스 이후의 비트 열에서 서로 다른 값을 가지는 부분까지 추출하는 장치이다. 리더는 BCS를 통해 전달된 비트 열은 다중의 충돌이 발생하더라도 모든 전송된 비트 유형을 복원할 수 있다.

예를 들어, 어떤 프리픽스와 일치하는 각 태그가 001과 01을 전송하면 리더의 버퍼에는 0x1로 적재된다. 리더는 x인 지점까지 복사한 후 x를 1로 대체하여 태그가 전달한 패턴을 인식할 수 있다. 또한 버퍼의 x는 0으로 채워 넣음으로써 001의 태그 값을 인식할 수 있다.

QT-BCS의 단점은 태그의 수가 단지 하나만 존재 한다고 할지라도, 한번에 모든 ID 비트의 값을 인식할 수 없다. 그러므로 태그의 수가 소수인 경우 QT-BCS는 질의 수를 증가시키게 되어 결국 태그의 인식에 필요한 프리픽스의 길이를 증가시켜 모든 태그의 인증에 드는 시간 비용을 증가시키게 된다. 그러므로, 태그들의 수가 소수이거나, 균등 분포를 이루는 경우에도 빠르게 태그를 인증할 수 있는 방법을 필요로 한다.



(그림 3) 비트 변화 감지 장치를 이용한 트리 알고리즘

3. 비트 변화 감지 장치를 이용한 이진 슬롯 충돌 추적 트리 알고리즘

본 논문에서는 BSCTTA 알고리즘을 기반으로 다중 그룹핑과 비트 변화 감지 센서를 이용한 트리 기반 충돌 방지 알고리즘(TABCS, Tree-based collision tracking Algorithm using Bit Change Sensing)을 제안한다.

TABCS 알고리즘의 동작 환경은 하나의 리더와 다수의 태그로 구성되고, 모든 태그들은 리더의 질의에 동시에 응답한다고 가정한다. 또한, 모든 태그들은 인식된 후 특정 요구를 제외한 어떠한 질의에 대해서도 자신의 ID 값을 전송하지 않는다고 가정한다.

3.1 비트 변화 감지 장치

기존의 하이브리드 충돌방지 알고리즘은 각 태그들을 위해 유일한 프리픽스를 빠르게 생성하여 전달함으로써 모든 태그들을 인식하는 시간을 단축시키는 것을 목적으로 한다. 이를 위해, BSCTTA는 리더가 전달된 특정 비트의 충돌을 감지하면 전송중인 모든 태그들에게 전송 중지 명령을 전파하고, 새로운 프리픽스를 생성하여 질의하였다. 이러한 방법에서 프리픽스의 길이를 PLEN이라고 하고, 나머지 태그 ID의 길이를 $R_{LEN}(= TAG_{LEN} - P_{LEN})$이라고 하면 첫 단계와 마지막 아이디 비트의 충돌이 발생하는 경우를 제외한 나머지 단계에서 태그 인식에 성공한다면, 인식 단계에서 필요한 전송 비트의 수는 다음과 같이 정의된다.

$$P_{LEN} + 2 \times R_{LEN}$$

위 식에서 알 수 있는 바와 같이, 프리픽스의 길이를 줄이는 것보다 태그들이 전달하는 잔여 ID의 비트 값을 줄이는 것이 인식속도를 빠르게 할 수 있다. 이를 위해 본 논문에서는 비트 변화 감지(BCS, bit change sensing) 장치를 사용한다.

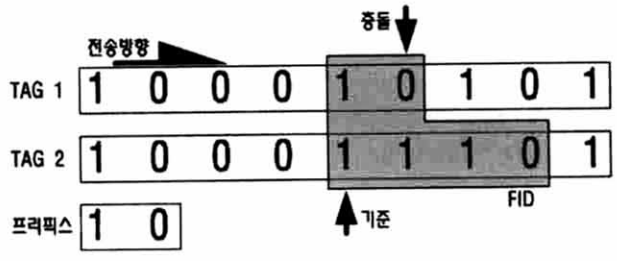
논문 [9]에서는 BCS장치를 이용함으로써 더 긴 ID 값을 인식하는 것이 가능하다는 것을 보였다. 그러나, QT-BCS는 전체 ID 값을 여러 개의 부분들로 분리하여 전송하기 때문에, 많은 질의 횟수와 통신 비용을 갖는다. 이러한 문제를 해결하기 위해, 본 논문에서는 BSCTTA 알고리즘[2]을 이용하여 태그는 비트 단위로 ID 값을 전송하면서, 충돌 정보가 전파되면 충돌 이전 지점을 이용하여 BCS 장치를 동작시킨다.

예를 들어, 태그 1의 ID가100010101, 태그 2의 ID가 100011101이고, 리더가 전달한 프리픽스가 10이라고 가정하자. $1 \le i,k \le Tag_{LEN}$, $i<k$ 일 때, IDi가 i번째 비트 값을, IDi,k는i에서 k사이의 비트 값 들을 나타낸다. 태그들은 리더가 전달한 프리픽스를 만족하므로 ID3부터 자신의 ID값 전달을 시도하고, ID6전송 후 리더로부터 중지 요청을 받게 된다. 이후, 태그 1은 ID5를 기준 비트로 하여 다른 비트 값을 가지는 ID8번 비트까지 더 전송하게 된다. 태그 2의 경우 기준 위치와 충돌위치의 비트 값이 다르기 때문에 아무런 전송이 발생하지 않는다. 이때 태그 1의 ID5,8과, 태그 2의 ID5,6을 FID로 정의한다.

[정의 1] 어떤 태그 ID의 어떤 위치(IDi+1)의 값을 전달한 후 리더로부터 충돌이 발생하였다는 메시지가 전달되면, 태그는 이전 비트(IDi)를 기준 비트로 하여 다른 값이 나타나는 비트(IDk)까지의 영역(IDi,k)을 ID의 일부분(FID, a fragment of ID)로 정의한다.

리더의 버퍼 영역은 인식된 데이터 영역과 FID 버퍼 영역(FIDB)으로 분류되고, FIDB의 길이는 가장 긴 FID의 길이와 같다. FID에서 모든 비트 열은 동일한 값의 연속과 비트의 마지막 위치를 알리는 다른 값의 접합으로 이루어 져 있다. 그러므로, 동일한 슬롯에서 전송 받은 동일한 길이의 비트 열은 동일한 비트 값을 갖는다.

[보조정리 1] FIDB에서 충돌이 발생한 위치는 어떤 태그가 전달한 마지막 비트의 위치와 같다.



(그림 4) 태그의 BCS에 의한 FID 추출방법

증명: FID로 분리된 비트 열은 FID의 MSB 비트와 동일한 값을 갖는 연속된 비트 열로 마지막 비트만 다른 값을 갖는다. 충돌은 서로 다른 값이 전송될 때 발생하므로 FID를 이용한다면, 태그 ID의 마지막 비트를 전송하는 경우를 제외하고 동일한 길이를 갖는 비트 유형에서는 충돌이 발생하지 않는다.

그러나, 서로 다른 길이를 가지고 있다면 상대적으로 길이가 짧은 비트 열의 마지막 위치는 길이가 긴 비트 열보다 짧기 때문에, 필연적으로 다른 값을 가지게 되고, 해당위치에서 반드시 충돌이 발생한다.

[보조정리 2] FIDB에서 $n(1 \le n \le FID_{LEN}-1)$개의 충돌이 존재한다면, 태그들이 전송한 비트 유형(pattern) 의 수는 n + 1 이다.

[증명] n+1개의 서로 다른 길이의 FID가 존재한다면 FIDB에는 n개의 충돌이 감지된다고 가정하자.

증명을 위해, FID의 길이에 따라 내림차순으로 정렬하고, FIDB의 길이를 m이라고 가정하자. b = {0, 1} 일 때, b의 반대 값을 b'로 표현한다. $FID_{ORD}(i), 0 \le i < n$는 길이가 i번째로 긴 비트 열이고, $FID_{LEN}(i), 0 \le i < n$ 이 i번째 비트 열의 길이를 표현한다. 이때 FID의 길이에 대해 다음과 같이 2가지 경우가 생긴다.

$$\begin{cases} FID_{LEN}(0) = FID_{LEN}(1) \\ FID_{LEN}(i) > FID_{LEN}(i+1) \end{cases}$$

$FID_{LEN}(1)=FID_{LEN}(0)$ 인 경우

어떤 두 태그들이 마지막 FID 값을 전송하는 경우이다. 마지막 FID의 유형은 $m \ge 2$ 일때, $\{b^m\}$과 $\{b^{m-1}b'\}$ 의 유형이 존재한다. 만일 $\{b^m\}$이 마지막 FID가 아니라면 BCS에 의해 $\{b^m b'\}$로 나타나며, 길이는 m+1이 되므로 $\{b^{m-1}b'\}$ 보다 길다. 그러므로, $\{b^m\}$은 마지막 FID에서만 나타난다. 두 개의 태그는 FIDB의 마지막 비트에서 충돌이 발생하기 때문에, BSCTTA [2]에서와 같이 두 개의 태그를 인식했다고 정의할 있다. 그래서, 이러한 두 개의 태그는 선 작업을 통해 인식된 것으로 간주하고 FIDB에서 제거하였다.

$FID_{LEN}(i+1)<FID_{LEN}(i)$인 경우

$FID_{ORD}(i)$ 의 비트 열은 길이가 $FID_{LEN}(i)$ 이고, $\{b^{FID_{LEN}(i)-1}b'\}$로 표현된다. $FID_{LEN}(i+1) \le FID_{LEN}(i)-1$을 만족하기 때문에 $\{b^{FID_{LEN}(i)-1}\}$의 영역 내에 $FID_{ORD}(i+1)$의 마지막 비트가 존재한다. 그러므로 충돌은 $FID_{ORD}(i)$의 마지막 비트를 제외한 나머지 비트에서 발생한다.

[정리 1] 리더가 전달한 프리픽스 (P)를 만족하는 인식

영역내의 다양한 태그들이 BCS를 이용하여 전달한 여분의 비트 열은 리더의 버퍼(B)에서 다중의 충돌이 발생하더라도 태그들이 전달한 모든 비트 열의 유형을 인식한다.

[증명] 보조정리 2에서 모든 태그들이 전달한 FID의 마지막 위치에서 충돌이 발생하므로 이를 역으로 사용하면 모든 비트 열의 유형을 얻을 수 있다. FIDB가 $b_0 \dots b_i \dots b_n$으로 이루어져 있을 때, i 번째 비트에서 첫 번째 발생한 충돌이 발생하였다면, $b_0 \dots b_{i-1} + b_0'$ 를 통해 어떤 하나의 FID 값을 생성할 수 있다. 이후 $b_i = b_0$로 설정한다. 또한 다른 위치 $j, i > j$에서 충돌이 발생하는 경우 $b_0 \dots b_{j-1} + b_0'$로 인식하고 $b_j = b_0$로 설정해 나감으로써 FIDB의 모든 비트 값을 점검하여 모든 유형의 FID값을 인식할 수 있다. 또한 FIDB의 모든 비트 유형을 검색하기 위해 필요한 시간은 은 $O(n)$이다.

[따름정리 1] FIDB 내에 존재하는 충돌 비트는 정리 1에 의해 사후 작업을 통해 정확한 의미를 인식할 수 있기 때문에 그 비트 값의 결정이 지연된 지연 비트(DB, delayed bit)이다.

3.2 프리픽스 선택

QT-BCS 알고리즘은 인식된 모든 비트 유형을 다음 프리픽스로 사용한다. 이로 인해 빠르게 프리픽스를 생성하는 것이 가능하지만, 많은 질의 수와 유휴 슬롯의 수를 가지게 되고, 따라서 태그 인식에 많은 시간을 소비하게 된다. 또한, 어떤 한 슬롯에서 인식된 비트 열의 유형은 중첩된 값을 갖는다. 다시 말해, 어떤 슬롯에서 인식된 비트 유형은 마지막 비트만 다르기 때문에 일정 부분의 비트 열은 중첩된다. 그러므로, 한번의 질의를 통해 모두 인식될 수 있는 태그들조차 2번의 질의를 필요로 하게 된다.

예를 들어, 001과 00001로 시작하는 태그들은 BSCTTA 알고리즘을 이용하는 경우 00의 프리픽스를 전파하면 0 슬롯에서 00001로 시작하는 태그 ID를, 1슬롯에서 001 로 시작하는 태그의 인식이 가능하다. 그러나, QT-BCS의 경우 001과 00001을 모두 질의로 사용하여야 하므로 많은 유휴 슬롯과 질의 횟수를 필요로 한다. 본 논문에서는 프리픽스 생성 과정에서 인접한 두 비트 유형에 대해 긴 비트 열을 중심으로 최대의 공통 부분인 질의 비트 열(QBS, query bit stream)과, 나머지 부분을 잔여 비트 열(RBS, remain bit stream)의 쌍인 (qbsi, rbsi) 구조체로 큐(Queue)에 저장된다. QBS 은 리더가 프리픽스를 전파하는 과정에서 사용되고, QBS 의 마지막 비트 값과 태그들이 응답하는 슬롯에 따라 RBS 을 QBS 과 조합함으로써 프리픽스의 길이를 축소시킬 수 있다. <표 1>은 최적화된 프리픽스 선택을 위한 알고리즘이다.

Padding(length, bit value) 함수는 길이와 채울 비트 값을 파라메터로 받아 비트 값을 주어진 길이만큼 채운다.

〈표 1〉 프리픽스 선택 알고리즘

```
1. i = B_LEN
2. QBS = null
3.
4.  If b_i == DB then
5.          PrePrefix + Padding(i, b_1) 과 PrePrefix + Padding(i -
            1, b_1) + !b_1를 인식 태그 리스트에 각각 저장
6.          i = 다음 DB의 위치
7.          RBS = not b_1
8. Else
9.          If i + PrePrefix_LEN == Tag_LEN then
10.             While b_i != DB
11.                 RBS = b_i
12.             PrePrefix + Padding(i - 1, b_1) + RBS를 인식
            태그 리스트에 저장
13.             RBS = not B_1
14.         End If
15. End If
16. If  i <= 0 then 종료
17. While i != 0
18.         If  b_i != DB then
19.             RBS = b_i + RBS
20.         Else
21.             RBS = b_1 + RBS
22.             QBS = Padding( i - 1, b_1)
23.             AddQueue( QBS, RBS )
24.             i = 다음 DB의 위치
25.             RBS = not b_1
26.             QBS = null
27.         End If
28. End While
29. If QBS is null then
30.         AddQueue(RBS, null )
31. End If
```

AddQueue(QBS, RBS) 는 QBS와RBS를 받아 쌍으로 이루어진 구조체를 큐에 저장하는 함수이다.

먼저 리더는 태그들로부터 받은 버퍼의 길이(B_{LEN})를 정의하고(표 1의1 line), 마지막 비트가 충돌이 발생하였는지 검사한다. 마지막 비트의 DB은 태그 전체 길이의 마지막 비트에서만 발생하므로 만일 인식된 버퍼의 마지막 비트에 DB가 발생하였다면 리더는 2개의 태그를 인식한다(4~5 line). 이 후 LSB에서 충돌이 발생한 이전지점을 찾아 i에 저장한다(6 line). 만일 충돌이 발생하지 않았다면 이전 프리픽스의 길이와 합이 전체 태그의 길이와 같은지 평가한다(9 Line). 길이가 같으면, 하나의 태그를 인식한 것으로 평가된다(10~13 line). 이후 버퍼에 검사할 값이 남아 있는 경우 LSB로부터 1비트씩 버퍼의 값을 평가하여 DB가 발생하지 않은 버퍼의 값은 잔여 비트열의 앞에 접목시키고(19 line), DB가 발생한 어떤 지점(bi)에 도달한 경우 Padding(i - 1, b1) 를 수행하여 QBS로 저장한다. 짝수 번째에 만나는DB는

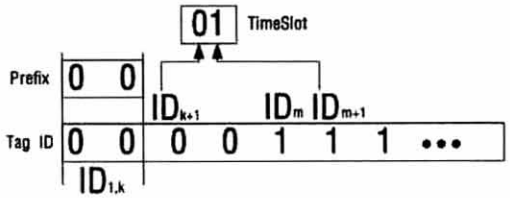
RBS의 생성에 사용되고, 홀수 번째 만나는 DB는 QBS로 생성되어 저장된다(21~26 line). 검사 위치가 MSB에 도달하지 않은 경우 작업을 반복 수행한다. 검사가 완료되었으나 충돌이 없는 경우 RBS 값을 QBS로 대체하여 큐에 저장한다(29~31 line).

예를 들어, 리더가 태그들로부터 받은 FID 값이 01, 001, 0001, 00001, 000000001 이라고 하면, 리더의 버퍼에는 0xxxx001로 적재되고 버퍼의 길이는 8이 된다. 태그의 전체 길이가 10비트인 경우 인식된 태그는 존재하지 않으므로 LSB인 b8의 값부터 충돌을 검사한다. b5에서 충돌이 발생하였기 때문에 QBS는 0000, RBS는 0001로 채워져 큐에 저장된다. 다시 b4에서 발생한 충돌은 무시되고, b3에서 발생한 충돌로 RBS에는 01이 생성되고, QBS은 00의 값을 가지게 된다. 마지막 검사에서는 RBS은 01이지만 충돌이 발생하지 않았기 때문에 이 값은 QBS로 저장되고 RBS은 null 값을 저장하게 된다. 그러므로 이후 프리픽스는 ε -> 0000 -> 00 -> 01의 순으로 발생된다. 그러나 BSCTTA의 경우 ε -> 0 -> 00 -> 000-> 0000의 순으로 5번의 질의와 응답 과정이 진행된다. 또한 QT-BCS의 경우 ε -> 01 -> 001 -> 0001 -> 00001 -> 000000001의 순으로 가장 많은 6번의 질의 / 응답 과정이 필요하다. 본 논문에서 제안하는 TABCS는 시뮬레이션에서 다른 기존의 알고리즘에 비해 질의 횟수를 20% 이상 줄이고 있음이 입증되었다.

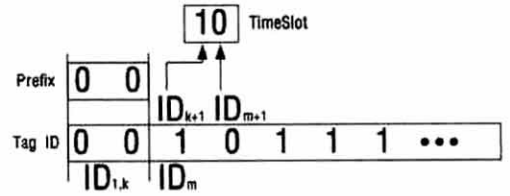
3.3 2단계 그룹핑

논문 [10]에서는 4-ary 탐색 알고리즘이 태그의 분포 환경에 따라 충돌 사이클의 수를 1/3으로 줄일 수 있음을 보이고 있다. 4-ary 방식과의 차이점은 전송할 ID의 첫 2비트 값을 이용하여 해당 슬롯을 결정하는 것이 아니라, 프리픽스 유형과 특정 위치의 비트 값에 따라 전송 슬롯을 결정한다. 다시 말해서, 태그는 리더가 전달한 프리픽스의 마지막 2비트를 검사하여 두 비트가 같은 값을 가지는 경우 4개의 슬롯 사용하고, 다른 값을 갖는 경우 2개의 슬롯을 사용한다. 이러한 방법을 이용하면, 프리픽스의 유형에 따라 슬롯의 수를 결정하기 때문에 리더와 태그는 특정 질의에 대해 슬롯의 수를 정의하는 동기화 과정이 필요하지 않다. 본 논문에서 이러한 방법을 가변형 슬롯 결정 방법이라고 정의한다.

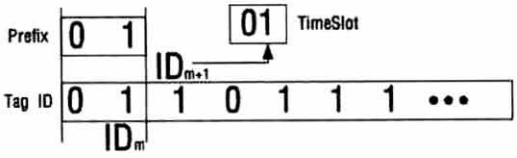
이렇게 가변형 슬롯의 수를 결정하는 이유는 TABCS의 프리픽스 선택과정에서 하나의 프리픽스를 통해 2개의 태그를 인식하기 위해 DB가 발생한 앞 지점까지를 질의로 선택하기 때문이다. 다시 말해서, DB발생 이전 지점까지를 질의 비트 열로 사용하기 때문에 마지막 2비트가 동일한 값을 갖는다면 질의로 전달된 프리픽스는 이미 2개의 서로 다른 프리픽스를 가지고 있는 것과 동일하다. 다시, 각각의 프리픽스는 프리픽스 다음 값에 따라 2개의 유형으로 분리될 수 있다. 그러나, 마지막 2비트가 다른 값을 갖는다면 하나의 프리픽스만 가진 것이므로 프리픽스 다음 값에 따라 2개의 유형으로 분리될 수 있다. 태그들이 자신의 비트 값을 전송할 슬롯을 결정하는 것은 다음과 같은 방법으로 구현될 수 있다.



(a) 프리픽스의 마지막 2비트가 동일하고 IDk+1이 IDk-1과 동일한 경우



(b) 프리픽스의 마지막 2비트가 동일하고 IDk+1이 IDk-1과 다른 경우



(c) 프리픽스의 마지막 2비트가 다른 경우

(그림 5) 2중 그룹핑 예제

길이가 k인 프리픽스(P)를 Pk, P를 만족하는 태그의 어떤 위치 IDj와 프리픽스의 위치 Pj는 동일한 위치를 나타내며, 태그가 가진 ID에서 IDk-1과 다른 값을 갖는 위치를 IDm으로 표시한다고 가정한다. 태그는 먼저 전달된 프리픽스의 Pk와 Pk-1비트의 값을 비교하고, IDk+1과 AND연산을 수행하여 1차 그룹핑에 따르는 슬롯을 선택한다. 태그는 Pk-1 비트와 다른 값을 갖는 위치 IDm을 찾고 IDm+1을 2차 세부 그룹의 값으로 선택한다. 프리픽스의 길이가 1인 경우 프리픽스의 Pk와 Pk-1 비트는 동일한 위치로 결정한다. 이를 식으로 나타내면 다음과 같다.

왼쪽 타임 슬롯 번호 = (Pk-1 ⊗Pk) • IDk+1
오른쪽 타임 슬롯 번호 = IDm+1

3.4 TABCS 알고리즘

본논문의 목적은 질의에 필요한 비트의 길이를 줄임으로써 상대적으로 인식 속도를 증가시키기 위한 것이다. 이를 위해 알고리즘은 전파, 프리픽스 검사, 프리픽스 그룹핑, ID 그룹핑, 전송, 인식, 비트 변화 감지, 프리픽스 선택, 8단계로 분리된다. 리더가 수행하는 알고리즘은 다음과 같은 과정을 갖는다. 초기 큐에는 'ε'이 존재한다.

1) REQUEST: 리더는 스택에서 저장된 질의를 얻어 영역 내의 모든 태그들에게 전파한다.
2) GROUPING: 리더의 인식 영역내의 태그들은 전달된 프리픽스와 일치하는 ID 값을 가진 경우 리더에게 소유한 ID 값을 되돌려준다.
   · 리더에게 응답할 때 태그들은 Pk-1과 Pk, IDk+1,

IDm+1을 이용하여 ID를 전달할 슬롯을 결정한다.
- 왼쪽 타임슬롯 번호 = (Pk-1 ⊗k) • Dk+1
- 오른쪽 타임 슬롯 번호 = IDm+1

3) TRANSMISSION: 태그는 IDm+1으로부터 비트 변화 감지 비트를 참조하여 한 비트씩 전송한다.
- 태그는 한 비트를 전송한 후 특정 시간을 기다린다.
- 리더로부터 중지 요청 메시지가 전달되면 태그는 비트 변화 감지 비트를 1로 설정한다.
- 감지 비트가 1인 경우: 이전 비트와 비교하여 다른 경우 중지한다.
- 감지 비트가 0인 경우: A단계로 이동한다.

4) RECOGNITION: 태그들이 전달하는 비트 값을 버퍼에 적재한다.
- 만일 어떤 비트에 대해 인식 불가 상태가 되면, 리더는 즉시 태그들에게 중지 요청을 한다.
- 1비트 전송 시간에 아무런 응답이 없으면 다음 슬롯 전송 메시지를 전파한다.
- 리더는 태그들이 비트 변화 감지 센서 장치를 통해 얻은 값의 전송이 완료될 때까지 버퍼에 적재한다.

5) DECISION: 버퍼의 내용을 조사하여 태그들이 전달한 값을 추출한다.
- 리더는 버퍼에서 각 태그들이 전달한 값을 불리하여 저장한다.
- 질의에 사용된 프리픽스와 인식된 전달 값의 합이 태그 ID의 길이와 같은 경우 인식 리스트에 삽입한다.
- 그렇지 않다면, 리더는 프리픽스와 추출된 값을 합쳐 큐에 넣는다.

6) 큐가 널이 아니라면 1)로 이동한다.

4. 성능평가

모든 태그는 리더의 인식 영역 내에 존재하고, 리더의 요청에 동시에 응답한다고 가정한다. 본 논문에서 제안하는 알고리즘의 성능 평가를위해 C# 언어를 이용하여 BSCTTA과 QT-BCS, TABCS 알고리즘을 구현하고, 다양한 부분에 대한 정보를 수집하고, 비교하였다. 시뮬레이션은 각각 100번의 샘플링과 실행을 수행하였다.

4.1 환경

성능 분석을 위한 각 샘플 태그들의 아이디 값은 EPCglobal에서 제안하는 태그 데이터 표준 (EPCglobal Tag Data Standard Version 1.4)에서 제안된 코드들 중 가장 널리 알려진 범용 식별자 (General Identifier, GID-96) 에 따라 샘플 코드를 선택하였다. 리더와 태그 사이에 데이터 전송률은 EPCglobal에서 제안된 ISO/IEC 18000-6 Type C의 보편적 스피드로 알려진 80kbps로 정의하였다.

GID는 96 bit의 길이를 가지며, Header, General Manager Number, Object Class, Serial Number 4가지로 구성된다. Header는 데이터 유형 및 길이를 정의하는데 일반적으로

Field Name	Header	General Manager Number	Object Class	Serial Number
Length	8	28	24	36

(그림 6) 범용 식별자 구조

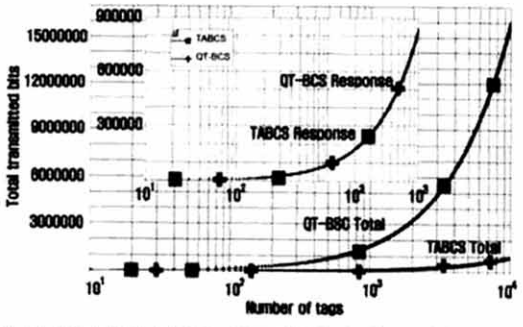
00110101의 고정된 값을 갖는다. General Manager Number는 상품에 대한 분류와 일련번호를 관리하는 기관이나 기업을 표시한다. Object Class는 바코드의 상품 품목코드에 해당하는 것으로 각 품목 또는 단위를 표시하고 Serial Number는 각 상품들에 대하여 부여되는 고유한 식별번호를 나타낸다

일반적으로 동일한 회사의 경우 범용 관리자 번호 (General Manager Number)는 같은 값을 가지게 되며, 동일한 유형의 제품인 경우 객체 분류 코드(Object Class) 또한 동일한 값을 가지게 된다. 만일 RFID 기술을 물류 분야에 적용하게 되면 각 태그의 일정 부분의 비트 값에 대한 중복이 많아지는 가능성이 크다는 것을 의미한다. 또한 객체의 일련번호 또한 순차적으로 존재할 확률이 높다. 이와는 반대로 유비쿼터스 스마트 홈 기반의 가정에서는 서로 다른 다양한 제품을 사용하게 됨으로 상대적으로 태그 ID의 중복은 낮을 것이다. 그러므로 본 논문에서는 이러한 2가지 상황에 대해 고려하여 환경을 구성하고 시뮬레이션을 통해 이를 분석하였다.

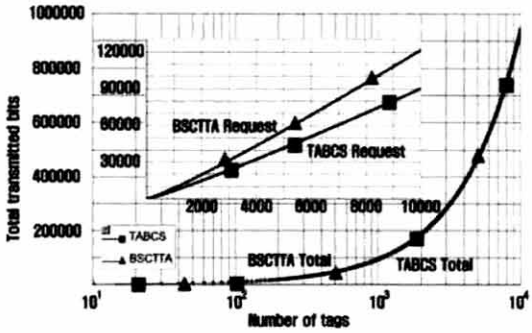
4.2 상황 1: 완전 임의 추출

첫 번째 시뮬레이션에서 고려되는 환경은 모든 태그 ID의 값이 임의적으로 생성된 환경이다. 이러한 상황을 고려하여 전체 태그 ID값을 임의로 10개에서 10000개까지 점차적으로 증가시키면서 생성하고 100번의 시뮬레이션을 수행하였다.

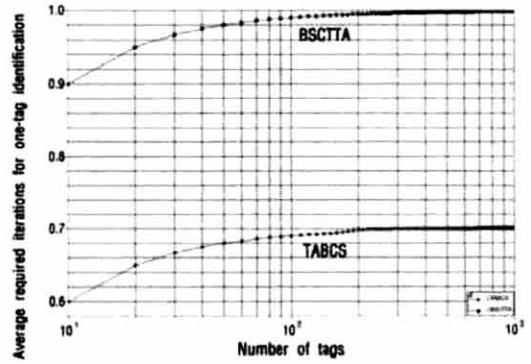
(그림 7)은 QT-BSC와 TABCS 알고리즘의 모든 태그의 인식을 위해 필요한 전체 비트 전송량과 응답 비트 전송량을 비교한 것이다. 태그의 총 응답 비트의 수는 유사하지만, 총 비트 전송 량은 약 17배 감소하였다. 이러한 이유는 QT-BCS알고리즘이 태그를 인식하기 위해 많은 질의와 비트 열의 길이를 요구한다는 것을 의미한다. 다시 말해서, QT-BCS 알고리즘은 충돌이 발생하지 않는 상황에서 태그 ID 값을 모두 인식할 수 있는 방법이 존재하지 않기 때문에 발생한다.



(그림 7) QT-BCS와 TABCS 알고리즘간의 태그 인식을 위해 필요한 총 전송 량과 전체 응답 비트 수 비교



(그림 8) QT-BCS와 TABCS 알고리즘간의 태그 인식을 위해 필요한 총 전송 량과 전체 요구 비트 수 비교



(그림 9) 태그 하나를 인식하는데 필요한 평균 반복 요구 횟수

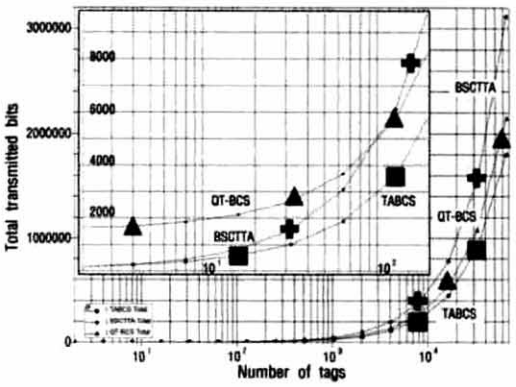
(그림 8)은 BSCTTA와 TABCS알고리즘의 모든 태그의 인식을 위해 필요한 전체 비트 전송량과 전체 요청 비트 수를 비교한 것이다. 전체 전송 비트 수는 1000개의 경우 BSCTTA가 95000이고 TABCS가 94592로 거의 변화가 없었다. 그러나 전체 요구 비트 수는 태그의 수가 10개인경우 약 20%, 10000개의 경우 26%정도 감소 되었다. 즉, TABCS의 질의가 더 효율적이라는 것을 보여주고 있다.

(그림 9)는 BSCTTA와 TABCS 알고리즘이 하나의 태그를 인식하기 위해 필요한 반복 횟수를 그래프로 나타낸 것이다. 시뮬레이션 비교에서 제안된 알고리즘인 TABCS는 기존 알고리즘인 BSCTTA에 비해 약 30%정도 적은 질의 횟수를 가졌다. 이러한 이유는 가변 4-ary 슬롯을 통해 인식할 수 있는 태그의 수를 증가시켰기 때문으로 이해된다.

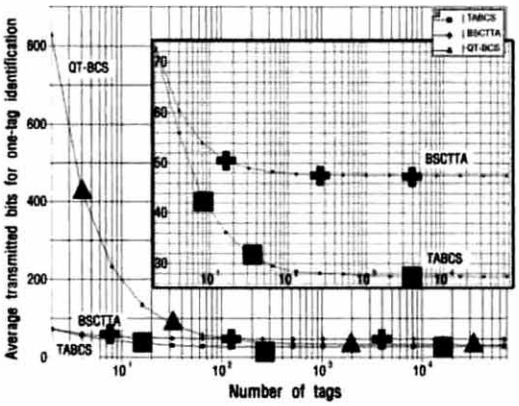
4.3 상황 2: 36비트 순차 증가 태그

두 번째 환경은 GID-96코드에서 사용되는 96비트 중 serial number만을 2개에서 65536까지 2의 배수로 증가시켜 시뮬레이션에 사용하였다.

(그림 10)에서 볼 수 있는 것처럼, 모든 태그를 인식하기 위해 요구되는 전체 전송 비트 수는 128개 이후 QT-BCS가 BSCTTA보다 높은 성능을 보였고, 제안된 TABCS는 QT-BCS와 비교해서 태그의 수가 16개 일 때 약 73%, 1024개 이후로는 지속적으로 약 20%정도의 성능향상을 보이고 있다. BSCTTA와 비교해서는 태그의 수가 16개 일 때 약 28%, 16384 개인 경우 약 43%의 성능 향상을 보였다.



(그림 10) 알고리즘 간 태그 인식을 위해 필요한 총 전송량



(그림 11) 하나의 태그를 인식하기 위해 필요한 평균 전송 비트 수

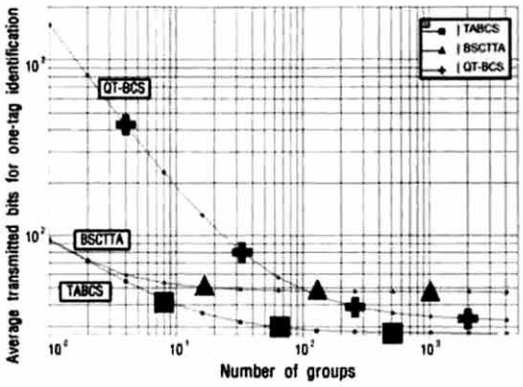
(그림 11)은 각 알고리즘이 하나의 태그를 인식하기 위해 요구되는 평균 전송 비트 수를 나타낸다. 제안된 TABCS 알고리즘은 다른 기존 알고리즘에 비해 감소하였으며, 100개 이후로는 BCSTTA에 비해 약 30%의 성능 개선을 보였으며 16384이후 QT-BCS와 근사한 값을 보였다.

4.4 상황 3: 그룹으로 분류된 순차증가 태그

실제 물류 유통에서 발생되는 상황에서는 동일한 회사의 경우 Header와 General Manager Number, Object Class가 같고 Serial Number는 순차적으로 증가할 가능성이 크다고 보고 시뮬레이션을 수행하였다. 이러한 환경에서 모든 태그의 수는 4096개로 고정하고 header와 General Manager Number, Object Class를 1에서 4096까지 2n ($0 \le n \le 12$)으로 증가 시키면서 시뮬레이션을 수행하였다. 다시 말해서, 태그의 수는 동일하기 때문에 그룹의 수가 작다면 태그는 순차적으로 나열되고, 그룹의 수가 많아지면 랜덤하게 나열되는 것에 가까워진다. (그림 12)는 그룹에 따르는 알고리즘의 모든 태그를 인식하기 위해 필요한 전체 전송 요구 비트 수를 나타낸다.

5. 결 론

RFID 시스템의 중요한 문제점 중 하나는 충돌 방지 알고



(그림 12) 그룹 수의 변화에 따른 총 전송 요구 비트 수

리즘의 구현이다. 이를 위해 다양한 방법이 제안되어 오고 있다.

본 논문에서는 TABCS라고 불리는 충돌 방지 알고리즘을 제안한다. 태그들은 비트 단위로 태그 ID 값을 순차적으로 전파하고, 충돌이 발생하였을 때 BCS 장치를 통해 FID로 불리는 여분의 비트들을 전송할 수 있게 함으로써 모든 태그를 인식하는데 필요한 전체 전송 비트 수를 줄이는 알고리즘을 제안하였다. 리더는 마지막 비트에서만 충돌이 발생한다는 점을 이용하여 태그들이 전달한 비트 열을 인식하는 것이 가능하다. 또한 가변 슬롯을 이용하여 불필요한 슬롯의 수를 줄일 수 있었다.

알고리즘의 성능을 측정하기 위해 기존의 제안된 QT-BCS와 BSCTTA 알고리즘에 대해 다양한 환경에서 비교하였다. 시뮬레이션 결과에서 제안된 논문은 태그의 수가 작은 경우 기존의 QT-BCS보다, 태그의 수가 많은 경우에는 BSCTTA보다 개선된 성능을 가지고 있음을 알 수 있었다. 또한 제안한 알고리즘인 TABCS는 하드웨어적인 복잡도가 낮아 실제 시스템에 구현하기 쉬울 것으로 예상된다. 따라서 비트 변화 감지(BCS) 장치와 가변형 슬롯 선택 방법을 이용하여 태그의 수에 상관없이 태그를 안정적인 성능을 갖는 TABCS를 이용한다면 시스템의 높은 성능향상을 가져올 것으로 기대된다.

참 고 문 헌

[1] F. Mattern, "The Vision and Foundations of Ubiquitous Computing," Upgrade, Vol.2, No.5, pp2-6, Oct., 2001.

[2] J. H. Chio, D. W. Lee, H. J. Lee, "Bi-Slotted Tree based Anti-Collision Protocols for Fast Tag Identification in RFID Systems," IEEE Commun. Lett., Vol,10, No.12, pp. 861-863, Apr., 2006.

[3] J.H. Myung, 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, Oct., 2006.

[4] "EPCTM Radio-Frequency Identification Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz Version 1.0.9," EPCglobal, Jan., 2005.

[5] "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, Nov., 2003.

[6] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. John Wiley & Sons, 2003.

[7] 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, Apr., 2006.

[8] J.H. Chio, D.W. Lee, H.J. Lee, "Query tree-based reservation for efficient RFID tag anti-collision," IEEE Commun. Lett., Vol,11, No.1, pp.85-87, Apr., 2006.

[9] Y.T. Kim, S.J. Lee, K.W. Ahn, "An Efficient Anti-collision Protocol Using Bit Change Sensing Unit in RFID System," In Proc. 14th Inter. Conf. on RTCSA, IEEE, pp.81-88. Aug., 2008.

[10] J. Ryu, H. Lee, Y. Seok, T. Kwon and Y.Choi, "A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID system," IEEE ICC, pp.5981-5986, Jun., 2007.

[11] C. Law, Kayi Lee, and K. Y. Sju, "Efficient Momoryless Protocol for Tag Identification," In Proc. 4th Inter. Workshop on DIALM, ACM, pp. 75-84. Aug. 2000.[1] F. Mattern, "The Vision and Foundations of Ubiquitous Computing," Upgrade, Vol.2, No.5, pp.2-6, Oct., 2001.

김 원 태

e-mail : wtkim@kaya.ac.kr

1983년 경북대학교 전자공학과(학사)

1993년 경북대학교 전자공학과(공학석사)

1998년 경북대학교 컴퓨터공학과 박사수료

1984년~1996년 한국전기연구원 선임연구원

1996년~현 재 가야대학교 교수

관심분야 : RFID 및 USN 응용, 무선 프로토콜, Embedded System

안 광 선

e-mail : gsahn@knu.ac.kr
1972년 연세대학교 전기공학과(학사)
1975년 연세대학교 전기공학과(공학석사)
1975년~1976년 스페리유니벡 근무
1980년 연세대학교 전기공학과(공학박사)
1976년~현 재 경북대학교 컴퓨터공학과 교수
관심분야 : 유비쿼터스, 센서네트워크, RFID, Embedded System

이 성 준

e-mail : imggaibi@keri.re.kr
1997년 경일대학교 컴퓨터공학과(학사)
1999년 경북대학교 컴퓨터공학과(공학석사)
2007년 경북대학교 컴퓨터공학과(공학박사)
2008년~2009년 경북대학교 초빙교수
2009년~현 재 한국전기연구원 선임연구원
관심분야 : 유비쿼터스, 스마트 홈, 센서네트워크, RFID, Embedded System