충돌 비트 위치를 활용한 RFID 다중 태그 인식 알고리즘

준회원 이 현 지*, 정회원 김 종 덕**

A New RFID Tag Anti-Collision Algorithm Using Collision-Bit Positioning

Hyun-Ji Lee* *Associate Member*, Jong-Deok Kim** *Regular Member*

요 약

RFID 다중 태그 인식이란 하나의 RFID 리더 영역 내에 있는 다수의 태그를 태그 간의 통신 간섭에 의한 충돌을 피하며 고속으로 인식하는 기술이다. 다중 태그 인식 기술은 RFID 시스템의 성능과 안정성을 결정하는 핵심 기술로 중요성이 높다. 논문은 대표적 다중 태그 인식 기술인 QT(Query Tree) 알고리즘의 충돌 비트 위치를 활용하여 개선된 QT-CBP(Query Tree with Collision-Bit Positioning) 알고리즘을 제안한다. 시뮬레이션을 통한 검증 결과 QT-CBP는 태그 정보 비트의 중복성이 높고, 태그 수가 많을 때 QT에 비해 뚜렷한 성능 개선이 있었다.

Key Words : RFID, Anti-Collision, Query Tree, Bit position

ABSTRACT

RFID Anti-Collision technique is needed to avoid collision problem caused by Radio interference between tags in the same RFID Reader area. It affects the performance and reliability of the RFID System. This paper propose the QT-CBP(Query Tree with Collision-Bit Positioning) Algorithm based on the QT(Query Tree) algorithm. QT-CBP Algorithm use precise collision bit position to improve the performance. We demonstrated the proposed algorithm by simulation. Our algorithm outperformed when each tag bit streams are the more duplicate and the number of tags is increased, compared with QT.

Ⅰ. 서 론

RFID는 무선 전파 기술을 이용하여 리더로 하여금 태그와의 접촉 없이도 태그가 가진 고유 정보, 즉 태그 ID를 인식케 하는 기술이다. 정보 처리의 대상이 되는 사물에 고유 태그 ID를 가진 RFID 태그를 부착하여 사물의 움직임과 정보의 흐름을 보다 빠르고 효율적으로 처리하게 하는 것이 RFID 시스템이다. RFID 시스템에서 사용하는 무선 주파수는 활용 분야에 따라 다르며 주로 13.56MHz(ISO 18000-3), 433.92MHz(ISO 18000-7), 860~960 MHz (ISO 18000-6), 2.45GHz(ISO 18000-4) 등을 사용하고 있다. 무선 기술의 사용은 편리한 인식 과정 및 넓은 인식 거리 등 다양한 장점을 가지지만 이종 무선통신과의 간섭, 다중 리더 환경에서의 리더 간 충돌, 그리고 다중 태그 환경에서의 태그 간 충돌 등 시스템 성능에 제약을 주는 여러 가지 문제를 발생시킨다. 그 중 가장 보편적으로 발생하는 태그 간 충돌은 하나의 리더 내에 있는 다수의 태그들이 통신 과정에서 서로 간섭을 일으키는 현상을 의미하며 RFID 시스템의 성능과 안정성에 결정적인 영향을 미친다. RFID 다중 태그 인식 기술은 이

※ "이 논문은 교육인적자원부 지방연구중심대학육성사업(차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음".
* 부산대학교 컴퓨터공학과 이동통신연구실(eastleap@pusan.ac.kr), ** 부산대학교 컴퓨터공학과 이동통신연구실 (kimjd@pusan.ac.kr)
논문번호 : KICS2005-12-497, 접수일자 : 2005년 12월 19일, 최종논문접수일자 : 2006년 3월 26일

러한 충돌 현상을 극복하여 다수의 태그를 고속으로 인식하는 기술이다. 다중 태그 인식 기술은 충돌 대처가 핵심으로 Anti Collision 기술로도 불린다.

논문은 대표적 다중 태그 인식 기술인 QT(Query Tree)[1]의 성능을 개선한 QT-CBP를 제안한다. QT-CBP는 태그 간 충돌 발생 시 단순 충돌 여부만을 판단하는 QT와는 달리 충돌이 발생한 비트의 개수와 위치 등 보다 세밀한 정보를 추출하고 이를 활용하여 인식 성능을 개선하는 방안이다. QT-CBP는 태그 동작 방식에는 변경 없이 리더 변경만으로 구현 가능한 알고리즘이다.

EPCglobal 등의 표준 기구에서 정의한 태그 ID는 <제조사, 제품, 일련번호>와 같은 계층적 구조를 가지며 이에 따라 유사한 속성을 가진 사물들은 태그 ID의 비트 중복성이 높을 개연성이 크다. 이는 QT-CBP의 효용 가능성이 높음을 의미한다. 시뮬레이션을 통한 검증 결과 QT-CBP는 태그 ID 간의 비트 중복성이 높을수록 좋은 성능을 보였다.

논문의 구성은 다음과 같다. 2장에서는 기존의 주요 다중 태그 인식 기술들을 분류하고 그 중 대표적 메모리리스 다중 태그 인식 기술인 QT 알고리즘의 동작 방식을 소개한다. 3장에서는 제안하는 QT-CBP의 동작 방식을 소개하고, 4장에서는 시뮬레이션을 이용하여 QT-CBP의 성능을 검증한다. 마지막 5장에서 논문의 결론을 내린다.

Ⅱ. RFID 다중 태그 인식 기술

2.1 RFID 다중 태그 인식 기술의 분류

다중 태그 인식 기술은 태그 ID 전송 방식, 태그 비트 수 또는 태그 내 메모리 유무 등을 기준으로 분류할 수 있다. 기술 분류의 대표적 기준은 태그 ID 전송 방식으로 크게 태그 전송이 예측 가능한 트리 기반 알고리즘과 태그 전송이 임의의 확률에 근거하는 Slotted ALOHA 알고리즘으로 나뉜다. 트리 기반 알고리즘은 태그 ID를 이용하여 이진트리를 구성하고 순회하면서 태그들을 차례로 인식하는 것이다. Slotted ALOHA 알고리즘은 태그가 자신의 ID를 전송할 슬롯을 임의의 확률로 선택한 후 일괄적으로 전송하는 것이다. 이들은 다시 태그 내 메모리의 필요 여부에 따라 메모리형 알고리즘[2]과 메모리리스형[3] 알고리즘으로 분류할 수 있다. 메모리형 알고리즘은 태그가 일정한 크기의 메모리를 가지고 있어서 태그 인식에 필요한 갖가지 태그 상태 정보들을 저장하는 형태이다. 반면 메모리리스형 알고리즘은 단지 리더의 질의에 의해서만 태그 응답이 결정되는 알고리즘으로써 태그 상태 정보를 저장하는 메모리형 알고리즘에 비해 태그 동작이 단조롭다는 특징을 지닌다. Slot-Aloha 알고리즘은 알고리즘 진행 방식에 따라 슬롯에 전송하는 태그의 정보가 태그 ID 전체인지 아니면 태그 ID 중 한 비트를 전송하는 방식 따라 ID-슬롯 방식[4]과 bit-슬롯 방식[5]으로 구분하기도 한다. 표 1은 트리 기반 알고리즘과 Slot-Aloha 알고리즘을 메모리 여부에 따라 구분한 것이다.[6]

표 1. 다중 태그 인식 알고리즘의 분류

Anti-Collision Algorithm		
Tree-based Algorithm	Memory	Splitting tree Bit-arbitration
	Memoryless	Tree-walking Query tree Collision tracking tree
Slot-Aloha Algorithm	Memory	Bit-Slot
	memoryless	I-Code STAC Bit-by-bit binary Tree Basic Slotted binary Tree

2.2 QT 알고리즘

트리 기반 메모리리스 알고리즘 중 대표적이라고 할 수 있는 QT 알고리즘은 태그 응답에 따라 리더가 전송하는 질의가 결정되는 특징이 있다. 먼저 리더는 질의가 저장된 큐에서 k-비트 길이의 질의를 가져와 태그에게 브로드캐스트한다. 각각의 태그들은 리더로부터 전송받은 질의를 자신의 태그 ID와 비트 순서대로 비교한다. 만약 자신의 태그 ID와 수신한 질의의 모든 비트가 일치하는 경우 자신의 태그 ID를 리더에게 전송하고, 일치하지 않으면 전송하지 않는다.

리더는 질의 후 태그의 응답에 따라 세 가지 상태로 나뉘는데, 응답을 받지 못하거나 또는 하나의 태그로부터 응답을 받은 경우, 그리고 하나 이상의 태그로부터 응답을 받은 경우이다. 먼저 태그로부터 아무런 응답을 받지 못하였을 때에는 질의가 저장된 큐에서 다른 질의를 가져와 다시 태그에게 질의한다. 리더가 하나의 태그 응답을 받은 경우는 태그를 하나 인식한 것이 되므로 응답받은 태그 ID를 메모리에 저장한다. 그 후 다른 태그를 식별하기 위하여 알고리즘을 반복 수행한다. 둘 이상의 태그로부터 응답을 받으면 리더는 충돌이 발생하였음을 인지하고 이전에 태그에게 전송했던 질의에 0과 1

을 추가하여 새로운 질의를 만든 후 이를 큐에 저장하고 알고리즘을 다시 수행한다.

알고리즘의 수행은 큐에 들어있는 모든 질의를 수행한 후 종료한다. QT 알고리즘에서 큐는 비어있는 상태로 시작하는데 이 경우 리더는 태그에게 null에 해당하는 ε라는 질의를 전송한다. 충돌 발생 시 이전 질의에 0 또는 1을 추가하여 새로운 질의를 만들기 때문에 알고리즘을 수행함에 있어서 충돌 여부에 따라 질의의 길이가 점차 증가하는 특성을 가진다.

표 2. QT 알고리즘의 Pseudo Code

The QT Protocol
Reader has a query queue **Q** and a TagId memory **M**.
Let ω_k be the k'th bits of a bit string

Reader
```
Begin
 Initially Q = < ε >, M = < >
 while(queue is not empty)
   Pop a query q from Q;
   Broadcast q;
   Switch (response result)
     Case "only one response":
        Save the responded tagID r in M
        Break;
     Case "more than one response":
        Add two new queries q1, q2 to Q
          q1=q0, q2=q1
        Break;
     Case "no response": // Do nothing
        Break;
  end while
end
```

Tag
```
Has a TagID r = ω1ω2ω3...ωtagID_Length
begin
    Wait (query q from the reader)
    if (q=ε or q=ω1ω2ω3...ωquery_Length)
       send r to the reader
end
```

QT 알고리즘의 동작과정을 표 2에서 슈도코드로 표현하였다. 리더의 질의는 큐(Q)에 저장되고 인식된 태그ID는 메모리 M에 저장된다. q는 리더의 질의를 뜻하고 태그 응답에 따라 새로 생성된 질의를 각각 q_1, q_2라 하였다. 태그ID r은 $\omega_1\omega_2\omega_3...\omega_{tagID_Length}$로 나타낼 수 있는데 여기서 ω_k는 태그ID의 k번째 비트로써 0 또는 1의 값을 갖는다. tagID_Length 는 태그ID의 길이를 나타낸다.

Reader	ε	0	1	00	01	10	11	000	001
response	collision	collision	collision	collision	No response	101	110	000	001
Tag1 (000)	000	000		000				000	
Tag2 (001)	001	001		001					001
Tag3 (101)	101		101			101			
Tag4 (110)	110		110				110		
Q={ε}	0 1	1 00 01	00 01 10 11	01 10 11 000 001	10 11 000 001	11 000 001	000 001	001	
Memory M						101	101 110	101 110 000	101 110 000 001

그림 1. QT 알고리즘의 동작 예

그림 1은 태그 ID가 000, 001, 101, 110 인 4개의 태그에 대해 QT 알고리즘을 적용한 경우 진행 과정을 도시한 것이다. 가장 윗줄은 리더가 태그에게 전송하는 질의를 나타내고 해당 질의의 세로줄은 리더가 질의를 보내고 난 후 태그의 응답과 충돌 여부, 큐(Q)에 남아있는 질의 그리고 인식된 태그ID가 저장되는 메모리(M)의 상태를 나타낸다. 먼저 알고리즘이 시작할 때에는 큐에 아무런 질의가 없으므로 리더는 null에 해당하는 ε를 브로드캐스트하고 모든 태그는 ε를 받은 후 자신의 태그ID를 리더에게 전송한다. 이 때 4개의 태그가 동시에 응답하였으므로 충돌이 발생하고 리더는 이전의 질의인 ε에 0과 1을 추가한 0, 1의 새로운 질의를 생성하여 큐에 저장한다. 마찬가지로 두 번째 단계에서 리더는 큐에서 0을 가져와 태그에게 전송하고 첫 비트가 0인 Tag1과 Tag2가 응답하여 또 다시 충돌이 발생한다. 이와 같이 QT 알고리즘을 이용하면 4개의 태그(000, 001, 101, 110)를 인식하는데 9번의 알고리즘 반복이 필요하다는 것을 알 수 있다.

Ⅲ. QT-CBP 알고리즘

3.1 상세 충돌 정보의 도출

QT 알고리즘은 수신한 태그 ID 중 어느 한 비트라도 충돌이 발생하면 전체 알고리즘에서 충돌이 발생한다고 인지한다. 앞서 살펴본 QT 알고리즘의 예제를 살펴보면 단계 2에서 '0' 질의를 보냈을 때 태그1과 태그2가 동시에 응답하여 충돌이 발생함을

알 수 있다. 그리고 단계 4에서 '00' 질의를 보냈을 때 또한 태그1과 태그2가 동시에 응답하여 충돌이 발생하였다. 1 비트 차이가 나는 두 태그를 인식하기 위하여 이미 2번의 질의를 보냈음에도 불구하고 태그 인식을 위해 다시 '000' 과 '001' 의 질의를 보내고 난 후에 태그1과 태그2를 인식하는 것을 살펴볼 수 있다. QT 알고리즘이 위와 같은 비효율적인 태그 인식 과정을 거치는 요인 중의 하나는 전체 태그 ID 중에서 단 1비트만 충돌이 발생하더라도 이를 단순 충돌로 인식하기 때문이다. 즉 1비트에서 충돌이 발생하던지 아니면 96비트 전체에서 충돌이 발생하던지 간에 리더는 단순히 이를 충돌로 인식하고 새로운 질의를 생성하게 되는 것이다. 본 논문에서 제안하는 QT-CBP 알고리즘은 이러한 문제점을 해결하고 개선함으로써 리더의 반복적인 알고리즘 수행을 방지하고자 한다.

실제 리더는 적절한 비트 코딩 기법을 사용함으로써 태그로부터 전송되는 태그 ID의 비트 구분이 가능하다. 일반적으로 널리 알려진 결정적 기법인 Binary Tree 충돌 방지 알고리즘의 경우 충돌이 발생하는 정확한 비트 위치를 리더에서 인지할 수 있음을 전제하고 있다. 예를 들면 EPCglobal에서 발표한 Class0의 anti-collision 기법인 Bit-by-Bit 이진 트리 알고리즘은 두 sub-carrier tone(Data '0' for 2.2MHz, Data '1' for 3.3MHz)을 사용함으로써 각 비트를 구별하고 있다.[7] 이는 이미 많은 충돌방지 알고리즘에서 데이터 충돌이 일어나는 정확한 비트 위치를 이용하고 있다는 것을 의미한다.

만일 태그 ID 내에서 다수의 충돌 비트가 넓은 범위에 분포한다면 충돌 비트 위치를 인식할 때 마다 처리하는 알고리즘은 오히려 QT 알고리즘에 비해 성능이 떨어질 염려가 있다. 그러나 물류 RFID에서 사용하는 Auto ID의 EPC 태그 데이터 구조를 살펴 볼 때 다수의 태그를 동시에 인식하는 경우 각 태그들이 동일한 EPC Manager나 Object Class 값을 가질 확률이 아주 높다. Auto ID의 EPC 태그 데이터(96 비트)는 Header(8 비트), EPC Manager(28 비트), Object Class(24 비트), Serial Number(36 비트) 등 4개의 필드를 가지는 구조로 이루어지는데 물류 RFID의 특성 상 Serial Number 즉 마지막 36 비트만 다른 태그들이 인식될 가능성이 높다는 것이다. 이를 바탕으로 살펴볼 때 QT 알고리즘의 태그 ID의 좁은 범위의 하위 몇 비트에서 발생하는 충돌을 처리하기 위해 트리를 다시 순회하는 방식은 알고리즘의 성능에 있어서 상당히 비효율적이다.

3.2 QT-CBP 알고리즘의 동작 방식

따라서 QT-CBP 알고리즘에서는 리더가 정확한 비트 위치를 알 수 있다는 점, 그리고 실제 충돌은 태그 ID 일부에서 발생할 확률이 크다는 특징을 반영하여 충돌이 일어난 비트를 처리하는 알고리즘을 수행함으로써 불필요한 태그 인식 단계를 줄이고자 하였다.[8]

표 3는 QT-CBP 알고리즘의 슈도코드를 나타낸 것이다. S는 리더의 질의(q)가 저장되는 스택을 나타낸 것으로 초기에는 ε가 들어있다고 가정한다. M

표 3. QT-CBP 알고리즘의 Pseudo Code

The QT-CBP Protocol
Reader has a query stack **S** and a TagId memory **M**.
Let ω_k be the k'th bits of a bit string

Reader
```
Begin
 Initially S = < ε >, M = < >
 while(stack is not empty)
   Pop a query q from S;
   Broadcast q;
   Switch (response result)
     Case "only one response":
       Save the responded tagID r in M
       Break;
     Case "more than one response":
       Get the aggregated response R
        R = ω1ω2ω3...ωk-1X... , X ->collision bit
       count the collision(X) bits -> Nc
       resolve the position of the first
        colliding bit->k
       If (Nc = 1)
         Get two new tagIDs r1, r2 from R
        r1=ω1ω2ω3...ωk-10..., r2=ω1ω2ω3...ωk-11...
          save r1, r2 in M
       else
         Push two new queries q1, q2 to S
           q1=ω1ω2ω3...ωk-10, q2=ω1ω2ω3...ωk-11
       Break;
     Case "no response": // Do nothing
       Break;
  end while
end
```

Tag
```
Has a TagID r = ω1ω2ω3...ωtagID_Length
begin
    Wait (query q from the reader)
    if (q=ε or q=ω1ω2ω3...ωquery_Length)
       send r to the reader
end
```

은 인식된 태그 ID를 저장하는 메모리 공간을 의미한다. 리더의 동작은 스택에 저장된 ε를 태그에게 브로드캐스트 하는 것으로 시작된다. 질의 후 리더는 태그의 응답에 따라 세 가지 경우로 나뉜다. 하나의 태그 응답을 전송받은 경우 해당 태그 ID(r)을 메모리에 저장하고 하나 이상의 응답을 받은 경우에는 충돌이 발생한 비트 수(Nc)와 처음 충돌이 발생한 비트의 위치(k)를 저장한다. 만일 충돌이 발생한 비트의 수가 1개이면 해당 충돌 비트에 각각 0과 1의 값을 설정한 뒤 이를 메모리에 저장한다. 두 개 이상의 충돌 비트가 감지된 경우 이미 인식된 k-1번째 비트까지의 질의에 각각 0과 1을 추가하여 질의를 생성한 다음 스택에 저장한다. 태그 응답이 없는 경우 아무런 동작을 하지 않고 다시 위의 알고리즘을 반복한다. 태그는 총 비트 수가 tagID_Length인 태그ID를 가지고 리더의 질의가 도착하면 질의와 태그ID를 비교 한 뒤 응답을 전송한다.

QT-CBP는 태그 ID 중 하나의 비트에서 충돌이 발생한 경우 QT와는 달리 이를 다시 처리하지 않고 해당 비트만 다른 두 개의 태그가 있는 것으로 인식하여 이를 메모리에 저장한다는 것에 차이가 있다. 또한 큐(Q)를 사용하지 않고 스택(S)을 사용하여 질의를 저장함으로써 세로방향으로 트리를 순회한다는 것도 차이점이라 할 수 있다.

그림 2는 그림 1에서 QT에 적용된 예제 태그를 QT-CBP에 적용한 알고리즘의 동작 과정을 나타낸 것이다. 리더의 질의가 가장 위쪽 줄에 나타나 있고 질의 후 태그 응답 및 충돌 감지 상태 그리고 스택 및 메모리의 상태에 대하여 세로줄로 나타내었다. 하나 이상의 태그 응답으로 인하여 충돌이 발생한 경우 충돌이 발생한 비트의 위치를 X로 표기하였다. QT-CBP 알고리즘에 따라 단계 5에서 태그1과 태그2가 동시에 응답하여 충돌이 발생하였지만, 태그ID의 마지막 비트에서 충돌이 발생하였음을 감지하고 마지막 비트에 각각 0과 1을 추가하여 태그ID를 인식하였다. 따라서 이전 QT 알고리즘을 적용하였을 때 단계 수가 9 인데 비해 QT-CBP 알고리즘을 적용하면 4번의 단계가 줄어서 총 5번의 단계로 4개의 태그를 인식할 수 있음을 알 수 있다.

Reader	ε	1	11	10	0
response	collision (XXX)	collision (1XX)	110	101	(00X) 000 001
Tag1 (000)	000				000
Tag1 (001)	001				001
Tag3 (101)	101	101		101	
Tag4 (110)	110	110	110		
S={ε}	0 1	0 10 11	0 10	0	
Memory M			110	110 101	101 110 000 001

그림 2. QT-CBP 알고리즘의 동작 예

Ⅳ. 성능 분석

4.1 태그 샘플링

성능 분석을 위한 샘플 태그는 EPCglobal에서 제안한 TDS 1.3v의 표준을 따르는 태그를 선정하였다. 태그의 길이는 총 96비트로 인코딩 방식을 결정하는 Header, EPCglobal에서 부여하는 General Manager Number, 상품 코드인 Object Class와 각 Object Class에 유일한 값을 부여하는 Serial Number로 구성된다.[9] (표 4)

표 4. EPCglobal GID 태그 구조

	Header	General Manager Number	Object Class	Serial Number
GID-96	8	28	24	36
	0011 0101 (Binary value)	268,435,455 (Max. decimal value)	16,777,215 (Max. decimal value)	68,719,476,735 (Max. decimal value)

일반적으로 RFID 기술이 널리 사용되는 물류 산업에서는 표 3과 같은 태그 구조를 이용하여 각 상품마다 고유한 값을 부여한다. 동일한 회사일 경우 각 상품은 같은 General Manager Number 값을 가지고, 만일 동일한 회사의 같은 품목일 경우에는 General Manager Number 뿐만 아니라 Object Class 필드 값이 같다. 이와 같이 물류 분야에서 RFID를 적용했을 때 나타나는 특정상황은 각 태그의 일정한 비트가 중복될 가능성이 크다는 것이고 이는 곧 각 태그 ID의 중복성이 높아짐을 의미한다.

본 논문에서는 이러한 상황을 가정하여 시뮬레이션에 반영하였다. 96비트 태그 ID 전체가 랜덤한 값일 수도 있지만 상위 필드는 같고 하위 몇 비트만 다른 태그가 한꺼번에 인식될 가능성이 높다고 가정하였다. 이를 토대로 충돌이 발생하는 비트 범

위가 한정적인 태그로 시뮬레이션을 수행하고 이를 분석하였다.

QT 알고리즘과 제안한 QT-CBP 알고리즘의 성능 측정 기준으로 리더가 전송하는 총 질의 비트 수와 태그 인식 과정의 질의-응답 단계 수를 측정하였다. 리더의 총 질의 비트 수는 전체 태그 인식 단계에서 리더가 태그에게 전송한 질의의 전체 비트 수를 계산한 것이고, 질의-응답 단계 수는 리더가 질의를 전송하고 태그가 그에 대한 응답을 보내는 총 인식 단계 수를 가리킨다. QT-CBP 알고리즘에서는 충돌 비트에 대한 처리를 수행함으로써 인식 단계를 줄여 전체 인식 시간을 감소시킨다. 인식 단계가 줄어듦에 따라 리더가 질의를 전송하는 횟수도 달라지는데 이것은 리더의 질의는 전체 태그 인식 시간을 결정하는 주요한 요인이 된다. QT-CBP에서 태그의 응답은 태그ID로 고정적이지만 리더의 질의는 태그 응답에 따라 가변적이기 때문에 QT와 비교할 때 중요한 척도가 될 수 있다.

시뮬레이션에 사용한 태그는 EPCglobal TDS 1.3v의 GID로 인코딩 된 96비트 태그를 사용하였다. 태그의 수를 100개에서 시작하여 1000개까지 증가시키면서 각 알고리즘을 수행하였다. Object Class 및 Serial Number는 무작위 추출하였고 시뮬레이션 결과는 각 태그 개수 당 한 번씩 측정된 값을 사용하였다.

4.2 충돌이 발생하는 범위가 한정적인 태그 ID

실제 물류 RFID에서 발생하는 상황에 맞추어 충돌이 발생하는 범위가 한정적인 태그 ID인 경우에 대한 시뮬레이션을 수행하였다. 또한 단지 한정적 범위 내 충돌이 발생한다는 가정뿐만 아니라 EPCglobal GID 인코딩 태그의 필드에 따라 모두 3가지 상황으로 나누어 QT-CBP 알고리즘과 QT 알고리즘을 측정하고 이를 토대로 두 알고리즘을 비교하였다.

4.2.1 상황 1 : 5가지 종류의 Object Class

물류 RFID에서 Object Class는 품목에 대한 고유한 번호를 의미한다. 즉 5가지 다른 Object Class를 갖는 태그는 서로 품목이 다른 5가지 물품을 의미한다. 5가지 종류의 Object Class를 가정하고 각 Object Class마다 순차적으로 증가하는 Serial Number를 가지는 태그를 샘플링하여 각 알고리즘에 대한 성능을 측정하였다.

태그 샘플링의 예는 다음과 같다. 각 태그ID 중에서 밑줄 그은 부분은 Object Class에 해당하는 부분이고 그 외 나머지 비트 부분은 Serial Number에 해당하는 부분이다. Tag1과 Tag2 그리고 Tag3은 동일한 종류의 Object Class를 가지지만 하위 Serial Number가 순차적인 태그이다. 또한 Tag4와 Tag5도 서로 같은 Object Class 값을 가지는 태그이다.

Tag1 : 11001010011110110100 1
Tag2 : 11001010011110110101 0
Tag3 : 11001010011110110101 1
Tag4 : 00101110101101100101 0
Tag5 : 00101110101101100101 1 ...

그림 3과 그림 4에서 5가지 종류의 Object Class를 가지는 태그에 대한 QT 알고리즘과 QT-CBP 알고리즘을 적용한 결과를 살펴볼 수 있다. 성능 측정 결과를 살펴볼 때 QT-CBP 알고리즘을 수행한 결과가 QT 알고리즘을 수행한 결과에 비해 낮은 단계 수와 적은 리더 질의 비트 수를 가지는 것으로 나타났다. 또한 단계 수 비교에서는 태그의 수가 클수록 차이 간격이 넓어지는 것을 알 수 있다.

5가지 종류의 Object Class를 사용한다는 것은 총 100개의 96비트 태그 중에서 상위 60비트의 값이 동일한 태그가 20개씩 5종류가 있다는 것을 의미한다. 그리고 각 Object Class에 대하여 하위 36비트 값은 순차적인 값을 가짐으로써 36비트 값 중에서도 상당 부분이 일치하는 값을 가질뿐더러 1비트의 차이만을 가지는 다수의 태그가 존재한다. 시뮬레이션 결과(그림 3)를 살펴보면 100개의 태그를 인식할 때 QT는 총 659단계가 소요되지만 QT-CBP는 총 530번의 단계를 거치는 것을 알 수 있

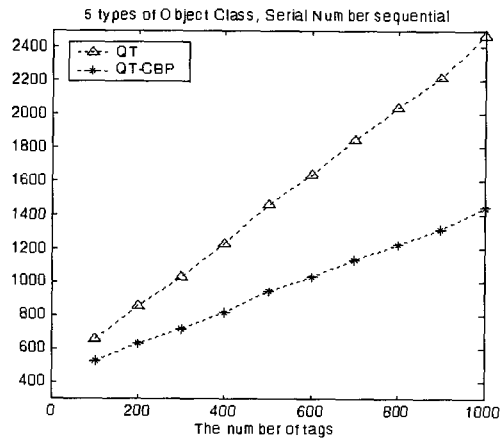

그림 3. 다양한 Object Class를 가지는 태그의 단계 수

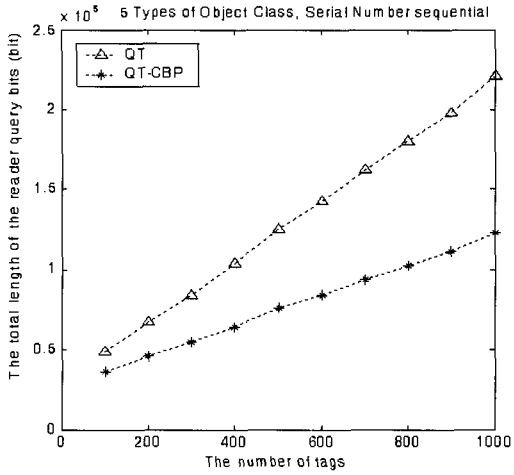

그림 4. 다양한 Object Class를 가지는 리더의 질의 비트 수

다. 태그의 개수가 늘어날수록 총 태그 인식 단계 수 차이는 점점 벌어져서 1000개의 태그를 인식하는데 QT는 2471번, QT-CBP는 1440번의 단계가 소요된다(그림 4). 이것은 태그의 개수가 늘어나면 동일한 Object Class를 갖는 태그가 늘어나므로 일정 범위가 동일한 비트를 갖는 태그가 더 많아지기 때문이다.

QT와 QT-CBP는 태그 응답 길이는 모두 태그ID로 일정하다. 따라서 인식 시간에 영향을 미치는 요인으로 리더 질의 길이를 따질 수 있다. 인식 단계가 줄어듦에 따라 불필요한 질의를 전송할 필요가 없기 때문에 그림 4에서와 같이 QT-CBP 에서의 리더 질의 비트가 QT에 비하여 적은 전송양을 가지는 것을 알 수 있다. 실제 QT-CBP 알고리즘은 QT에 비하여 34.48퍼센트 정도 적은 리더 질의를 전송하는 것으로 나타났다. 단계 수를 줄임으로써 전체 리더가 전송하는 질의 양을 줄여 전체 인식 과정의 시간 단축을 기대할 수 있는 것이다.

4.2.2 상황 2 : random Serial Number

이번 장에서는 하위 36비트 Serial Number만 다르고 나머지 필드 값은 같은 값을 가지는 다수의 태그를 가정하였다. 하위 36비트 값은 랜덤한 값을 생성하여 측정하였다. 물품을 생산하는 공장에서는 같은 물품에 Serial Number만 다른 상품을 동시에 인식하게 되는 경우가 많고 이 때 충돌이 발생하는 비트의 수는 일반적인 경우보다 적을 수 있기 때문에 QT보다 QT-CBP 알고리즘의 효용성을 더욱 높이 측정할 수 있다.

상황 2의 시뮬레이션 결과는 상황 1에 비하여 더

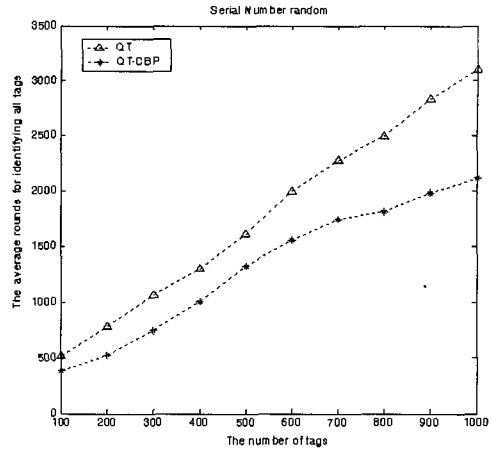

그림 5. Serial Number가 랜덤인 태그 단계 수

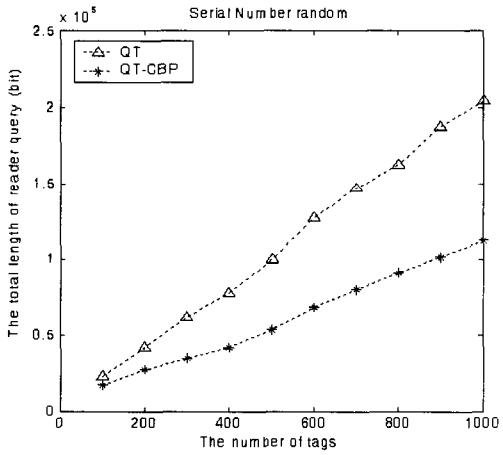

그림 6. Serial Number가 랜덤인 태그의 리더 질의 비트 수

많은 태그 인식 단계를 보인다. 이것은 앞서 상황 1이 5가지 종류의 Object Class이지만 하위 36비트의 Serial Number가 순차적인데 비하여 상황 2는 하나의 Object Class를 갖는다. 따라서 Serial Number가 랜덤한 값을 가지므로 전체 태그ID에서 동일한 비트가 차지하는 비율이 상황 1에 비하여 작기 때문에 태그 인식에 더 많은 단계를 거치게 된다. 하지만 상위 60비트가 동일한 Object Class 값이기 때문에 QT-CBP의 결과가 QT에 비하여 더 놓은 효율을 보이고 있다. QT-CBP에서 리더가 전송하는 질의 비트 수는 QT에 비하여 42퍼센트에 해당하는 감소율을 나타낸다.

4.2.3 상황 3 : Serial Number sequential

일반적으로 어느 한 공장에서 생산되어 출고되는 물품의 태그는 같은 Object Class를 가지면서 Serial

Number가 연속적인 값을 가지는 경우가 많다. 이러한 상황을 설정하여 Header, General Manager Number, Object Class 필드 값은 같고 Serial Number의 값이 연속적으로 증가하는 다수의 태그를 사용하여 시뮬레이션을 수행하였다.

전체 태그 ID 중 Serial Number인 하위 36비트 값을 순차적인 값을 갖는 태그를 생성하여 성능 측정을 한 결과 이미 언급된 두 상황에 비해 QT 알고리즘과 QT-CBP 알고리즘의 성능 차이가 가장 큰 것을 알 수 있다. (그림 7 그림 8) 이는 순차적인 Serial Number를 갖게 되는 경우 태그 ID의 대부분 비트가 동일하고 Serial Number 중에서도 하위 일부분만 다른 태그들이 생성되기 때문에 다른 상황에 비해 상대적으로 충돌이 발생하는 비트 수가 적기 때문으로 해석할 수 있다. 또한 다른 상황의 결과와 마찬가지로 태그의 개수가 많아질수록 QT-CBP 알고리즘이 QT 알고리즘에 비하여 높은 성능을 보임을 알 수 있다.

그림 7은 QT와 QT-CBP의 태그 인식 단계 수를 나타낸다. 각 태그ID내에서 서로 동일한 비트를 가질 확률이 세 상황 중 가장 높기 때문에 전체적인 태그 인식 단계 수가 다른 상황보다 적은 것을 알 수 있다. 또한 다른 두 상황에 비해 QT와 QT-CBP의 태그 인식 단계 수의 차이가 약 44퍼센트로 가장 크게 나타난다. 그림 8에서는 리더가 전송한 질의 비트 수를 나타낸다. 이때 태그의 개수가 100개일 때에는 QT와 QT-CBP의 질의 비트 수는 별 차이가 없지만 태그의 개수가 커질수록 리더 질의 비트 수의 차이가 점점 벌어지게 되는데 이는 태그 인식 단계 수에 리더 질의 비트 수가 비례하는 것을 알 수 있다.

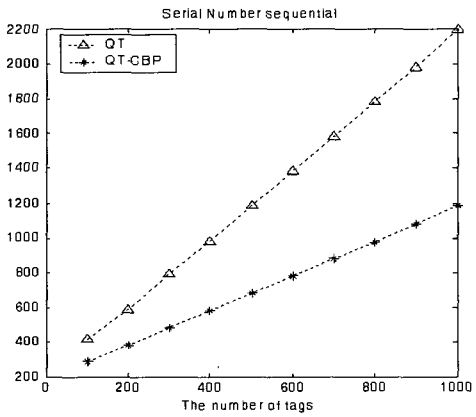

그림 7. Serial Number가 순차적인 태그의 단계 수

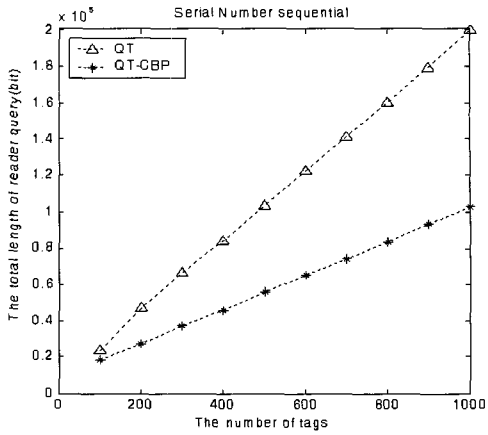

그림 8. Serial Number가 순차적인 태그의 리더 질의 비트 수

4.2.4 QT와 QT-CBP 비교

앞서 살펴본 세 가지 상황에 대한 시뮬레이션 결과를 토대로 QT 알고리즘과 QT-CBP 알고리즘을 비교해본다.

전체적인 시뮬레이션 결과는 QT-CBP 알고리즘이 QT 알고리즘에 비해 우수한 성능을 보임을 알 수 있다. 또한 태그의 개수가 증가할 수 록 QT-CBP 알고리즘의 성능이 더 우수한 것으로 나타났다. 이는 태그 수가 증가하면 QT에서 생성되는 불필요한 질의 수가 줄어들고 이에 따라 전체 태그 인식하기 위한 단계 수가 감소하기 때문이다. 평균적으로 상황 1의 경우 약 34.32퍼센트의 단계 감소를 보이고 상황 2의 경우에는 26.38퍼센트 그리고 상황 3에서는 약 41.20퍼센트의 단계 감소율을 보이고 있다. 즉 QT-CBP 알고리즘은 각 태그의 중복성에 따라서 절반에 가까운 단계 감소의 효과를 보인다. 또한 각 알고리즘의 리더 질의 비트 수를 비교해 볼 때 QT 알고리즘에 비하여 QT-CBP 알고리즘에서 전송되는 리더 질의 비트 양이 전체적으로 약 40퍼센트 정도 적은 것으로 나타났다.

각 상황에 대하여 Serial Number 값에 차이를 둔 상황 2과 상황 3을 비교해보면 점차적으로 증가하는 Serial Number를 가지는 태그 일 때 QT-CBP 알고리즘의 성능이 더 우수한 것으로 나타났다. 실제 RFID 시스템에서의 다중 태그 ID 값의 분포를 살펴볼 때 각 태그 ID 값은 일정한 구조와 규칙이 있다. 특히 물류 RFID 시스템에서는 일정한 Object Class 와 Serial Number의 필드 값을 갖는 다중의 태그들이 동시에 읽혀질 가능성이 더욱 크다. 따라서 높은 태그 비트의 중복성인 상황에서 QT 알고리즘에 비하여 QT-CBP 알고리즘은 더욱 높은 효

용성을 가지게 된다.

QT-CBP는 QT에 비하여 부가적인 저장 공간이 필요하지만 충돌 비트 위치를 저장하는 변수 하나와 충돌 비트 개수를 저장하는 변수 하나만 추가적으로 더 필요할 뿐이다. QT-CBP의 한계점이 있다면 각 리더들이 정확한 비트를 구별하면서 태그ID를 수신할 수 있는가에 있다. 이것은 각 리더의 종류 및 성능에 의한 것으로 비트 구별을 지원하지 못하는 리더는 QT-CBP 알고리즘을 사용할 수 없게 된다.

V. 결 론

논문은 대표적 다중 태그 인식 기술인 QT의 성능을 개선한 QT-CBP를 제안하였다. QT-CBP는 태그 간 충돌 발생 시 단순 충돌 여부만을 판단하는 QT와는 달리 충돌이 발생한 비트의 개수와 위치 등 보다 세밀한 정보를 추출하고 이를 활용하여 QT의 트리 순회 성능을 개선하는 방안이다. 충돌 비트의 개수가 적은 경우 리더가 전송하는 질의의 수를 줄이면서 정확한 충돌 비트 위치 및 개수를 판단하여 태그를 인식하는 방법을 모색하였다. QT-CBP는 태그 동작 방식에는 변경 없이 리더 변경만으로 구현 가능한 알고리즘이다. QT-CBP의 성능을 측정하기 위하여 EPCglobal의 표준 태그 ID 구조를 채택하였고, RFID 물류 시스템의 특정 상황을 적용하여 시뮬레이션을 수행하였다. 시뮬레이션을 통한 검증 결과 QT-CBP는 태그의 수가 많고 태그 ID 간의 비트 중복성이 높을수록 QT에 비해 우수한 성능을 보였다. 표준 태그 ID의 구조 및 물류 시스템과 같은 대표적 RFID 응용 환경의 특성을 고려할 때 태그 ID 간의 비트 중복성의 개연성은 크다고 할 수 있으며 이는 QT-CBP의 효용성이 높음을 의미한다.

참 고 문 헌

[1] Ching Law, Kayi Lee, and Kai-Yeung Sju, "Efficient Momoryless Protocol for Tag Identification", *In Proceedings of the 4th International Workshop on Discrete Algorithms and Metohds for Mobile Computing and Communication*, Pages 75-84. ACM. August 2000.

[2] Hush, Don R. and Wood, Cliff. Analysis of Tree Algorithms for RFID Arbitration *In IEEE International Symposium on Information Theory*, Pages 107-. IEEE, 1998.

[3] A. Juels, R. Rivest, and m. Szydlo. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. *Proceedings of the 10th ACM conference on Computer and communication security*, ISBM:1-58 113-738 -9, Pages 103-111. 2003.

[4] Vogt, H. Efficient Object Identification with Passive RFID Tags. In International Conference on Pervasive Computing, *LNCS*. 2002.

[5] Changsoon Kim, Kyunglang Park, Hiecheol Kim, Shindug Kim. An Efficient Stochastic Anti-collision Algorithm using Bit-Slot Mechanism. *PDP'2004*, July 2004.

[6] 권성호, 홍원기, 이용두, 김희철, "저비용 RFID 시스템에서의 충돌방지 알고리즘에 대한 성능평가", *한국통신학회논문지*, Vol. 30, No. 1B, January 2005.

[7] Auto-ID Center. Draft protocol specification for a 900MHz Class 0 Radio Frequency Identification Tag. *Auto-ID Center*. February 23, 2003.

[8] 차재룡, 김재헌, "Ubiquitous ID 시스템에서 고속 충돌 방지 알고리즘", *한국통신학회논문지*, Vol. 29, No. 8A, pp. 17-26. August 2004.

[9] Auto-ID Center. EPC™ Tag Data Standards Version 1.3. *Auto-ID Center*. 9 Sep, 2005.

이 현 지 (Hyun-Ji Lee) 준회원

2005년 2월 부산대학교 전자전기정보컴퓨터공학부 정보 컴퓨터공학과 졸업

2005년 3월~현재 부산대학교 컴퓨터공학과 석사

<관심분야> RFID, Sensor Network, MAC 프로토콜, IEEE 802.11/15

김 종 덕 (Jong-Deok Kim) 정회원

1994년 2월 서울대학교 계산통계학과 졸업

1996년 2월 서울대학교 전산과학과 석사

2003년 2월 서울대학교 컴퓨터공학과 박사

2004년 2월~현재 부산대학교 정보컴퓨터공학과, 조교수

부산대학교 컴퓨터및정보통신연구소 정보기술연구원

<관심분야> 무선통신, 이동통신망, RFID/USN