RFID 시스템에서 충돌비트 수를 활용한 개선된 충돌방지 알고리즘

정회원 김 명 환*, 국 광 호*

An Improved Anti-Collision Algorithm using the Number of Collision Bits in RFID System

Myeoung-Hwan Kim*, Kwang-Ho Kook* ***Regular Members***

요 약

RFID 기술은 RF 신호를 이용하여 물품에 부착된 전자태그의 고유 정보 즉 태그 ID를 식별하는 비접촉식 자동 인식기술이다. 본 논문은 RFID 기술의 핵심인 다중 태그 인식 알고리즘 중 QT(Query Tree) 알고리즘의 성능을 개선한 QT-DCBP(Query Tree - Double Collision Bit Partitioning) 알고리즘을 제안한다. QT-DCBP 알고리즘은 충돌된 비트의 수와 리더 영역 내에 존재하는 태그의 수에 기초한다. 충돌한 비트의 수가 3개 이상이고 태그로부터 수신한 모든 비트가 충돌하면 처음 충돌한 2개의 비트를 '00', '01', '10', '11'로 고정한 새로운 프리픽스를 만듦으로써 QT 알고리즘의 성능을 향상시킨다. QT-DCBP 기법의 성능을 평가하고자 시뮬레이션을 통해 QT 기반 기법들의 질의 응답 횟수를 비교하였다. 비교 결과 QT-DCBP 기법의 성능이 QT 기법의 성능보다는 30~50%, QT-CBP(Query Tree - Collision Bit Positioning) 기법의 성능보다는 10~20% 정도 향상됨을 볼 수 있다.

Key Words : RFID, Anti-Collision algorithm, Query-Tree, Collision Tracking, Multi-tag Identification

ABSTRACT

RFID enables the automatic identification of the electronic tag attached to an object without contact. This paper proposes a QT-DCBP(Query Tree - Double Collision Bit Partitioning) algorithm, which is a modified version of the Query Tree algorithm, as a multiple tag identification algorithm. It is based on the number of collided bits and the density of tags residing in the reader area. If the number of collided bits is greater than or equal to 3 and all the bits received from the tags are collided, the new prefix is made by fixing the first 2 collided bits as '00', '01', '10', '11'. The simulation results show that the proposed QT-DCBP's performance is 30~50% higher than that of QT algorithm and is 10~20% higher than that of QT-CBP algorithm.

Ⅰ. 서 론

RFID(Radio Frequency IDentification) 기술은 RF 신호 리더(reader)를 이용하여 물품에 부착된 전자태그(tag)의 고유정보 즉 태그 ID를 식별하는 비접촉방식의 자동 인식 기술이다[1]. 이러한 태그는 전원의 유·무에 따라 수동식(passive) 태그와 능동식(active) 태그로 분류된다. 능동식 태그는 태그 자체에 전원이 내장되어 있으나 수동식 태그는 내부에 자체 전원을 가지고 있지 않고 리더의 신호로부터 에너지를 공급받아 동작한다. RFID 무선 기술은 편리한 인식, 넓은 인식 거리 등 많은 장점을 가지고 있지만 태그의 저가

* 서울산업대학교 IT정책전문대학원 산업정보시스템공학과 통신시스템연구실(capiya@hanmil.net, khkook@snut.ac.kr)
논문번호 : KICS2010-05-201, 접수일자 : 2010년 5월 6일, 최종논문접수일자 : 2010년 9월 7일

격, 저전력, 초소형화 문제와 이종 무선통신과의 간섭, 다중 리더 환경에서의 리더 간 충돌, 그리고 다중 태그환경에서 태그 간 충돌 등의 여러 가지 문제를 발생시킨다. 특히 다중 태그 식별문제는 리더기의 식별영역 내에 여러 개의 태그가 존재할 때 태그의 정보를 충돌 없이 전송받아서 식별해야 하는 태그와 리더기 사이의 다대일 통신 문제로 정의되며 충돌방지(anti-collision) 알고리즘은 RFID 시스템에서 가장 핵심적인 기술이다.

충돌방지 알고리즘은 Tree-Based 알고리즘과 Slot-Aloha 기반 알고리즘으로 분류된다. Tree-Based 알고리즘은 결정형(Deterministic) 충돌방지 알고리즘으로 영역 내의 모든 태그를 인식할 수 있는 데, Tree-walking 알고리즘[2], QT(Query Tree) 알고리즘[3], Bit-arbitration 알고리즘[4] 등이 있다. Slot-Aloha 기반 알고리즘은 확률적 충돌방지 알고리즘으로 리더의 질의 영역 내의 태그들은 프레임을 구성하는 N개의 슬롯 중 태그의 정보를 전송할 슬롯을 임의로 선정한 후 선정된 슬롯에 태그 식별자를 전송한다. Slot-Aloha 기반 알고리즘은 인식 영역 내의 태그의 개수를 정확히 알 수 없으므로 태그의 개수에 적당한 프레임당 슬롯의 개수와 태그 인식 종료시점의 계산에 확률적 방법을 사용하기 때문에 영역 내의 모든 태그의 인식을 보장할 수 없다는 단점이 있다. 대표적인 알고리즘으로 I-Code 알고리즘[5]이 있다.

본 논문은 대표적 다중 태그 인식 기술인 QT 기법의 성능을 개선한 QT-DCBP(Query Tree - Double Collision Bit Partitioning) 알고리즘을 제안한다. QT-DCBP는 리더의 요청에 따라 태그들이 리더로 ID를 전송할 때 충돌된 비트의 수에 기초하는 기법으로 기존의 QT 알고리즘보다 좋은 성능을 보여준다. 이를 위해서는 복수 개의 태그들이 리더로 ID를 동시에 전송할 때, 태그 간 충돌이 발생한 비트들의 위치를 식별하는 것이 필요한 데, 이는 맨체스터 코딩 기법을 사용하면 구현이 가능하다[1,6-8]. 단 리더기는 인식을 완료한 태그 ID를 저장할 메모리 외에 태그로부터 수신한 비트 정보를 저장할 메모리가 추가로 필요하다.

논문의 구성은 다음과 같다. 2장에서는 기존의 다중 태그 인식 기술 을 QT 알고리즘을 중심으로 소개하고 기존 알고리즘의 문제점을 알아본다. 3장에서는 본 논문에서 제안하는 QT-DCBP 알고리즘을 설명하고 4장에서는 시뮬레이션을 통해 제안하는 QT-DCBP 알고리즘의 성능이 기존의 QT 알고리즘들의 성능보다 우수함을 보인다. 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. QT 기반 충돌방지 알고리즘

본 논문은 결정형 충돌방지 알고리즘인 Tree-Based 알고리즘 중 태그 자체에 메모리가 없는 수동형 태그에 적용될 수 있는 QT 알고리즘의 성능을 개선시키고자 한다.

먼저 QT 알고리즘과 그 성능을 개선시키기 위해 제안된 QT 기반 기법들을 살펴보면 다음과 같다.

2.1 QT(Query Tree) 알고리즘

리더가 d비트 길이의 프리픽스 $B_d(B=b_1b_2, ..., b_d)$를 태그에게 질의하면, 태그들은 자신의 식별자와 프리픽스를 비교하여 그 값이 동일하면 d+1비트부터 마지막 n비트까지의 값을 리더에게 전송한다. 이때 태그의 응답결과에 따라 다음과 같은 세 가지 경우가 발생할 수 있다. 첫째, 오직 하나의 태그만 응답을 하는 경우이다. 이 경우엔 하나의 태그를 성공적으로 인식한 것이 되므로 응답받은 태그를 메모리에 저장하고 큐에 저장된 새로운 프리픽스를 가져와 태그에게 질의한다. 둘째, 태그로부터의 응답이 하나도 없는 경우이다. 응답한 태그가 없는 경우엔 큐에 저장된 새로운 프리픽스를 가져와 태그에게 질의한다. 셋째, 다수 개의 태그가 동시에 응답하여 충돌이 발생하는 경우이다. 이때는 기존의 프리픽스 B_d에 '0'과 '1'을 추가하여 새로운 프리픽스 '$B_d(B=b_1b_2, ..., b_d)||0$' 와 '$B_d(B=b_1b_2, ..., b_d)||1$'를 생성한 후 큐에 저장하고, 큐로부터 새로운 프리픽스를 가져와 다음 질의를 계속한다. 이러한 과정은 프리픽스 큐가 비게 될 때까지 반복된다. 표 1.은 리더의 영역 내에 '010', '011', '101', '110'의 네 개의 Tag_ID가 있다고 할 때 QT 알고리즘의 동작 과정을 나타낸다.

표 1에서 첫 번째 질의는 프리픽스 값이 없는 경우로(ε) 모든 태그들이 응답을 하므로 출동이 발생한다. 충돌이 발생하므로 기존의 프리픽스(ε)에 '0'과 '1'을 추가하여 큐에 저장한다. 첫 번째 질의 실행 후 큐에 '0'과 '1'이 추가되어 있는 것을 볼 수 있다. 네 번째 질의에서 프리픽스는 '00'인데 영역 내에 '00'으로 시작하는 태그가 없으므로 응답하는 태그가 없음을 볼 수 있다. 다섯 번째 질의에서 리더는 프리픽스 '01'로 질의하고, 영역 내의 태그 중 '01'로 시작하는 '010', '011'이 응답하여 역시 충돌이 발생하므로, 기존의 프리픽스에 '0'과 '1'을 추가하여 큐에 저장함을 볼 수 있다. 여섯 번째 질의에서 프리픽스 '10'으로 질의한 경우 '101' 하나만이 응답을 하므로 인식되어 메모리에 저장된다. 이 과정을 프리픽스 큐가 비게 될 때까

표 1. QT 알고리즘

단계	1	2	3	4	5	6	7	8	9
Query Prefix	ε	0	1	00	01	10	11	010	011
Response	XXX	01X	1XX	no R	01X	101	110	010	
Tag_01 010	010	010			010			010	
Tag_02 011	011	011			011				011
Tag_03 101	100		101			101			
Tag_04 110	101		110				110		
Queue	0 1	1 00 01	00 01 10 11	01 10 11	10 11 010 011	11 010 011	010 011	011	
Memory						101	101 110	101 110 010	101 110 010 011

지 반복 수행한다. 표 1로부터 영역 내의 태그 네 개를 모두 인식하는데 총 9번의 질의가 수행됨을 볼 수 있다.

2.2 QT-CBP(Query Tree with Collision-Bit Positioning)

QT 알고리즘은 리더와 태그간의 질의-응답 과정에서 태그로부터 받은 정보로부터 단지 충돌 발생 여부만을 감지하기 때문에 충돌 감지(Collision Detecting) 방식이라 볼 수 있다. 충돌추적 QT 알고리즘은 QT 알고리즘의 성능을 개선하고자 제안되었는데, 태그들의 충돌이 발생하면 충돌한 비트의 위치를 알아내고 이를 이용하여 질의-응답 횟수와 전송 비트수(질의를 위해 전송되는 총 비트수)를 줄이고자 한다. 대표적으로 충돌추적(CT, Collision Tracking) QT 알고리즘[6]이 있으며 개선된 QT(IQT: Improved QT) 알고리즘[9]과 충돌 트리 기반 알고리즘[10] 등도 이에 속한다.

CT 알고리즘의 충돌인식방법에 대한 예로써 8비트 길이의 식별자를 갖는 4개의 태그 '01010000', '01010001', '01010100', '01010101'가 리더기의 질의에 대해 각자의 태그 식별자를 리더기로 동시에 전송한다고 하자. 충돌감지 알고리즘을 사용하는 경우에는 다수개의 태그가 동시에 응답하므로 단순히 충돌이 발생한 것으로 간주한다. 반면에 CT 알고리즘은 충돌이 발생한 위치를 추적하는 기법으로, 충돌이 발생한 모든 비트 즉 6번째 비트와 8번째 비트의 위치를 모두 추적하는 완전충돌추적(CCT: Complete Collision Tracking) 방식과 처음 충돌이 발생한 6번째 비트의 위치만 추적하는 처음충돌추적(FCT: First Collision Tracking) QT방식으로 구분된다.

구현이 보다 간단한 FCT QT 방식의 원리를 살펴보면 다음과 같다. 리더기가 d비트 길이의 프리픽스 $B_d(B=b_1b_2, ..., b_d)$를 인자로 태그에 질의하면 영역 내의 태그들은 수신한 프리픽스를 자신의 식별자와 비교하여 그 값이 동일하면 태그는 d+1비트부터 마지막 n비트까지의 값을 리더에게 순차적으로 전송한다. 리더기는 태그가 전송한 식별자 정보를 수신함과 동시에 수신된 비트의 충돌 발생 여부를 검사한다. 이때 모두 '0' 혹은 '1'이 수신되면 나머지 비트의 수신을 계속하고, '0'과 '1'이 동시에 수신되어 충돌이 발생하면 나머지 비트의 수신을 중지하고, 태그로 식별자 전송 중지 명령을 보낸다. 모든 비트에서 충돌 없이 식별자 정보의 마지막 비트까지 수신되면 한 개의 태그가 식별된다. 충돌이 발생할 경우, 기존 프리픽스에 단순히 '0'과 '1'을 추가하는 QT 알고리즘과는 달리 충돌이 발생하기 전까지 수신된 모든 비트들에 '0'과 '1'을 추가하여 새로운 프리픽스를 생성하고 이를 큐에 저장하여 다음 번 질의-응답과정에 프리픽스로 사용함으로써 QT에 비해 전송횟수를 줄일 수 있다. 표 2.는 리더의 영역 내에 '010', '011', '101', '110'의 네 개의 Tag_ID가 있다고 할 때 FCT QT 알고리즘의 동작 과정을 나타낸다.

표 2에서 보는 것처럼 FCT QT 알고리즘의 기본 원리는 QT 알고리즘과 같다. 차이점은 표 2의 두 번째 질의의 경우 프리픽스 '0'으로 질의하면 영역 내의 두 개의 태그 '010'과 '011'이 응답하여 충돌이 발생

표 2. FCT QT 알고리즘

단계	1	2	3	4	5	6	7
Query Prefix	ε	0	1	010	011	10	11
Response	XXX	01X	1XX	010	011	101	110
Tag_01 010	010	010		010			
Tag_02 011	011	011			011		
Tag_03 101	100		101			101	
Tag_04 110	101		110				110
Queue={ε}	0 1	1 010 011	010 011 10 11	011 10 11	10 11	11	
Memory				010	010 011	010 011 101	010 011 101 110

한다. 이 때 QT 알고리즘은 단순히 충돌만을 감지하여 전송한 프리픽스 '0'에 '0'과 '1'을 추가한 '00', '01'을 큐에 저장하고 큐에서 다음 프리픽스를 가져와 질의를 계속 수행한다. 반면 FCT QT 알고리즘은 프리픽스 '0'으로 질의 했을 때 처음 충돌이 발생할 때까지 한 비트씩 계속 수신하고 충돌이 발생하면 나머지 비트의 수신을 중지하고 태그에게 전송을 중지하도록 요청한다. 따라서 표 2에서와 같이 프리픽스 '0'으로 질의하면 '0'으로 시작하는 태그 중 '010'과 '011'이 응답하게 되고 첫 번째 수신한 비트와 두 번째 수신한 비트가 모두 '0', '1'로 동일하여 충돌이 발생하지 않고 세 번째 비트에서 충돌이 발생하므로 충돌이 발생하기 전까지의 비트열인 '01'에 '0'과 '1'을 추가한 '010', '011'을 큐에 추가한 다음 큐에서 다음 프리픽스를 가져와 질의를 계속한다. 리더는 같은 방법으로 프리픽스 큐가 비게 될 때까지 반복 수행하며 결과적으로 모든 태그를 인식하는데 총 7번의 질의를 수행하게 된다. 즉 QT 알고리즘을 사용할 때 필요한 총 9번의 질의보다 적은 질의를 필요로 함을 볼 수 있다.

QT-CBP 알고리즘은 CT알고리즘을 발전시킨 알고리즘으로 리더가 정확히 충돌이 발생한 비트의 위치를 알 수 있다고 할 때, 리더가 수신한 비트 중 하나의 비트에서만 충돌이 발생하는 경우에는 각 태그의 ID가 고유하므로 단지 2개의 태그만 전송되었다는 것을 알 수 있으므로 이를 고려하여 태그 인식에 필요한 질의 횟수를 줄이고자 하는 기법이다[11]. 리더의 동작은 QT 알고리즘과 같이 ε를 태그들에게 브로드캐스트 하는 것으로 시작된다. 질의 후 리더는 태그의 응답에 따라 다음 네 가지 형태중 하나로 대응한다. 첫째, 하나의 태그만 응답을 하는 경우 충돌이 발행하지 않았으므로 해당 태그를 메모리에 저장하고 다음 질의를 계속한다. 둘째, 하나 이상의 응답을 받은 경우 충돌한 비트가 감지되면 충돌이 발생한 비트 수와 처음 충돌이 발생한 비트의 위치를 저장한다. 만일 충돌이 발생한 비트의 수가 1개 이면 응답한 태그가 단지 2개인 경우이므로 해당 충돌 비트에 '0'과 '1'을 설정하고 메모리에 저장함으로 한 번에 두 개의 태그를 인식한다. 셋째, 두 개 이상의 충돌 비트가 감지된 경우 첫 번째 충돌비트 전까지의 프리픽스에 '0'과 '1'을 추가하여 새로운 프리픽스를 생성한 후 큐에 저장한다. 넷째, 응답하는 태그가 하나도 없는 경우 큐에서 새로운 프리픽스를 가져와 위의 과정을 큐가 빌 때 까지 반복한다. QT-CBP는 하나의 비트에서 충돌이 발생한 경우 QT와는 달리 이를 다시 질의하지 않고 해당 비트만 다른 두 개의 태그가 응답한 것을 고려하여 이들을 메모리에 저장하게 된다.

표 3은 리더의 영역 내에 '010', '011', '101', '110'의 네 개의 Tag_ID가 있다고 할 때 QT-CBP 알고리즘의 동작 과정을 나타낸다. QT 알고리즘과의 차이점은 두 번째 질의에서 볼 수 있다. 두 번째 질의에서 프리픽스 '0'으로 질의하면 영역 내의 태그 중 '010'과 '011' 두 개의 태그가 응답한다. 두 태그의 ID는 한 개의 비트(세 번째 비트)에서만 충돌이 발생했으므로 세 번째 비트에 각각 '0'과 '1'을 추가한 후 메모리에 저장하고 큐로부터 다음 프리픽스를 가져와 질의를 계속 수행한다. 결과적으로 모든 태그를 인식하는데 3번의 질의를 수행하게 된다. 즉 QT알고리즘을 사용할 때 필요한 총 9번의 질의와 CT 알고리즘을 사용할 때 필요한 총 7번보다 적은 질의를 필요로 함을 볼 수 있다.

표 3. QT-CBP 알고리즘

단계	1	2	3
Reader	ε	0	1
Response	XXX	01X	10X
Tag_1 010	010	010	
Tag_2 011	011	011	
Tag_3 101	100		100
Tag_4 110	101		101
Queue={}	0		
	1	1	
Memory		010	010
		011	011
			100
			101

2.3 기존 알고리즘의 문제점

위에서 살펴보았듯이 QT 알고리즘은 많은 무응답 태그와 충돌이 문제가 되며, 이는 태그가 리더기로 전체 태그 ID 정보를 전송함에도 불구하고 리더기가 수신된 식별자 정보를 전혀 이용하지 않고 단순히 충돌 발생여부만 결정하기 때문이다[6].

CT 알고리즘은 충돌을 감지한 리더가 전송 중지 명령을 태그들에게 전달하면 태그에는 이를 식별할 수 있는 별도의 기능이 추가되어야 한다. 즉 태그들은 프리픽스 B를 받고 자신의 태그 ID와 비교하여 일치하는 경우 한 비트씩 차례로 자신의 태그 ID를 전달하게 되는 데 전송 도중에 전송중지 명령을 받게 되면 이를 감지하고 처리할 수 있는 하드웨어와 소프트웨어가 추가적으로 필요하게 된다[10]. 이는 태그의 생산 비용을 증가시키게 되어 초경량화 및 저전력, 저가격 태그의 생산에 많은 어려움을 주게 된다. 뿐만 아니라 리더와 태그사이에 FDX(Full Duplex)가 필요하게 되

어 추가 비용이 발생한다.

QT-CBP 알고리즘은 충돌 비트를 추적하여 한 개의 비트만 충돌 시에는 별도의 프리픽스를 생성하지 않고 충돌된 비트에 '0'과 '1'의 값을 가진 두 개의 태그가 전송되었음을 자동 인식함으로써 질의 횟수를 획기적으로 단축하는 기법을 사용하였다. 그러나 이 역시 충돌을 추적하여 처리하는 과정에서 두개 이상의 비트에서 충돌이 발생하는 경우에 그 정보를 이용하여 성능을 개선시키는 방법에 대해서는 고려하지 않았다.

Ⅲ. 제안하는 충돌방지 알고리즘

본 논문은 기존의 QT 알고리즘과 QT-CBP 알고리즘을 토대로 태그 인식 성능을 개선할 수 있는 방안에 대해 연구하였다.

QT-CBP 알고리즘에서는 한 개의 비트에서만 충돌이 발생하면 충돌된 비트가 '0'과 '1'인 두 개의 태그가 전송된 것으로 인식하며, 2개 이상의 비트에서 충돌이 발생할 경우에는 충돌이 처음 발생하는 비트 이전까지의 비트열에 '0'과 '1'을 더해 새로운 프리픽스로 하고 프리픽스 큐에 저장한다. 즉 충돌이 발생할 경우 한 번에 한 개의 충돌된 비트를 해결 한다. 본 논문에서 제안하는 QT-DCBP(Query Tree - Double Collision-Bit Partitioning) 알고리즘은 QT-CBP와 동일하게 한 개의 비트에서만 충돌이 발생하면 충돌된 비트가 '0'과 '1'인 두 개의 태그가 전송된 것으로 인식하며, 2개 이상의 비트에서 충돌이 발생하는 경우에는 한 번에 두 개의 충돌된 비트를 동시에 해결해 주고자, 충돌한 처음 두 개의 비트를 모든 가능한 경우인 각각 '00', '01', '10', '11'로 치환하여 새로운 프리픽스를 생성하고 큐에 저장한다.

다음의 예들은 QT-DCBP의 성능이 QT-CBP 기법보다 좋아질 수 있음을 보여준다.

태그가 2개의 비트로 구성된다고 가정하고 리더의 영역 내에 '00', '01', '10', '11'의 네 개의 태그가 존재한다고 가정하자. QT 알고리즘을 적용하면 'ϵ', '0', '00', '01', '1', '10', '11'과 같이 7번 질의를 하여야 한다. FCT QT 알고리즘은 QT 알고리즘과 동일하게 'ϵ', '0', '00', '01', '1', '10', '11'과 같이 7번(=1+2+4) 질의를 하여야 하며 QT-CBP 알고리즘을 적용하면 'ϵ', '0', '1'과 같이 3번(=1+2)만 질의하면 된다. 반면 본 논문에서 제안하는 QT-DCBP 알고리즘을 적용하면 'ϵ', '00', '01', '10', '11'과 같이 5번(=1+4) 질의를 하면 된다.

태그가 3개의 비트로 구성된다고 가정하고 리더의 영역 내에 '000', '001', '010', '011', '100', '101', '110', '111'의 8개의 태그가 존재한다고 가정하자. QT 알고리즘을 적용하면 'ϵ', '0', '00', '000', '001', '01', '010', '011', '1', '10', '100', '101', '11', '110', '111'과 같이 15번(=1+2+4+8) 질의를 하여야 한다. FCT QT 알고리즘은 QT 알고리즘과 동일하게 15번 질의를 하여야 하며 QT-CBP 알고리즘을 적용하면 'ϵ', '0', '00', '01', '1', '10', '11'과 같이 7번(=1+2+4)만 질의하면 된다. QT-DCBP 알고리즘을 적용하면 'ϵ', '00', '01', '10', '11'과 같이 5번(=1+4)만 질의를 하면 된다.

다음 표 4는 태그 비트수가 변화할 때 그리고 리더 영역 내에 가능한 모든 태그들이 존재한다고 가정할 때(태그가 n비트라면 2^n개의 태그가 존재 가능함) 각 기법별 리더의 질의 수를 나타낸다. 표 4에서 보는 바와 같이 태그 비트가 홀수 일때는 QT-DCBP 성능이 가장 좋아지고 짝수일 때는 QT-DCBP의 성능이 QT-CBP 성능보다 나빠짐을 볼 수 있다.

그런데 표 4에서 보는 바와 같이 태그의 비트수가 2개일 때에는 QT-CBP 알고리즘의 성능이 QT-DCBP 알고리즘의 성능보다 좋으므로 이를 고려하여 리더기가 수신한 식별자 중에서 충돌된 태그의 비트수가 3개 이상일 때에만 QT-DCBP 알고리즘을 적용한다고 하자. 그러면 태그가 4개의 비트로 구성되고 리더의 영역 내에 가능한 모든 태그인 '0000', '0001', …, '1111'의 16개의 태그가 존재한다고 할 때 QT-CBP 알고리즘을 적용하면 15(=1+2+4+8)번의 질의를 해야 하지만 QT-DCBP 알고리즘을 적용하면 13(=1+4+8)번의 질의만 하면 된다. 즉 표 4의 21번 대신 13번의 질의만 하면 되므로 QT-DCBP 알고리즘의 성능이 가장 좋아짐을 알 수 있다.

태그 밀도를 리더 영역 내에 존재 가능한 모든 태그 수 의 비율로 나타낸다고 하자. 즉 태그 ID 비트가 8비트일 때, 영역 내에 최대 256개의 태그가 존재할 수 있으므로 256개의 태그가 존재한다면 태그 밀도는 100%, 128개의 태그가 존재한다면 태그 밀도는 50%

표 4. 태그 비트 수에 따른 리더 질의 수 비교

태그 비트수	리더 영역 내 태그 수	기법별 리더 질의 수		
		QT	QT-CBP	QT-DCBP
2	4	7	3	5
3	8	15	7	5
4	16	31	15	21
5	32	63	31	21

가 된다. 표 4에서 보는 바와 같이 태그가 3개의 비트로 구성되고 리더 영역 내에 가능한 모든 태그들이 있을 때(태그밀도가 1일 때)에는 QT-DCBP의 성능이 가장 좋음을 볼 수 있었다. 태그 밀도가 1보다 작을 때에는 QT-DCBP의 성능이 어떻게 변화하는지를 살펴보면 다음과 같다.

태그가 3개의 비트로 구성되고 리더의 영역 내에 '000', '111'의 두 개의 태그가 존재한다고 가정하자(태그밀도 =0.25). 이때 QT 알고리즘을 적용하면 'ε', '0', '1'과 같이 3번 질의를 하여야 한다. 처음충돌추적 QT 알고리즘을 적용하면 'ε', '0', '1'과 같이 3번 질의를 하여야 하며 QT-CBP 알고리즘을 적용하면 'ε', '0', '1'과 같이 3번 질의를 하여야 한다. QT-DCBP 알고리즘을 적용하면 'ε', '00', '01', '10', '11'과 같이 5번의 질의를 해야 된다.

태그가 3개의 비트로 구성된다고 가정하고 리더의 영역 내에 '000', '001', '010', '011'의 네 개의 태그가 존재한다고 가정하자(태그밀도 0.5). 이때 QT 알고리즘을 적용하면 'ε', '0', '00', '000', '001', '01', '010', '011', '1'과 같이 9번 질의를 하여야 한다. 처음충돌추적 QT 알고리즘을 적용하면 'ε', '00', '000', '001', '01', '010', '011', 같이 7번 질의를 하여야 하며 QT-CBP 알고리즘을 적용하면 'ε', '00', '01'과 같이 3번만 질의를 하면 된다. QT-DCBP 알고리즘을 적용하면 'ε', '000', '001', '010', '011'과 같이 5번의 질의를 해야 된다.

다음 표 5와 표 6은 태그의 비트가 3비트, 4비트일 때 리더 영역 내에 존재 가능한 모든 경우를 고려한 평균 질의-응답 수를 나타낸다. 예로서 태그의 비트가 3비트일 때 태그밀도가 25%가 되는 경우는 리더영역 내에 전체 8개의 가능한 태그중 2개의 태그가 존재하는 경우로 ('000', '001'), ('000', '010'), …('110', '111')와 같이 28($=_8C_2$) 경우가 존재한다. 이 각각의 경우에 대해 태그 질의-응답수를 구해서 평균 질의-응

표 5. 3비트 태그일 때의 질의 응답 횟수

태그밀도 (%)	경우의수	QT	QT-CBP	QT-DCBP
25	8C2	2.0	1.1	1.4
37.5	8C3	4.0	2.6	3.1
50	8C4	6.0	3.6	3.8
62.5	8C5	8.0	4.6	4.0
75	8C6	10.0	5.4	4.0
87.5	8C7	12.0	6.0	4.0
100	1	14.0	6.0	4.0

표 6. 4비트 태그일 때의 질의 응답 횟수

태그밀도(%)	경우의수	QT	QT-CBP	QT-DCBP
25	16C4	6.0	4.4	4.4
50	16C8	14.0	9.1	7.4
75	16C12	22.0	12.6	10.6
100	1	30.0	14.0	12.0

답횟수를 구하면 표 5에서 보는 바와 같이 QT, QT-CBP, QT-DCBP 기법이 각각 2.0, 1.1, 1.4개가 됨을 볼 수 있다.

표 5에서 보는 바와 같이 3비트 태그일 때에는 태그밀도가 62.5% 이상이 되면 본 논문에서 제안하는 QT-DCBP의 성능이 가장 좋아짐을 볼 수 있다. 한편 표 6은 4비트 태그일 때에는 태그밀도가 25% 이상이 되면 QT-DCBP의 성능이 가장 좋아짐을 보여준다.

표 5에서 태그밀도가 25% - 50%일 때 보는 바와 같이 태그 밀도가 낮을 때에는 QT-DCBP 알고리즘의 성능이 다소 나빠질 수 있음을 볼 수 있다. 따라서 태그 밀도가 다소 높을 때에만 QT-DCBP 알고리즘을 적용하는 것이 바람직하다 할 수 있는데, 리더 영역 내에 존재하는 태그의 수를 정확히 예측하는 것은 어렵기 때문에 본 논문에서는 태그가 리더에게 전송하는 모든 비트가 충돌하면 태그 밀도가 어느 정도 높다고 생각하여 모든 비트가 충돌될 때에만 QT-DCBP 알고리즘을 적용하고자 한다.

본 논문에서 제안하는 QT-DCBP 알고리즘은 충돌된 비트수가 3개 이상일 때 그리고 태그밀도가 다소 높을 때 충돌된 2 비트를 해결해 주고자 한다. QT에서와 같이 리더가 d비트 길이의 프리픽스 $B_d(B=b_1b_2, ..., b_d)$를 태그에게 질의하면, 태그들은 자신의 식별자와 프리픽스를 비교하여 그 값이 동일하면 d+1비트부터 마지막 n비트까지의 값을 리더에게 전송한다고 하자. QT-DCBP 알고리즘 하에서 리더는 태그들의 응답결과에 따라 다음과 같은 절차에 따라 태그들을 인식한다.

(1) 하나의 태그만 응답 한 경우 : 이 경우는 충돌이 발생하지 않았으므로 해당 태그를 메모리에 저장한다.

(2) 한 개의 비트에서만 충돌이 발생한 경우 : 이 경우는 단지 2개의 태그만 전송되었으므로 충돌이 발생한 비트에 '0'과 '1'을 추가한 후 메모리에 저장한다.

(3) 태그가 응답한 비트가 세 개 이상이고, 응답한 모든 비트들이 충돌한 경우 : 충돌이 발생한 첫 번째 위치와 두 번째 위치에 '00', '01', '10', '11'을 추가

표 7. QT-DCBP 알고리즘의 과정

리더기 동작
Step 1: 모든 태그에 질의
Step 2: 태그로부터 응답 확인
- 무응답한 경우 -> Step 7
- 하나의 태그만 응답한 경우
-> Step 3
- 한 비트만 충돌이 발생한 경우 -> Step 4
- 응답 비트가 3개 이상이고 응답한 모든 비트가 충돌이 발생한 경우 -> Step 5
- 그 이외의 경우 -> Step 6
Step 3: 충돌이 발생하지 않음
- 메모리에 태그 ID 저장
-> Step 7
Step 4: 2개의 태그만 전송된 경우
- 충돌 비트에 '0'과 '1'을 추가하여 2개의 태그 ID를 메모리에 저장 -> Step 7
Step 5: 충돌이 발생한 첫 번째 위치와 두 번째 위치에 '00', '01', '10', '11'을 추가한 새로운 프리픽스를 큐에 저장 -> Step 7
Step 6: 충돌이 처음 발생하는 비트 이전까지의 비트열에 '0'과 '1 을 추가하여 2개의 태그 ID를 메모리에 저장
-> Step 7
Step 7: 큐에 저장된 프리픽스를 가져와 다음 질의 수행
- 큐에 저장된 프리픽스가 없는 경우 -> Step 8
Step 8: 알고리즘 종료

태그 동작
Step 1: 리더로부터 수신한 프리픽스(b_d)와 태그 ID비교
- 일치하면 -> Step 2
- 일치하지 않으면 -> Step 3
Step 2: d+1비트부터 마지막 n비트까지의 값을 리더에게 전송
Step 3: 무응답

하여 큐에 저장하고, 큐에서 다음 프리픽스를 가져와 질의를 수행한다. 표 7은 QT-DCBP 알고리즘의 전체 과정을 나타낸다.

표 8은 리더의 영역 내에 '0000', '0010', '0011', '0101', '0111', '1000', '1001', '1010', '1011', '1110'의 열 개의 Tag_ID가 있다고 할 때 QT-DCBP 알고리즘의 동작 과정을 나타낸다.

QT-DCBP 알고리즘의 특징은 표 8.의 두 번째 열에서 보는 것처럼(리더가 ε 으로 질의했을 때) 응답비트가 3개 이상이고, 응답한 모든 비트들이 충돌한 경우 충돌이 발생한 첫 번째 위치와 두 번째 위치에 '00', '01', '10', '11'을 추가하여 큐에 저장한다. 표 8.의 과정을 트리로 표현하면 그림 1과 같다. 이 때 영역 내의 모든 태그를 인식하는데 9번의 질의가 필요함을 볼 수 있다(노드위에 네모상자가 있는 노드 내에 있는 비트들이 질의되는 프리픽스들을 나타냄).

표 8. QT-DCBP 알고리즘

Request Prefix	ε	00	01	10	11	000	001	100	101
Response	XXXX	00XX	01X1	10XX	1110	0000	001X	100X	101X
Tag_01 0000	0000	0000				0000			
Tag_02 0010	0010	0010					0010		
Tag_03 0011	0011	0011					0011		
Tag_04 0101	0101		0101						
Tag_05 0111	0111		0111						
Tag_06 1000	1000			1000				1000	
Tag_07 1001	1001			1001				1001	
Tag_08 1010	1010			1010					1010
Tag_09 1011	1011			1011					1011
Tag_10 1100	1100				1100				
Queue={ε}	00 01 10 11	01 10 11 000 001	10 11 000 001	11 000 001 100 101	000 001 100 101	001 100 101	100 101	101	
Memory			0101 0111	0101 0111	0101 0111 1110	0101 0111 1110 0000	0101 0111 1110 0000 0010 0011	0101 0111 1110 0000 0010 0011 1000 1001	0101 0111 1110 0000 0010 0011 1000 1001 1010 1011

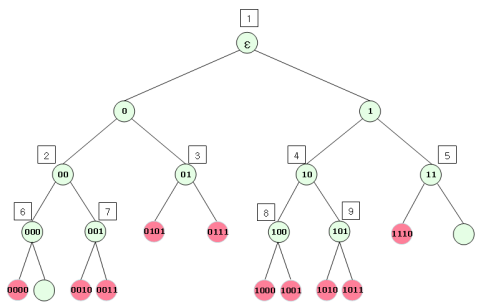

그림 1. QT-DCBP 알고리즘

Ⅳ. 성능평가

본 논문에서 제안하는 QT-DCBP 알고리즘의 성능을 평가하기 위해 기존의 충돌해결기법인 QT알고리즘과 QT-CBP 알고리즘의 성능과 비교하였다. 태그들

이 각각 8비트, 10비트, 11비트, 12비트, 13비트로 구성된다고 가정하고, 하나의 리더 영역 내에 존재하는 태그의 수에 따라 제안하는 알고리즘의 성능이 달라지는 지를 평가하기 위해서 표 8.에서 보는 바와 같이 하나의 리더 영역 내에 존재하는 태그의 수를 변경시키면서 성능을 비교하였다. 예로서 태그들이 8비트로 구성된다면 리더 영역 내에 존재 가능한 태그의 수는 256(=2^8)개가 된다. 따라서 모든 가능한 태그 수의 20%, 40%, 60%, 80%인 50, 100, 150, 200개의 태그들이 리더 영역 내에 존재할 때에 각 기법의 성능을 비교하였다.

기법 간 성능은 태그인식을 위해 리더와 태그 간에 주고받는 질의 응답과정을 JAVA로 프로그램한 후 전체 태그를 인식하는 데 소요되는 질의-응답 수를 토대로 비교하였다.

표 9의 경우처럼 각 태그 비트 별 태그 밀도에 맞도록 태그 개수를 랜덤하게 발생시킨 후(8bit 태그비트의 20% 밀도의 경우 50개) 발생된 태그 전체를 인식하는 데 소요되는 질의응답수를 구하는 과정을 각 10회씩 시뮬레이션한 수의 평균값을 구함으로써 태그의 밀도에 따른 성능을 비교하였다.

표 10은 시뮬레이션 결과를 나타낸다. 하나의 리더 영역 내에 가능한 전체 태그 수의 20%, 40%, 60%, 80% 만큼의 태그가 존재할 때, 태그 ID가 각각 8비트, 10비트, 11비트, 12비트, 13비트일 경우 모든 태그들을 식별할 때까지 리더기로 전송되는 총 태그 수를 10번 시뮬레이션 해서 얻은 평균값을 나타낸다.

예로서 표 10에서 리더 영역 내의 태그밀도가 20%이고 태그 비트가 8비트일 때 QT 기법, QT-CBP기법, QT-DCBP 기법 하에서 각각 평균적으로 98.0, 79.8, 68.9 개의 태그 ID가 리더기로 전송됨을 볼 수 있다. 따라서 QT-CBP 기법은 QT 기법보다 태그 응답 횟수가 18.6% 줄어들었고 QT-DCBP 기법은 QT 기법보다 28.8% 줄어들었음을 볼 수 있다. 또한 QT-DCBP 기법이 QT-CBP 보다도 12.5% 줄어들었음을 볼 수 있다. 시뮬레이션 결과 태그 밀도별 각 태그 비트의 질의응답 감소 비율은 거의 비슷함을 알 수 있다. 반면 태그 밀도가 증가함에 따라 질의 응답 감소비율도 증가함을 볼 수 있다. 이것은 각 기법의 상대적인 성능은 주로 태그밀도에 의존함을 알 수 있고 태그밀도가 높아질수록 성능이 더욱 좋아짐을 볼 수 있다. 즉 태그밀도가 20%라면 태그 비트가 8, 10, 11, 12, 13 비트일 때 모두 약 19%와 29%의 성능 개선 효과를 볼 수 있다. 주어진 태그밀도 하에서 모든 경우에 QT 알고리즘, QT-CBP 알고리즘, QT-DCBP 알고리즘 순으로 성능이 향상됨을 알 수 있다. 특히 QT-CBP 알고리즘과 QT-DCBP 알고리즘을 비교하면 모든 경우에 10%이상 성능이 향상되었음을 알 수 있다.

그림 2에서 그림 6은 각 비트별 태그밀도에 따른 질의 응답 수를 그래프로 나타낸 것이다. 그림에서 볼 수 있듯이 모든 경우에 질의 응답 수가 QT > QT-CBP > QT-DCBP 로 줄어들었음을 볼 수 있다.

그림 7은 태그 ID가 8비트, 10비트, 11비트, 12비트, 13비트일 경우 태그밀도가 20%, 40% 60% 80%인 경우 영역 내의 모든 태그를 인식하는데 필요한 질의-응답 횟수의 평균을 비교한 그래프이다. QT 알고리즘과 비교했을 때 QT 알고리즘보다는 QT-CBP 알

표 9. 태그 비트수 별 생성된 태그 수

태그 비트	Full	20 (%)	40 (%)	60 (%)	80 (%)
8	256	50	100	150	200
10	1024	200	400	600	800
11	2048	400	800	1200	1600
12	4096	800	1600	2400	3200
13	8192	1600	3200	4800	6400

표 10. 태그 밀도별 리더가 전송하는 질의 응답 수

태그 밀도	태그 비트	QT [1]	QT-CBP [2]	QT-DCBP [3]	[1] vs. [2]	[1] vs. [3]	[2] vs. [3]
20%	8	98.0	79.8	69.8	18.6	28.8	12.5
	10	398.0	320.4	278.8	19.5	29.9	13.0
	11	798.0	644.6	560.8	19.2	29.7	13.0
	12	1598.0	1290.8	1125.8	19.2	29.5	12.8
	13	3198.0	2584.2	2268.2	19.2	29.1	12.2
	평균				19.1	29.4	12.7
40%	8	198.0	144.8	117.6	26.9	40.6	18.8
	10	798.0	574.8	473.2	28.0	40.7	17.7
	11	1598.0	1151.8	946.6	27.9	40.8	17.9
	12	3198.0	2314.6	1907.2	27.6	40.4	17.6
	13	6398.0	4615.2	3799.6	27.9	40.6	17.7
	평균				27.6	40.6	17.9
60%	8	298.0	193.8	155.6	35.0	47.8	19.7
	10	1198.0	783.8	629.4	34.6	47.5	19.7
	11	2398.0	1573.0	1261.4	34.4	47.4	19.8
	12	4798.0	3128.0	2517.6	34.8	47.5	19.5
	13	9598.0	6298.6	5063.4	34.4	47.2	19.6
	평균				34.6	47.5	19.7
80%	8	397.2	235.6	194.0	40.7	51.2	17.7
	10	1597.4	941.2	772.2	41.1	51.7	18.0
	11	3198.0	1894.8	1555.8	40.8	51.4	17.9
	12	6398.0	3774.0	3097.4	41.0	51.6	17.9
	13	12798.0	7552.6	6197.8	41.0	51.6	17.9
	평균				40.9	51.5	17.9

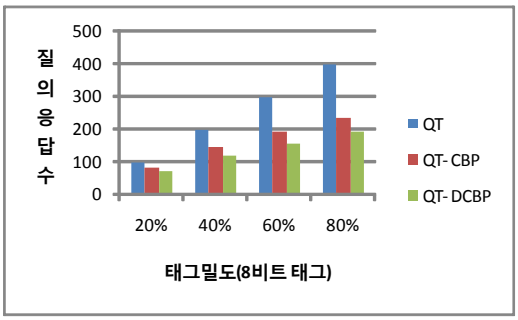

그림 2. 8비트 태그일 때의 질의 응답 수

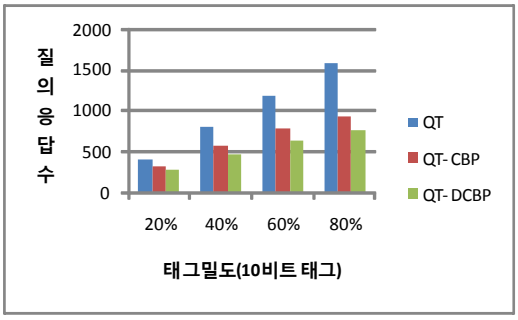

그림 3. 10비트 태그일 때의 질의 응답 수

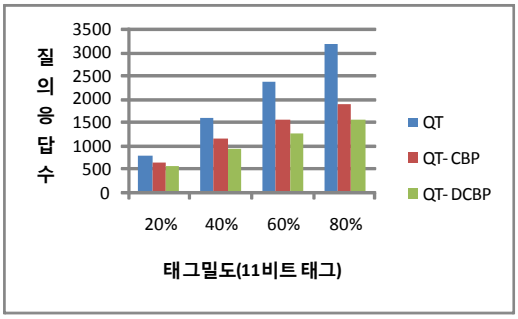

그림 4. 11비트 태그일 때의 질의 응답 수

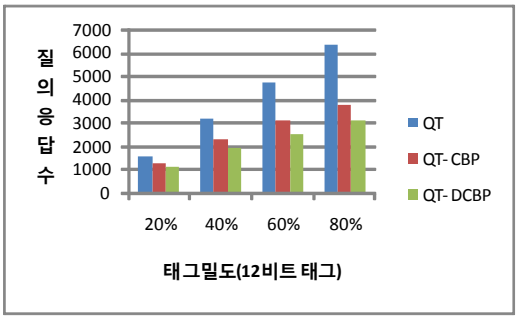

그림 5. 12비트 태그일 때의 질의 응답 수

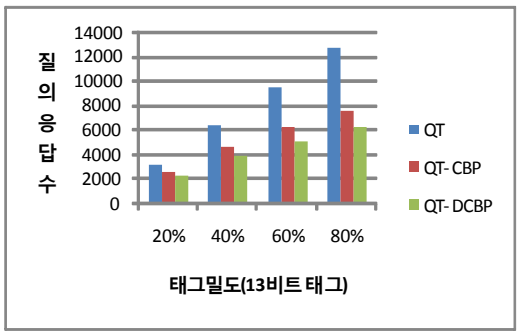

그림 6. 13비트 태그일 때의 질의 응답 수

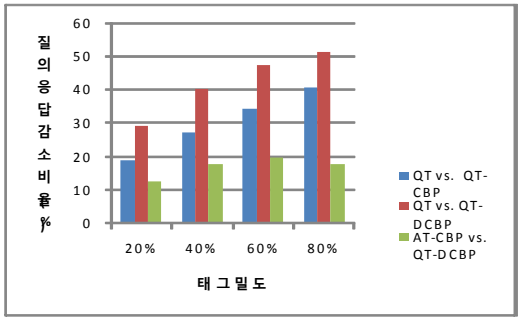

그림 7. 태그밀도별 질의 응답 감소 비율

고리즘이, QT-CBP 알고리즘보다는 QT-DCBP 알고리즘의 성능이 모든 경우에 거의 일정한 비율로 성능이 우수함을 볼 수 있다. 이는 태그 비트와 태그 밀도가 증가해도 QT-DCBP 알고리즘이 QT알고리즘은 물론 QT-CBP 알고리즘보다도 성능이 우수함을 나타낸다.

V. 결 론

본 논문에서는 RFID anti-collision 알고리즘 중 QT 알고리즘의 성능을 개선할 수 있는 QT-DCBP기법을 제안하고 기존의 QT 알고리즘 및 QT-CBP 알고리즘과 성능을 비교하였다. 성능 비교결과 QT-DCBP 알고리즘은 QT-CBP 기법보다도 질의-응답횟수를 10% 이상 줄일 수 있음을 볼 수 있었다.

본 논문에서는 리더 영역 내에 존재하는 태그의 개수를 정확히 알 수 없으므로 태그가 전송하는 모든 비트(3bit 이상)가 충돌되는 경우에만 2개의 충돌된 비트를 4개의 가지로 분지하는 기법을 사용 하였는데 리더 영역 내에 존재하는 태그의 개수를 알 수 있다면 이에 기초하여 태그의 개수가 많을 때에는 4개의 가지 수로 분지하고 그렇지 않으면 기존 QT-CBP 기법

을 사용할 수 있을 것이다. 향후 이와 관련된 연구를 통해 보다 성능을 개선시킬 수 있을 것이다.

참 고 문 헌

〔1〕 K.Finkenzeller, “RFID handbook, Fundamentals and Applications in Contactless Smart Cards and Identification”, 2th Edition, *John Wiley & Sons*, 2003

〔2〕 A. Jules, R. Rivest, and M. Szydol. “The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy.” *Proceedings of the 10th ACM conference on Computer and communications security,* ISBN:1-58113-738-9, pp.103-111. 2003

〔3〕 Ching Law, Kayi Lee, and Kai-Yeung Sju, “Efficient Memoryless Protocol for Tag Identification”, *In Proceeding of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp.75-84, ACM. August 2000.

〔4〕 Jacomet M, Ehrsam A, Gehrig U. “Contactless identification device with anti-collision with algorithm”, *IEEE computer Society, CSCC’99, Conference on Circuits, Systems, Computers and Communications*, Athens. 4-8 July 1999.

〔5〕 Vogt, H. “Efficient Object Identification with Passive RFID Tags”. *In International Conference on Pervasive Computing, LNCS*. Spring-Verlag 2002.

〔6〕 권성호, 홍원기, 이용두, 김희철, “RFID 시스템에서 트리 기반 메모리레스 충돌방지 알고리즘에 대한 연구”, *정보처리학회논문지 C 제11권 제6호*, 12월 2004.

〔7〕 Leian Liu, Zhenhua Xie, Jingtian Xi, Shengli Lai, “An Improved Anti-collision Algorithm in RFID System”, *2nd International Conference on Mobile Technology, Applications and Systems*, 3-2A-3, Nov. 2005.

〔8〕 SungSoo Kim, YongHwan Kim, SeongJoon Lee, KwangSeon Ahn, “An Improved Anti Collision Algorithm using Parity Bit in RFID System”, *7th IEEE International Symposium on Network Computing and Applications*, pp. 224-227, July 2008

〔9〕 Feng Zhou, Dawei jin, Chenling Huang and Hao Min, “White Paper: optimize the power Consumption of Passive Electronic TAgs for Anti-collision Schemes”, *Auto-ID center Fudan Univ.*, October, 2003

〔10〕 서현곤, “RFID 시스템에서 충돌 트리 기반 충돌방지 알고리즘”, *정보과학회논문지,* 정보통신 제34권 제5호, 10월 2007.

〔11〕 이현지, 김종덕, “충돌 비트 위치를 활용한 RFID 다중 태그 인식 알고리즘”, *한국통신학회논문지*, Vol.31 No.4A, 4월 2006.

김 명 환 (Myeoung-Hwan Kim) 정회원

2003년 8월 숭실대학교 컴퓨터학부

2004년 8월~현재 서울산업 대학교 IT정책전문대학원 석·박사 통합과정

<관심분야> 정보통신

국 광 호 (Kwang-Ho Kook) 정회원

1979년 2월 서울대학교 산업공학과 학사

1981년 2월 서울대학교 산업공학과 석사

1989년 2월 : Georgia Institute of Technology 산업공학과 박사

1993년 2월~현재 서울산업대학교 산업정보 시스템공학과 교수

<관심분야> 정보통신, 모델링&스케쥴링