RFID 다중 태그 인식을 위한 스택 Bit-By-Bit 알고리즘

준회원 이 재 구*, 유 대 석*, 정회원 최 승 식*

A Stack Bit-by-Bit Algorithm for RFID Multi-Tag Identification

Jae-ku Lee*, Dae-suk Yoo* *Associate Members*, Seung sik Choi* *Regular Member*

요 약

RFID(Radio Frequency IDentification) 리더기가 영역내의 다수의 태그를 인식하기 위해선 충돌방지 알고리즘이 반드시 필요하다. 본 논문은 Auto ID Class 0에서 정의한 충돌방지 알고리즘인 Bit-by-Bit(BBB) 이진트리 알고리즘의 충돌 위치를 스택에 저장하고 이를 통해 다음 질의어를 결정함으로써 성능이 크게 개선된 Stack-Bit-by-Bit (SBBB) 알고리즘을 제안한다. 시뮬레이션을 통한 검증결과 대표적인 충돌 방지 기술인 Query Tree(QT)는 물론 기존의 BBB 알고리즘에 비해 질의-응답 횟수, 질의어의 크기, 응답어의 크기의 모든 면에서 성능이 개선된 것을 확인할 수 있었다.

Key Words : RFID, Anti-collision algorithm, Bit-by-Bit algorithm, EPC, Query Tree algorithm

ABSTRACT

For the implementation of a RFID system, an anti-collision algorithm is required to identify multiple tags within the range of a RFID Reader. A Bit-by-Bit algorithm is defined by Auto ID Class 0. In this paper, we propose a SBBB(Stack Bit-by-Bit) algorithm. The SBBB algorithm save the collision position and makes a query using the saved data. SBBB improve the efficiency of collision resolution. We show the performance of the SBBB algorithm by simulation. The performance of the proposed algorithm is higher than that of BBB algorithm. Especially, the more each tag bit streams are the duplicate, the higher performance is.

Ⅰ. 서 론

RFID(Radio Frequency IDentification) 기술이란 관리할 모든 개별 사물에 전자태그를 부착하여 리더기로 무선 전파를 태그에 보내 태그가 전파신호를 인식한 후 자신의 ID정보를 리더기로 전달하는 방식으로 리더기와 태그사이에 비접촉으로 태그 정보를 인식 할 수 있는 기술이다. 기존의 바코드 시스템과는 달리 무선 전파를 사용함으로써 바코드의 문제점을 극복 할 수 있을 뿐만 아니라 개별 사물을 인식하고 인터넷을 통하여 우리생활의 모든 분야 즉, 농축산물, 물류·유통, 보안 등의 영역에서 정보화를 가능하게 하는 기술이다. 특히 최근 물류 비용의 절감이 기업들의 화두가 되면서 물류의 SCM(Supply Chain Management) 측면에서 RFID 기술 도입이 빠르게 진행 중이며 관련 연구 또한 활발히 진행 중이다. 하지만 장치들 사이의 호환성, 리더기 사이의 충돌, 태그 사이의 충돌 등 아직 해결해야할 문제점이 있는 상황이다. 여기서의 충돌이란 태그들 간의 통신 과정에서 간섭을 뜻하며 RFID 시스템의 성능과 안정성을 위해 반드시 해결해야할 문제이다. 이렇듯 다중 태그 환경에서 태그 간의 충돌 현상을 회피하는 기술을 다중 태그인식 기술[1]이라 한다. 본 논문은 Auto ID Class 0 에서

※ 본 연구는 한국산업자원부지정 인천대학교 동북아전자물류연구센터의 지원에 의한 것임
* 인천대학교 컴퓨터공학과 무선정보 네트워크 연구실(whistler29, lagnarok82, sschoi@incheon.ac.kr)
논문번호 : KICS2007-04-170, 접수일자 : 2007년 4월 11일, 최종논문접수일자 : 2007년 7월 11일

정의된 Bit-by-Bit(BBB) 알고리즘[2]의 성능을 개선한 Stack-Bit-by-Bit(SBBB) 알고리즘을 제안한다. SBBB 알고리즘은 기존의 BBB 알고리즘에 태그 충돌 위치 정보를 스택에 저장하고 이를 활용하여 성능을 개선하는 방안이다.

RFID 표준화는 현재 Auto-ID 센터를 중심으로 EPCglobal에 의해 표준화가 진행 중이다. EPC Class 0에서 정의된 EPC(Electronic Product Code) code는 64비트로 구성되어 있다. 하지만 현재 EPC코드는 96비트가 사실상 RFID 표준코드로 자리잡아가고 있다. EPC코드는<Version Header, EPC manager, Product Class, Serial Number>의 계층적 구조를 갖으며[3] 상품 하나하나에 96비트의 EPC코드를 부여하여 해당 상품에 관한 생산정보나 유통이력 등 상품의 상세정보를 인터넷을 통해 알 수 있도록 하는 것을 목표로 하고 있다. 따라서 물류환경에서 비슷한 상품의 각 태그들의 ID는 중복될 확률이 크다. 이는 SBBB의 효과가 클 것임을 의미한다. 시뮬레이션을 통한 검증결과 질의-응답 횟수, 질의어의 크기, 응답어의 크기의 모든 면에서 성능이 개선된 것을 확인 할 수 있었으며 특히 태그ID의 중복성이 클수록 우수한 성능을 보였다.

Ⅱ. 관련 연구

2.1 충돌방지 알고리즘의 분류

다중태그 식별을 위한 충돌회피 방법으로는 크게 트리기반의 결정적 알고리즘과 슬롯 알로하 기반의 확률적 알고리즘으로 구분 할 수 있다. 결정적 알고리즘은 태그 식별자들이 이진비트로 표현되어 이진트리를 구성한다. 인식은 이진트리를 순회하면서 태그들을 차례로 인식하게 되는데 태그의 비트정보를 알고 있다면 알고리즘에 의해 태그의 인식 순서를 예측 할 수 있다. 확률적 알고리즘은 태그가 시간차에 따라 서로 다른 슬롯을 선택함으로써 태그를 인식하는 방법이다. 확률에 의해 태그를 인식함으로 모든 태그를 완벽하게 인식 하는 것은 불가능 하다. 하지만 결정적 알고리즘과는 달리 모든 트리를 순회하지 않아도 되기 때문에 상대적으로 빠른 시간내에 태그 인식할 수 있다는 장점이 있다. 확률적 알고리즘은 태그의 슬롯 선택에 따라 그 성능이 좌우 되므로 슬롯의 수가 중요한 이슈가 된다. 슬롯의 수가 적을 경우 충돌이 발생한 슬롯의 재전송에 따른 성능 저하가 심하고 슬롯의 개수가 필요이상 많으면 또한 성능을 저하시키는 요인이 된다. 최근 리더 인식 범위 내의 태그개수를 추정하여 최적의 슬롯의 수를 찾고 이를 통해 성능을 개선하기 위한 연구가 진행 중이지만 아직 보안해야 할 문제점이 많은 상태이다.

트리기반의 결정적 알고리즘은 태그의 메모리 유무에 따라 메모리형 알고리즘[4]과 비메모리형 알고리즘[5]으로 나눌 수 있다. 메모리형 알고리즘은 태그가 태그ID 저장 공간 이외의 추가적인 메모리를 갖는 알고리즘으로 비트-중재(bit-arbitration) 알고리즘[6], 분할 트리(splitting tree) 알고리즘[4], 비트바이비트(Bit-by-Bit) 알고리즘[2]이 있다. 비메모리형 알고리즘은 태그가 태그ID만을 저장하는 알고리즘으로 트리-워킹(tree-walking) 알고리즘[5], Query Tree 알고리즘[7], 충돌 추적 트리(collision tracking tree) 알고리즘[8], Bin-Slot 이진트리 알고리즘[9]들이 연구되었다.

알로하 기반의 확률적 알고리즘으로는 I-Code 알고리즘[10], STAC(Slotted Terminating Adaptive Collection) 알고리즘[11], Bit-Slot 알고리즘[12], Framed Slotted ALOHA 알고리즘[13], Dynamic Framed Slotted ALOHA 알고리즘[14]들이 있다. 표 1은 지금까지 살펴본 충돌방지 알고리즘들을 보여주고 있다.

표 1. 충돌방지 알고리즘의 분류

충돌 방지 알고리즘의 분류		
트리 기반 알고리즘(결정적)	메모리형	비트-중재 알고리즘
		분할트리 알고리즘
		비트바이비트 알고리즘
	비메모리형	트리-워킹 알고리즘
		쿼리트리 알고리즘
		충돌추적트리알고리즘
슬롯-알로하 기반 알고리즘(확률적)		I-Code
		STAC
		Bit-Slot
		Framed Slotted ALOHA
		Dynamic Framed Slotted ALOHA

2.2 Query Tree 알고리즘

QT 알고리즘은 트리기반의 비메모리형 알고리즘으로 태그 응답에 따라 리더가 응답하는 질의가 결정되는 구조로 되어있다. 알고리즘의 수행은 큐가 비어 있게 됨으로 브로드케스팅을 하는 것으로 시

작한다. 리더기가 큐에서 k-비트 프리픽스를 가져와 모든 태그들에게 질의를 하다. 리더기가 보낸 k-비트 프리픽스와 매칭 된 태그들은 자신의 ID를 리더기로 전송한다. 이때 응답은 세 경우로 나뉜다. 첫 번째는 오직 하나의 태그만 응답을 한 경우이고 두 번째는 응답이 없는 경우이다. 세 번째는 두 개 이상의 태그가 동시에 응답을 한 경우이다. 첫 번째와 두 번째 경우에는 큐에 저장된 값을 가져와 새로운 프리픽스로 사용한다. 세 번째 경우에는 충돌이 발생한 경우로서 기존 프리픽스에 '0'과 '1'을 추가하여 각각을 큐에 저장하게 된다. 그 후 큐에서 새로운 프리픽스를 가져와 다음번 질의-응답 과정을 계속 하게 된다. 이러한 과정은 큐가 빌 때까지 반복하게 되는데 큐가 비게 되면 리더영역 내의 모든 태그인식이 완료된다.

QT 알고리즘은 비메모리형 알고리즘으로 태그 인식은 질의어에 의해서만 이루어진다. 질의어는 0과 1을 계속 더하여 만들어지는데 동일한 질의어가 생기지 않게 된다. 따라서 태그를 액티브 상태와 인액티브 상태로 변경할 필요가 없다. 그리고 질의어가 하나의 태그ID와 매칭 되면 태그는 자신의 모든 ID를 응답하고 리더는 이 신호를 인식한다. 따라서 리더기는 태그ID 정보를 저장할 필요가 없다. 질의어와 여러 태그가 동시에 매칭이 되게 되면 이때 충돌이 발생 한 것으로 인식한다. 충돌이 발생하면 여러 태그가 동시에 자신의 모든ID를 응답함으로 충돌이 많이 발생 할수록 전체응답어의 크기는 커지게 된다. QT 알고리즘은 이러한 특성 때문에 태그ID가 길고 태그의 개수가 적어 태그ID의 중복성이 낮을 경우 높은 성능을 기대 할 수 있으나 태그ID가 짧고 태그의 개수가 많아져 태그ID의 중복성이 커지게 되면 태그가 모든 비트정보를 전송함으로 인해 응답어의 크기가 커지고 비효율적인 질의-응답 과정이 존재하게 된다. 결국 짧은 질의어에 의해 태그 인식이 가능할 때 높은 성능을 보이는 것이다.

그림 1에서 태그 ID가 4비트인 0000, 0001, 1100, 1101 인 4개의 태그에 대해 QT 알고리즘을 적용한 경우의 태그인식 과정을 도식화 한 것이다. 가장 윗줄은 리더기가 태그에 보낸 질의어를 나타내며 두 번째 줄은 응답한 태그들의 충돌 또는 인식 여부를 나타낸다. 알고리즘은 큐가 비어있음으로 브로드캐스팅을 하면서 시작한다. 처음의 경우 모든 태그가 동시에 응답을 함으로 충돌로 인식하고 0과 1을 큐에 저장한다. 다음 단계에서 큐에서 0을 가

표 2.QT 알고리즘의 전체 과정

QT 프로토콜 리더기는 프리픽스를 저장할 큐와 태그ID를 저장할 메모리 공간을 가짐
리더기 동작 Step 1 : 모든 태그에 브로드캐스팅 Step 2 : 태그로부터 응답 확인 - 아무 응답이 없다면 Step 3으로 - 하나의 응답만 있다면 Step 4로 - 동시에 두 개 이상의 태그에서 응답이 온다면 Step 5로 Step 3 : 아무 일도 하지 않음 - Step 7로 Step 4 : 하나의 태그 식별 - 큐가 비었다면 Step 6으로 - 큐가 비어있지 않다면 Step 7로 Step 5 : 충돌 발생 - 기존 프리픽스에 '0'을 추가하여 큐에 저장 - 기존 프리픽스에 '1'을 추가하여 큐에 저장 - Step 7로 Step 6 : 종료 - 모든 태그가 식별 된 상태로 알고리즘 종료 Step 7: 새로운 프리픽스 생성 - 큐에서 값을 빼내어 새로운 프리픽스 생성 - Step 1로
태그 동작 Case1 : 질의어와 매칭 되지 않을 경우 - 매칭 되는 질의어를 수신할 때까지 대기 Case2 : 질의어와 매칭 되는 경우 - 태그ID를 리더기에 전송

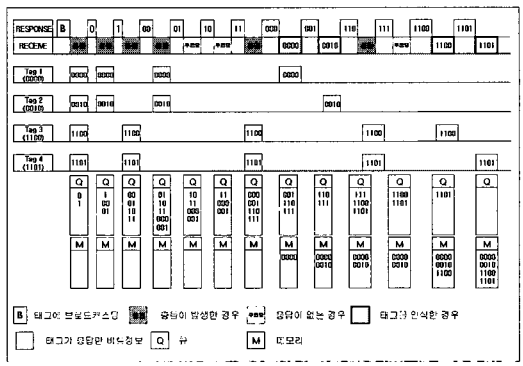


그림 1. QT 알고리즘의 동작 예

져와 질의를 하고 Tag1(0000)과 Tag2(0010)가 동시에 응답하여 충돌이 발생했음으로 0에 0과 1을 추가한 00과 01을 큐에 저장하게 된다. 다음 리더가

1을 질의 하였을 때 충돌이 발생하였음으로 1에 0과 1을 추가하여 큐에 저장한다. 이러한 과정을 반복하다 질의어가 01이 되었을 때는 모든 태그가 응답하지 않았음으로 큐에서 값을 꺼내와 다음 단계를 계속 한다. 질의어가 000이 되었을 때 Tag1만이 응답을 하였음으로 태그ID가 0000인 Tag1을 인식한다. 하나의 태그가 인식되었음으로 큐에서 값을 가져와 다음 질의어로 사용한다. 001을 질의 하였을 때 응답한 태그는 0010인 Tag2 밖에 없음으로 리더는 Tag2를 인식하게 된다. 위의 동작 예를 살펴보면 태그ID가 4비트인 4개의 태그를 인식하는데 반복횟수는 13번, 질의어의 총 비트 수는 31비트, 태그가 총 응답한 응답어의 크기는 72비트인 것을 확인 할 수 있다. 충돌 횟수에 비해 질의어와 응답어의 크기가 급속하게 커지는 이유는 태그가 자신의 모든 ID를 응답하기 때문이다. 위의 예를 살펴보면 QT 알고리즘은 하나의 비트에서 충돌이 발생하여도 전체 알고리즘에서 충돌로 인식하고 과정을 반복하게 된다. 단계 3에서 1을 질의하였을 때 Tag3과 Tag4가 동시에 응답하여 충돌이 발생하고 단계 7에서 11을 질의 하고도 Tag3과 Tag4를 인식하지 못한다. 이는 한 비트 차이의 두 개의 태그를 인식함에 있어 비효율적인 과정을 반복하고 있는 것이다.

2.3 Bit-by-Bit 알고리즘

BBB 이진트리 알고리즘은 Auto ID Class 0에서 정의된 충돌방지 알고리즘이다. BBB 알고리즘은 QT 알고리즘과는 달리 메모리형 알고리즘으로 태그는 태그ID와 태그의 상태를 저장하는 추가적인 메모리를 갖는다. 또한 태그의 응답은 한 비트 만을 응답한다. 따라서 리더기는 각 비트를 저장해야 하는 메모리가 필요하게 된다. 태그를 인식하는 과정은 다음과 같다. 태그의 초기화 과정이 끝나면 리더기가 브로드캐스팅 하는 것으로 알고리즘이 시작된다. 모든 태그들은 자신의 최상위 한 비트(MSB)를 응답한다. 이때 응답은 두가지 경우가 존재하게 된다. 첫 번째는 리더기가 0 또는 1 둘 중 하나만을 수신하는 경우이다. 이때는 충돌이 발생하지 않은 경우로써 리더기는 응답받은 MSB를 메모리에 저장하고 다시 태그에 전송한다. 자신의 첫 비트를 수신한 태그는 다음 비트를 전송한다. 두 번째 경우는 0과 1을 동시에 수신한 경우로 충돌이 발생한 경우이다. 충돌이 발생하게 되면 리더기는 0을 전송한 그룹과 1을 전송한 그룹 중 알고리즘에 의해 하나의 그룹을 선택하고 그 비트를 태그에게 질의한다. 전송받은 태그는 자신의 다음 번째 비트를 다시 리더기에 전송한다. 만일 선택 되지 않은 태그는 일시적으로 태그의 전송 명령에 응답하지 않는 비활성화 상태가 된다. 위와 같은 일련의 과정을 반복할 때마다 리더기의 메모리에는 한 비트씩 수신한 비트를 저장하고 태그ID 길이만큼 반복하였을 때 하나의 태그를 인식하게 된다. 하나의 태그를 인식하게 되면 비활성화 상태의 태그를 다시 활성화 상태로 바꾸게 된다. 따라서 태그ID의 길이가 j이고 태그의 개수가 n개인 환경에서 모든 태그를 인식하기 위한 반복횟수R은 (1)과 같다.

$$R = j \; X \; n \tag{1}$$

브로드캐스팅을 1비트 신호로 간주했을 때 리더기가 태그에 보낸 총 질의어의 크기는 질의-응답 횟수와 동일하게 된다. 또한 BBB 알고리즘의 특성상 인식되는 태그의 순서는 알고리즘에 따라 태그ID가 가장 큰 수인 태그 또는 태그ID가 가장 작은 수인 태그 순으로 인식 된다. 왜냐하면 알고리즘이 충돌 발생시 0의 그룹을 우선 선택하게 되면 가장 작은 태그ID부터 인식되기 때문이다.

그림 2는 태그 ID가 0000, 0010, 1100, 1101 인 4개의 태그에 대한 BBB 알고리즘을 적용하였을 경우의 일련의 태그 인식 과정을 나타낸 것이다. 처음 브로드캐스트를 함으로써 태그 인식을 시작한다. 충돌이 발생한 경우에는 알고리즘에 의해 0을 선택하고 1을 응답한 태그는 비활성화 상태로 만들어 추후 충돌을 회피하게 된다. 한번에 한 비트를 인식하여 태그 ID 길이만큼 질의 하였을 때 하나의 태그를 인식 완료하게 된다. 따라서 질의-응답 반복횟수는 16(4 X 4bit)이다. 질의어의 크기는 한 비트씩

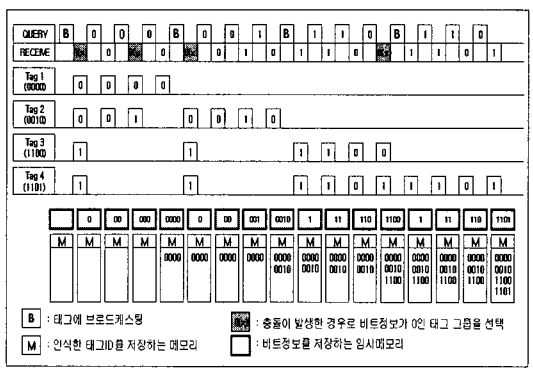


그림 2. BBB 알고리즘의 동작 예

표 3. BBB 알고리즘의 전체 과정

Bit-by-Bit 프로토콜 리더기는 태그로부터 한 비트씩 수신한 응답어를 임시로 저장할 메모리와 인식 완료한 태그ID를 저장할 메모리를 가짐
리더기 동작 Step 1 : 모든 태그에 브로드캐스팅 Step 2 : 태그로부터 응답 확인 - 0 또는 1 둘 중 하나의 신호를 수신한 경우 Step 3으로 - 0과 1을 동시에 수신한 경우 Step 4로 - 태그의 마지막 비트를 수신한 경우 Step 5로 - 모든 태그를 인식 하였다면 Step 6으로 Step 3 : 충돌 미발생 - 메모리에 수신한 비트를 저장 - 수신한 비트를 태그에 질의 - Step 2로 Step 4 : 충돌 발생 - 0을 송신한 그룹을 선택 - 1을 수신한 그룹을 비활성화 상태로 전환 - 메모리에 0을 저장 - 0을 송신 - Step 2로 Step 5 : 하나의 태그 인식 - 비활성화 상태의 태그를 활성화 상태로 전환 - Step 2로 Step 6 : 종료 - 모든 태그가 식별 된 상태로 알고리즘 종료
태그 동작 Case 1 : 태그가 활성화 상태인 경우 - 질의어와 매칭 될 때 - 다음 비트를 송신 - 질의어와 매칭되지 않을 경우 - 대기 Case 2 : 태그가 비활성화 상태인 경우 - 리더의 질의에 응답하지 않음 Case 3 : 비활성화 명령 수신한 경우 - 비활성화 상태로 전환 Case 4 : 활성화 명령 수신한 경우 - 활성화 상태로 전환

만 질의하게 됨으로 브로드캐스팅을 한 비트의 신호라 가정하였을 경우 질의횟수와 같게 된다. 응답어의 크기는 태그ID의 혼잡도가 높을수록 커지게 되는데 위의 경우는 27비트이다. BBB 알고리즘은 태그를 인식함에 있어 모든 태그의 태그ID를 모두 질의하여 태그를 인식 하게 된다. 단계 2에서 0을 수신하고, 단계 3에서 0을 수신하고, 단계 4에서 0을 수신하여 단계 5에서 충돌이 발생하였다. 이는 000을 태그ID로 하는 태그가 2개 존재한다는 것이 된다. 하지만 BBB 알고리즘에서는 이러한 정보를 인지하지 못하고 단지 하나의 태그를 인식하면 인식한 태그를 비활성화 상태로 바꿔 다음번 질의에 응답하지 않게만 하여 충돌을 회피하는 방법이다. 따라서 하나의 태그를 인식 후 처음부터 다시 태그를 인식하는 과정을 반복 하게 되는 것이다. 본 논문에서 제안하는 SBBB 알고리즘은 전 단계에서의 인식정보를 활용하여 필요 없는 질의-응답의 반복을 방지 하고자 한다.

Ⅲ. 제안된 SBBB 알고리즘

BBB 알고리즘의 단점은 이미 읽은 태그 정보를 활용을 전혀 하지 못하고 처음부터 다시 태그 인식을 시작 한다는 것이다. Auto-ID Class 0에서 정의된 BBB 알고리즘은 두 개의 sub-carrier를 사용하여 0과 1을 구분한다. 이는 BBB 알고리즘에서 충돌이 발생한 비트의 위치를 알 수 있다. 이에 본 논문이 제안 하는 알고리즘은 리더기 내에 스택 변수를 두어 태그에서 발생한 충돌의 위치를 저장하고 다음 태그 인식 시에 충돌 위치정보를 활용하는 것이다. 스택의 크기는 전체 태그인식 과정을 봤을 때 그 크기가 아주 작기 때문에 전체 성능에는 크게 지장을 주지 않을 정도이다. 태그를 인식할 때 충돌이 발생하면 해당 태그를 비활성화 상태로 바꾸고 동시에 스택에 해당 충돌 위치를 저장하는 것이다. n비트에서 충돌이 발생 하였다는 뜻은 n-1번째까지 현재 읽은 태그 ID 정보와 동일한 태그가 존재 한다는 것을 의미한다. 따라서 스택에 충돌 위치를 저장하고 하나의 태그 인식이 완료되어 비활성화 상태의 태그를 활성화 상태로 전환할 때 스택에서 n 값을 가져와 현재 메모리에 저장된 데이터에서 n-1 번째까지의 데이터를 다음번 질의어로 사용하는 것이다. 이는 기존에 하나의 태그를 인식하고 다른 태그를 인식할 때 처음부터 다시 앞의 과정을 반복하는 기존 BBB 알고리즘에 비해 불필요한 질의-응답 과정을 줄임으로써 획기적으로 성능 개선이 가능하게 된다. 또한 스택에서 가져온 n 값이 태그 ID의 마지막 부분을 나타낸다면 현재 읽은 태그와 마지막 한 비트만 다른 태그가 존재한다는 것을 의미한다. 다시 말해 마지막 비트가 다른 두 개의 태그가 존재한다면 한번의 질의-응답 과정을

표 4. 제안한 SBBB 알고리즘의 전체 과정

Stack-Bit-by-Bit 프로토콜 리더기는 응답어를 임시로 저장할 메모리, 인식 완료한 태그ID를 저장할 메모리, 충돌위치를 저장할 스택변수를 갖는다.
리더기 동작 Step 1 : 모든 태그에 브로드캐스팅 Step 2 : 태그로부터 응답 확인 - 0 또는 1 둘 중 하나의 신호를 수신한 경우 Step 3으로 - 0과 1을 동시에 수신한 경우 Step 4로 - 태그의 마지막 비트를 수신한 경우 Step 5로 - 아무 응답이 없다면 Step 6으로 Step 3 : 충돌 미발생 - 메모리에 수신한 비트를 저장 - 수신한 비트를 태그에 질의 - Step 2로 Step 4 : 충돌 발생 - 스택에 충돌 위치를 저장 - 0을 송신한 그룹을 선택 - 1을 수신한 그룹을 비활성화 상태로 전환 - 메모리에 0을 저장 - 0을 송신 - Step 2로 Step 5 : 하나의 태그 인식 - 비활성화 상태의 태그를 활성화 상태로 전환 - 스택에 값이 있다면 N값을 가져온다. - 메모리의 N-1번째까지 값을 태그에 질의 - Step 2로 Step 6 : 종료 - 모든 태그가 식별 된 상태로 알고리즘 종료
태그 동작 Case 1 : 질의어와 매칭되는 경우 - 다음 비트를 송신 Case 2 : 질의어와 매칭되지 않는 경우 - 대기 Case 3 : 비활성화 명령을 수신한 경우 - 비활성화 상태로 전환 Case 4 : 활성화 명령을 수신한 경우 - 활성화 상태로 전환

통해 두 개의 태그 인식이 가능하게 된다.

그림 3은 앞서 살펴본 BBB 알고리즘에서 알아본 동일한 조건에서의 태그 인식 과정을 적용해 본 것이다. 시작은 기존과 동일하다. 태그의 초기화 과정이 끝나면 리더기가 브로드캐스팅 하는 것으로 알고리즘이 시작된다. 모든 태그들은 자신의 MSB

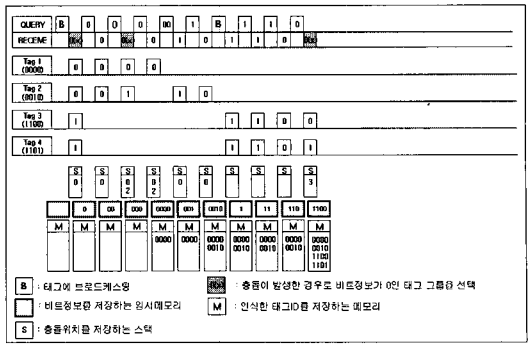


그림 3. SBBB 알고리즘의 동작 예

를 응답한다. Tag1과 Tag2는 0을 응답하였지만 Tag3과 Tag4는 1을 응답하였다. 충돌이 발생한 경우인데 기존에는 1을 응답한 태그를 비활성화 상태로만 두고 과정을 반복하지만 제안한 알고리즘에서는 스택에 충돌 위치를 저장하게 된다. 따라서 스택에 0을 저장하고 알고리즘에 의해 0을 응답한 태그를 선택하고 1을 응답한 Tag3과 Tag4는 비활성화 상태로 변환시켜 다음번 질의-응답 과정에서 응답하지 않게 한다. 0을 인지한 리더기는 0을 메모리에 저장하고 다시 태그에 질의하게 된다. 태그는 자신의 첫 번째 비트와 0이 매칭되게 되면 두 번째 비트를 전송하게 된다. 위의 과정을 반복 하다 3번째 단계에서 충돌이 발생하게 된다. 충돌이 발생하였음으로 스택에 2를 저장한다. 하나의 태그 인식이 끝나면 스택에서 2를 가져와 현재 메모리에 저장된 0000에서 (2-1)번째까지의 태그 ID를 다음번 질의어로 사용하게 된다. 따라서 5번째 질의어는 00이 되고 메모리에는 00이 저장되어 있는 상태가 된다. 00과 매칭이 되는 태그는 다음 한 비트를 전송한다. 10번째 단계에서 하나의 태그를 인식하고 스택에서 값을 가져왔다. 그 값이 태그 ID의 마지막 비트임으로 앞의 3비트는 동일하고 마지막 비트가 다른 즉 0001를 태그 ID로 하는 태그가 존재하는 것을 확인할 수 있다. 따라서 굳이 한번의 인식과정을 더 하지 않고도 태그 인식이 가능하게 된다. 제안된 SBBB 알고리즘에서 질의-응답횟수는 10회이며 총 질의어의 크기는 11비트, 총 응답어의 크기는 19비트이다. 기존의 알고리즘들에 비해 성능개선이 뚜렷함을 확인 할 수 있다.

Ⅳ. 실험 및 평가

4.1 시뮬레이션 환경 결정

EPCgolbal에서 제안하는 EPC 코드는 사용 목적

이나 응용분야에 따라 코드크기나 인코딩 방식이 다르다. 본 논문에서는 일반적인 인코딩 방식인 GID-96 인코딩 방식을 따르는 태그를 선정하였다. GID-96 방식에서 태그의 길이는 96비트로 4개의 부분으로 구성되어 있다. Header는 인코딩방식을 결정하는 부분으로 8비트로 구성되며, General Manager Number는 28비트로 영역관리자를 Object Class는 24비트로 상품 코드를 마지막으로 Serial Number는 36비트로 각 상품의 시리얼 넘버를 나타내게 되어 있다.

표 5는 인코딩 방식을 나타내며 표 7은 인코딩 방식에 따른 Header 값을 나타낸다. 그림 4는 EPCglobal의 GID-96 태그 구조를 나타낸다.

물류 산업의 특성상 상품은 한곳에 머무는 것이 아니라 계속해서 이동과 보관이 반복되는 특징이 있다. 따라서 현재 상품의 위치를 실시간으로 추적할 수 있는 Traceability의 확보와 전체 공급사슬의 실시간 Visibility을 가능하게 하는 것이 RFID 기술이다. 이에 물류산업에 RFID 기술을 활용하기 위한 연구가 활발히 진행 중이다. 물류 산업에 RFID 기술을 활용하기위해서는 우선 모든 상품에 태그를 부착한다. 재조공장에서 동일한 상품에 태그를 부착하면 태그ID값의 Header, General Manager Number, Object Class까지 동일하고 Serial Number는 순차적이게 된다. 또한 여러 상품이 혼재되어 포장된 상태로 이동 할 때도 태그ID의 중복성이 발생

표 5. 태그 코드의 인코딩 방식[15]

Code Scheme		Explanation
1	GID	General Identifier
2	GTIN	Serialized version of the EAN.UCC Global Trade Item Number
3	SSCC	EAN.UCC Serial Shipping Container Code
4	GLN	EAN.UCC Global Location Number
5	GRAI	EAN.UCC Global Returnable Asset Ldentifier
6	GIAI	EAN.UCC Global Individual Asset Identifier

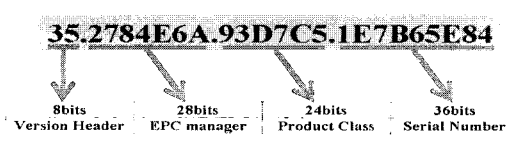


그림 4. EPCglobal의 GID-96 태그 구조[3]

표 6. 인코딩 방식에 따른 Header 값[15]

Header Value	Tag Lenth(bits)	EPC Encoding Sscme
10	64	SGTIN-64
0000 1000	64	SSCC-64
0000 1001	64	GLN-64
0000 1010	64	GRAI-64
0000 1011	64	GIAI-64
0011 0000	96	SGTIN-96
0011 0001	96	SSCC-96
0011 0010	96	GLN-96
0011 0011	96	GRA-96
0011 0100	96	GIAI-96
0011 0101	96	GID-96

하게 된다. 본 논문은 이러한 상황을 고려하여 태그 ID의 Serial Number가 순차적인 경우와 랜덤 한 경우를 가정하여 시뮬레이션을 진행하였다.

성능 측정 기준은 하나의 태그를 인식하기 위해 태그와 리더사이에 질의-응답 횟수와 리더기가 모든 태그를 인식할 때까지의 총 질의어의 크기, 태그가 리더기의 질의어와 매칭이 되었을 경우 이에 응답하는 태그들의 총 응답어의 크기를 기준으로 성능을 측정 하였다.

시뮬레이션은 C언어 기반으로 설계하였으며 태그의 수는 100개부터 500개까지 50개씩 증가시켜가며 성능을 측정하였다.

4.2 Serial Number 36비트가 랜덤한 경우

Head, General Manager Number, Object Class까지 동일하고 마지막 Serial Number만 랜덤 한 경우로 제조공장, 유통업체, 도매상과 같은 곳에서 제품을 팔렛트 단위로 운반할 때를 가정하고 실험을 실시하였다.

그림 5를 보면 BBB 알고리즘의 경우 증가량이 일정한 것을 확인 할 수 있다. BBB 알고리즘의 경우 충돌을 전혀 고려하지 않는다. 따라서 하나의 태그를 인식하기 위해 96번의 질의-응답과정을 반복하게 된다. 결국 충돌 횟수는 태그ID 길이에 태그 개수를 곱한 값이 된다. SBBB 알고리즘이 반복 횟수 면에서 기존의 BBB 알고리즘에 비해 현격한 성능 향상을 확인 할 수 있다. 이는 SBBB 알고리즘의 경우 충돌이 발생한 위치를 스택에 저장하고 다음 태그 인식을 할 때 그 정보를 활용하여 질의어를 만들고 이를 태그에 질의하기 때문에 중복된 질의-응답 과정을 현저히 줄어들게 되어 위와 같은 성능개선을 가져온 것이다. SBBB 알고리즘이 QT 알

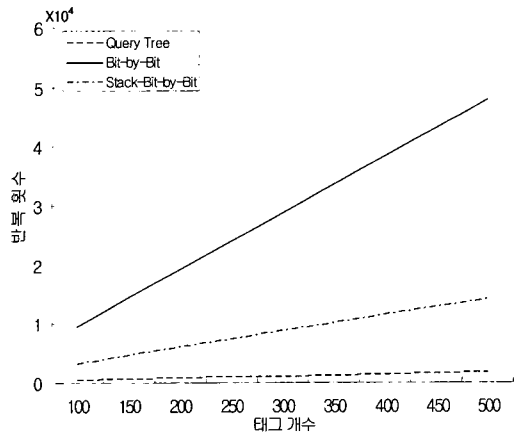


그림 5. Serial Number가 랜덤인 경우 반복 횟수

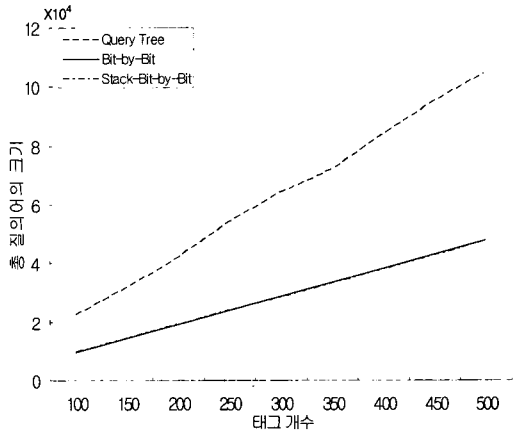


그림 6. Serial Number가 랜덤인 경우 총 질의어의 크기

고리즘에 비해 반복 횟수 면에서 성능이 낮게 나오는데 이는 태그를 인식하는 메커니즘이 다른데서 기인한다. QT의 경우는 질의어의 길이를 길게 하여 질의어가 유일한 태그를 인지 가능하게 되면 태그를 인식하게 된다. 따라서 태그 ID가 랜덤하여 상대적으로 짧은 질의어를 이용하여 태그 인식이 가능하다. 반면 SBBB 알고리즘은 태그로부터 한 비트씩 태그ID를 수신받기 때문에 기본적으로 96번의 질의-응답 과정을 거쳐야 태그 인식이 가능한 구조이다. 36비트 Serial Number가 랜덤한 경우는 상대적으로 태그ID의 중복성이 낮은 경우이기 때문에 QT에 비해 성능이 낮게 나오는 것을 확인 할 수 있다.

그림 6은 총 질의어의 크기 면에서 성능을 비교한 결과이다. QT는 질의어의 길이를 길게 하여 태그를 인식하는 방법이기 때문에 질의어의 길이가 길어 질 수밖에 없다. 현재의 상황은 60비트가 동일한 태그들을 인식 하는 경우임으로 질의어를 단계적으로 증가시켜 결국 60비트 이상이 되어야만 태그를 인식 이 가능하다. 이는 60비트 이상의 질의가 계속 해서 태그에 질의된다는 것을 의미한다. 이런 질의어의 크기의 증가는 태그ID 길이가 길어 질수록크게 나타나게 된다. 반면 BBB 알고리즘과 SBBB 알고리즘은 QT에 비해 상대적으로 좋은 결과를 보이고 있다. 이 역시 QT 알고리즘과의 태그 인식 방법에서 오는 차이에서 기인한다. BBB 알고리즘의 경우 질의어는 오직 한 비트이며 이를 태그에 질의하기 때문에 태그ID 길이가 길어 져도 질의어의 길이가 길어 지지 않는다. 따라서 QT에 비해 높은 성능을 보이고 있다. BBB 알고리즘과 SBBB 알고리즘의 경우 비슷한 성능을 보이고 있다. SBBB 알고리즘은 충돌 위치를 활용하여 질의어를 달리 하여 질의-반복 횟수를 줄이게 된다. 하지만 질의어의 크기는 한 비트이상이다. 물론 BBB 알고리즘에 비해 SBBB 알고리즘의 경우 브로드캐스팅하는 과정이 줄어들지만 줄어든 질의-반복 횟수와 질의어의 크기의 증가가 비슷하기 때문에 다음과 같은 결과를 보이고 있다.

그림 7은 총 응답어의 크기 면에서 성능을 비교해본 것으로 QT는 태그 개수에 비례하여 증가하는 것을 확인 할 수 있다. QT에서 태그의 응답은 질의어와 매칭이 되면 자신의 모든 태그ID 정보를 송신한다. 반면 한 비트만 응답하는 BBB 알고리즘에 비해 태그수가 적을 경우 BBB 알고리즘이 좋은 성능을 보인다. 하지만 태그 개수가 증가하여 태그 ID의 혼잡도가 높아지면 BBB 알고리즘은 질의어와 매칭되는 태그가 QT에 비해 상대적으로 크다. 따라서 응답하는 태그의 수도 많아지는 것이다. 이러한 현상은 태그 개수가 증가 할수록 커져 태그 수가 증가 할수록 급속도로 응답어의 크기가 커지는 것을 확인 할 수 있다. 반면 QT 알고리즘의 경우 질의어가 길어지면서 질의어와 매칭되는 태그 또한 적어지게 되어 전체적인 총 응답어의 크기는 태그 개수에 비례하여 증가한다. SBB 알고리즘은 QT나 BBB 알고리즘에 비해 현저한 성능 개선을 확인 할 수 있는데 SBBB 알고리즘은 태그ID의 혼잡도가 높아져도 태그의 응답어가 한 비트이고 질의어와 매칭되는 태그가 상대적으로 적으며 반복 횟수 또한 BBB 알고리즘에 비해 적기 때문에 가장 좋은 성능을 보이고 있다.

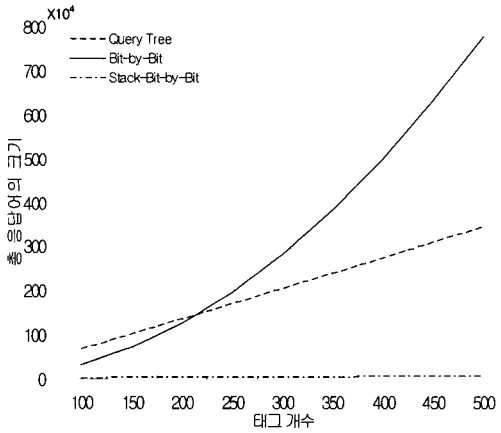


그림 7. Serial Number가 랜덤인 경우 총 응답어의 크기

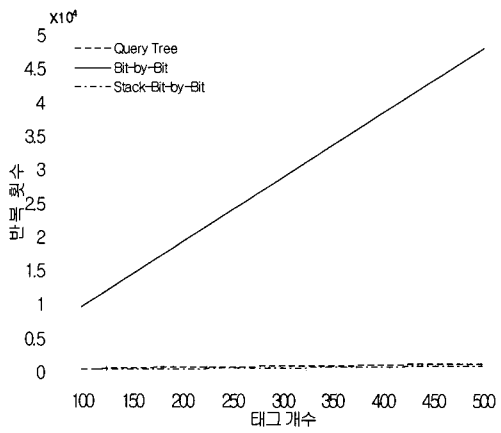


그림 8. Serial Number가 순차적인 경우 반복 횟수

4.2 Serial Number가 순차적인 경우

그림 8과 그림 9는 Serial Number가 순차적인 경우의 반복 횟수를 보이고 있다. 그림 9는 QT 알고리즘과 SBBB 알고리즘만을 표시하여 두 알고리즘의 성능을 비교 한 것이다. BBB 알고리즘의 경우 태그ID와 상관없이 태그를 인식하기 위한 반복 횟수는 태그ID 길이와 태그 개수를 곱한 값으로 일정한 증가를 보이고 있다. Serial Number가 순차적으로 증가하기 때문에 태그ID의 대부분의 비트가 동일함으로 다른 상황에 비해 태그 중복성이 가장 큰 상황이다. QT 알고리즘의 경우 태그ID가 순차적임으로 큐의 질의어를 질의 하였을 때 무응답이 발생하는 빈도가 크게 감소한다. 따라서 Serial Number가 랜덤한 경우에 비해 순차적인 경우 성능이 개선된 것을 확인 할 수 있다. SBBB 알고리즘의 경우도 역시 Serial Number가 순차적임에 따라 태그ID의 중복성이 커져 Serial Number가 랜덤한 경우에 비해 성능이 개선된 것을 확인 할 수 있다. 특히 Sirial Number가 순차적임으로 마지막 한 비트가 다른 태그가 2개씩 존재하게 된다. 따라서 하나의 태그 인식과정을 통해 2개의 태그 인식이 가능하게 된다. 시뮬레이션 결과 QT 알고리즘에 비해 SBBB 알고리즘의 성능이 평균 31.7%의 개선된 것을 확인 할 수 있었다.

그림 10을 보면 총 질의어의 크기는 태그 개수의 증가에 비례하여 증가하는 것을 확인 할 수 있다. 앞서 설명한 것과 마찬가지로 QT 알고리즘의 경우 질의어가 점점 길어져 유일한 태그를 인식 가능할 때 하나의 태그 인식이 완료된다. 현재의 경우 태그ID가 순차적으로 증가함으로 96비트에 가까운 질의

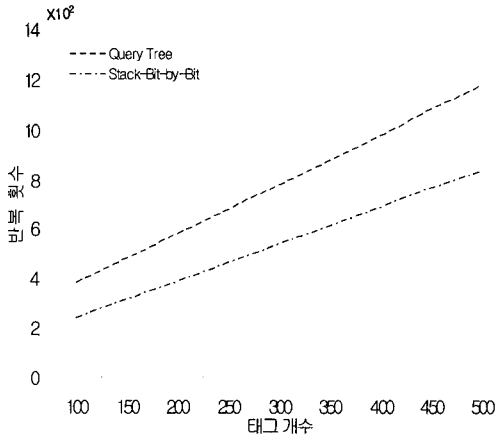


그림 9. Serial Number가 순차적인 경우 반복 횟수

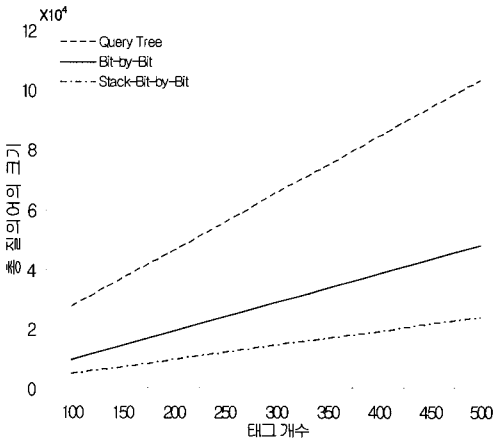


그림 10. Serial Number가 순차적인 경우 총 질의어의 크기

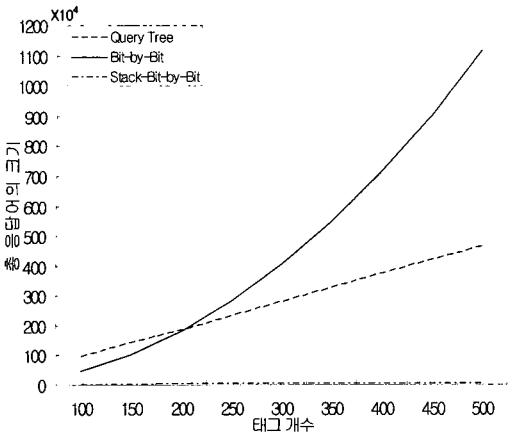


그림 11. Serial Number가 순차적인 경우 총 응답어의 크기

어를 질의해야 만이 하나의 태그 인식이 가능하다. 따라서 질의어의 총 크기는 가장 크게 나타나며 SBBB 알고리즘이 BBB 알고리즘에 비해 반복 횟수가 적음으로 가장 좋은 성능을 보이고 있다.

그림 11은 태그ID가 순차적일 때 태그들이 응답한 총 응답어의 크기를 나타내고 있다. Serial Number가 랜덤한 경우와 유사한 결과를 보이고 있다. QT 알고리즘과 BBB 알고리즘의 경우 Serial Number가 랜덤한 경우에 비해 총 응답어의 크기가 증가한 것을 확인 할 수 있다. 이는 태그ID의 중복성이 증가함으로서 질의어와 매칭되는 태그가 증가하게 되어 응답한 태그가 증가하고 따라서 응답어의 크기 또한 증가하였다. 하지만 SBBB 알고리즘의 경우 태그ID의 중복도가 커짐으로 총 응답어의 크기 또한 증가하지만 태그ID가 순차적임으로 한번의 태그 인식 과정으로 두 개의 태그 인식이 가능하게 됨으로 인해 전체적인 성능은 다소 개선된 것을 확인 할 수 있었다.

V. 결 론

본 논문은 Auto-ID Class 0에서 정의된 BBB 알고리즘의 성능을 개선한 SBBB 알고리즘을 제안하였다. SBBB 알고리즘은 스택을 이용하여 충돌 위치를 저장하고 이를 활용하여 BBB 알고리즘의 성능을 개선한 알고리즘이다. SBBB 알고리즘의 성능 향상을 좀 더 객관적으로 나타내고 활용가능성을 증명하기 위해 QT 알고리즘과도 자세히 비교해 보았다. EPCglobal에서 표준으로 채택한 태그를 사용하여 물류환경을 감안한 시뮬레이션 결과 태그ID와 상관없이 모든 경우에서 SBBB 알고리즘이 BBB 알고리즘에 비해 성능이 개선된 것을 확인 할 수 있었다. 특히 태그ID의 중복성이 크고 태그의 개수가 증가할수록 좋은 성능을 확인 할 수 있었는데 이는 실제 물류 환경에서 태그ID의 중복성을 고려하였을 때 SBBB 알고리즘의 활용 가능성이 크다고 할 수 있겠다.

참 고 문 헌

[1] 이현지, 김종덕, "충돌 비트 위치를 활용한 RFID 다중 태그 인식 알고리즘," 한국통신학회논문지, Vol. 31, No. 4A, December 2005.

[2] Auto-ID Center, "Draft Protocol Specification for a Class 0 Radio Frequency Identification tag," 2003.

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

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

[5] 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.

[6] Jacomet M, Ehrsam A, Gehrig U, "Contactless identification device with anti-collision algorithm. IEEE Computer Society," CSCC '99, Conference on Circuits, Systems, Computers and Communications, Athens. 4-8 July 1999.

[7] Law, Ching, Lee, Kayi and Siu, Kai-Yeung. "Efficient Memoryless protocol for Tag Identification," In Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Pages 75-84. ACM. August 2000.

[8] 권성호, 박경랑, 김회철, "충돌 추적 기법을 적용한 다중 태그 식별 알고리즘의 설계 및 구현," 대한전자공학회하계종합학술대회, Vol. 27, No.

1, Pages 679-682. June 2004.

[9] Auto-ID Center, "Draft Protocol Specification for a Class 1 Radio Frequency Identification tag," 2003.

[10] Vogt, H. "Efficient Object Identification with Passive RFID Tags," In International Conference on Pervasive Computing LNCS. Springer-Verlag. 2002.

[11] Auto-ID Center. "13.56MHz ISM Band Class 1Radio Frequency Identification Tag Interface Specification: Candidate Recommendation, Version 1.0.0," Auto-ID Center. May 1, 2003.

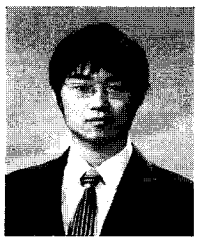
[12] Changsoon Kim, Kyunglang Park, Hiecheol Kim, Shindug Kim, "An Efficient Stochastic Anti-collision Algotithm using Bit-Slot Mechanism," PDP'2004, July 2004.

[13] ISO/IEC 18000-6 : 2003(E), Part 6 : Parameters for air interface communications at 860-960 MHz, Nov. 26, 2003.

[14] Jae-Ryong Cha, Jae-Hyun Kim, "Dynamic framed slotted ALOHA algorithms using fast tag estimation method for RFID system," Consumer Communications and Networking Conference, 2006. CCNC 2006. 2006 3rd.

[15] EPCglobal, "EPCglobal Tag Data Standards Version 1.3," Ratified Specification, March 2006.

이 재 구 (Jae-ku Lee) 준회원



2001년 3월 ~ 현재 : 인천대학교 컴퓨터공학과 학사과정

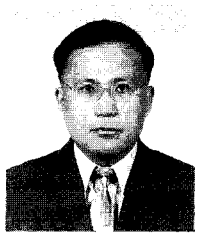
<관심분야> RFID, Sensor Network

유 대 석 (Dae-suk Yoo) 준회원

2001년 3월 ~ 현재 : 인천대학교 컴퓨터공학과 학사과정

<관심분야> Sensor Network, RFID

최 승 식 (Seung-sik Choi) 정회원



1988년 2월 : 연세대학교 전자공학과 학사

1990년 2월 : KAIST 전기 및 전자공학과 석사

2002년 2월 : KAIST 전기 및 전자공학과 박사

1990년 3월 ~ 1998년2월 KT 통신망연구소 전임연구원

2002년 ~ 2004년 KT서비스개발연구소 선임연구원

2004년 ~ 현재 인천대학교 컴퓨터공학과 조교수

<관심분야> 무선엑세스제어, 무선자원관리, 무선인터넷 프로토콜