논문 13-38B-06-03

한국통신학회논문지 '13-06 Vol.38B No.06
http://dx.doi.org/10.7840/kics.2013.38B.6.435

연속적인 태그 ID들을 위한 *M*-ary 쿼리 트리 알고리즘의 향상에 관한 연구

양 동 민◆, 신 종 민°

EMQT : A Study on Enhanced *M*-ary Query Tree Algorithm for Sequential Tag IDs

Dongmin Yang◆, Jongmin Shin°

요 약

RFID 및 많은 관심을 받고 있는 NFC 시스템에서 가장 중요한 이슈 중에 하나가 리더의 전송범위 내에 존재하는 다수의 태그들을 정확하고 빠르게 식별하는 충돌 방지 기법이다. 확률적인 충돌 방지 기법들과는 달리, 쿼리 트리 기반의 프로토콜들은 리더의 전송범위 내에 존재하는 모든 태그들을 식별하는 것을 보장한다. 이러한 프로토콜에서는 태그 ID들이 균등 분포를 이루고 있다는 가정을 한다. 그러나 실제 RFID 제품들을 제작할 때에는, 각각의 태그 ID의 프리픽스는 EPCglobal에서 유일하게 정해지고 태그 ID의 나머지 부분은 회사 또는 제조업체에 의해서 ID의 값을 하나씩 증가시키면서 연속적으로 할당된다. 본 논문에서는 향상된 *M*-ary 쿼리 트리 기법의 개선 방안(Enhanced *M*-ary Query Tree: EMQT)을 제안한다. EMQT는 *m*-비트인식 및 예측 기반의 알고리즘을 이용하여 연속적인 태그 ID들을 식별할 때, 불필요한 질의-응답 횟수를 효과적으로 줄인다. 그리고 이론적 추정을 통한 수학적 분석과 시뮬레이션을 통한 실험적 분석을 통해 EMQT가 다른 기법들보다 식별시간, 식별효율, 통신비용 관점에서 훨씬 성능이 우수하다는 것을 보여준다.

Key Words : RFID, anti-collision, query tree, aloha protocol, M-ary tree

ABSTRACT

One of the most challenging issues in radio frequency identification (RFID) and near field communications (NFC) is to correctly and quickly recognize a number of tag IDs in the reader's field. Unlike the probabilistic anti-collision schemes, a query tree based protocol guarantees to identify all the tags, where the distribution of tag IDs is assumed to be uniform. However, in real implements, the prefix of tag ID is uniquely assigned by the EPCglobal and the remaining part is sequentially given by a company or manufacturer. In this paper, we propose an enhanced M-ary query tree protocol (EMQT), which effectively reduces unnecessary query-response cycles between similar tag IDs using m-bit arbitration and tag expectation. The theoretical analysis and simulation results show that the EMQT significantly outperforms other schemes in terms of identification time, identification efficiency and communications overhead.

※ 이 논문은 2011학년도 대전대학교 신진교수연구장려금에 의해 지원되었음.
※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2012R1A1A1042703)
◆ 주저자 : 대전대학교 정보통신공학과 무선이동통신 연구실, dmyang@dju.kr, 정회원
° 교신저자 : 삼성전자 VD사업부, jm0621.shin@samsung.com, 정회원
논문번호 : KICS2013-01-047, 접수일자 : 2013년 1월 16일, 최종논문접수일자 : 2013년 5월 8일

Ⅰ. 서 론

RFID(Radio Frequency IDentification)는 사물에 전자 태그(tag)를 부착하고 무선통신 기술을 이용하여 태그의 정보를 읽어와 물체를 식별하는 비접촉 근거리 무선인식 기술이다[1]. 이러한 RFID 기술은 물류, 유통, 금융 분야에서 다양한 서비스를 제공할 것으로 기대되고 있다. 그러나 RFID 기술은 산업 전반에 걸쳐 적용되지 못하고 건물 출입 통제나 대중 교통 결제 수단 등 아직까지는 극히 제한된 영역에만 적용되고 있다. 이러한 RFID 기술의 보다 폭넓은 대중화 및 산업화를 이루기 위해서는 물류 및 유통 분야 같은 기반 산업 분야에 적용되어야 활성화될 수 있다. 이를 위해서는 몇 가지 해결되어야 할 문제들이 있다. 그 중 가장 큰 문제 중 하나가 다중 태그 식별 문제이다. 다중 태그 식별 문제는 리더가 여러 개의 태그 정보를 읽기 위해서 질의(query)를 전송하고 이 질의를 수신한 태그들이 동시에 응답(response)할 때, 전파 사이에 충돌(collision)이 발생하여 리더가 태그들의 정보를 식별하는데 실패하는 문제이다. 이러한 문제를 해결하기 위해서 다양한 태그 충돌 방지 기법(anti-collision)들이 빠르고 정확하게 여러 태그들의 정보를 식별하는 것을 목표로 제안되어 왔다.

태그 충돌 방지 기법은 크게 확률적인(probabilistic) 방법과 결정적인(deterministic) 방법의 2가지로 구분된다. 확률적인 방법에서는 알로하(Aloha) 기반 프로토콜[1-4]이 대표적이고, 결정적인 방법에서는 트리(tree) 기반 프로토콜[5-10]이 대표적이다. 알로하 기반 프로토콜에서는 시간을 슬롯(slot) 단위로 나누고 태그들이 임의로 하나의 슬롯을 선택하여 응답한다. 만약 해당 슬롯에서 응답하는 태그가 하나만 존재한다면 리더가 태그의 정보를 인식할 수 있다. 대표적인 방법으로 슬롯 알로하 프로토콜이 있다. 이 방식은 본질적으로 확률의 불확실성을 가지고 있기 때문에 리더의 전송범위 내에 있는 모든 태그들을 식별하는 것을 보장하지 못한다. 트리 기반 프로토콜은 태그 ID 정보를 얻기 위해서 여러 번의 질의-응답 과정을 통해서 태그를 식별한다. 이 때, 이 태그 식별 과정이 트리 형태로 표현된다. 이 방식은 리더의 전송범위 내에 있는 모든 태그를 인식하는 것을 보장하지만, 유사한 ID를 갖는 태그들이 다수 존재할 경우에는 충돌이 빈번히 발생하고 질의-응답 횟수가 많아져서 모든 태그들을 식별하는데 시간이 오래 걸릴 수 있다. 본 논문에서는 리더의 전송 범위 내에 존재하는 모든 태그들을 인식할 수 있는 결정적 방법 즉, 트리 기반 프로토콜의 식별시간을 줄이는 방안에 초점을 맞춘다.

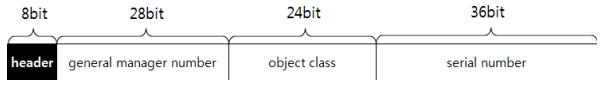

그림 1. 태그 데이터 구조 (GID-96)
Fig. 1. Tag Data Structure (GID-96)

EPC 태그 데이터 표준[11] 에 따르면, 데이터의 엔코딩 방식에 따라 비트 길이와 용도는 조금 다를 수 있지만 그림 1의 GID-96 인코딩 방식과 유사하다. GID-96 인코딩 방식은 총 96 비트로 하나의 헤더(header) 필드와 세 개의 데이터 필드로 구성되어 있다. 헤더(8비트)는 태그 정보의 길이와 어떤 인코딩 기법으로 코딩되어 있는지를 나타내고, GID-96 인코딩 경우에는 "0011 0101"이다. 일반관리번호(general manager number)는 회사, 관리자 또는 조직을 나타내는 숫자로써 EPCglobal에 의해서 정해진다. 일반관리번호 다음에 오는 객체클래스(object class)와 일련번호(serial number)는 일반관리번호에 해당하는 회사가 정할 수 있다. 객체 클래스는 제품의 분류를 나타내고, 일련번호는 객체클래스 내에서 반드시 유일하도록 정해져야 한다. 이 때 일련번호는 일반적으로 제품이 생산될 때 차례대로 하나씩 증가하면서 각 제품들에게 부여된다. 만약 같은 회사 또는 유사한 분류의 제품들이면, 그들의 태그 ID는 비슷할 것이다. 제품 생산 직후, 물류 유통 과정에 있는 같은 회사 유사한 분류의 제품들이 함께 모여 있는 제품 창고나 쇼핑 센터에는 헤더, 일반관리번호, 객체클래스는 똑같고 일련번호가 하나씩 증가하는 수많은 태그 ID들로 이루어져 있을 것이다. 이 경우에 기존의 트리 기반의 충돌 방지 프로토콜들은 태그 정보를 식별할 때 질의에 응답하는 태그의 수가 하나일 때만 태그를 식별한 것으로 판단한다. 그렇기 때문에 반드시 하나의 태그가 응답할 때까지 질의-응답 과정을 계속 진행하여 식별 시간이 상당히 증가될 수 있다. 이러한 트리 기반의 충돌 방지 프로토콜은 트리를 구성하는 노드들이 각각 하나의 질의-응답 과정에 해당한다. 때문에 트리의 깊이(depth)를 하나를 감소하면 이진 트리(binary tree)의 경우에는 전체 질의-응답 횟수를 절반 정도 줄이는 효과를 얻을 수 있다. 만약, M-ary 트리일 경우에는 1/M만큼 질의-응답 횟수를 줄일 수 있다. 본 논문에서는, M-ary 트리를 이용하여 m-비트인식(m-bit arbitration) 방식을 제공하고

있는 *M*-ary 트리 검색 기법(*M*-ary Query Tree Scheme: MQT)[10]에 MQT의 매핑함수로부터 태그 ID를 예측 하는 알고리즘을 적용하여 개선된 *M*-ary 쿼리 트리 기법의 개선 방안(Enhanced *M*-ary Query Tree: EMQT)을 제안한다. (단, 2^m=*M*이다.) EMQT는 연속적인 태그 ID들을 식별할 때, 불필요한 질의-응답 횟수를 상당히 효과적으로 줄인다.

본 논문은 다음과 같이 구성되어 있다. 1장에서는 본 논문의 개요에 대해서 간략히 소개했다. 2장에서는 관련 연구들과 그들의 장단점을 분석한다. 3장에서는 본 논문에서 제안하는 EMQT에 대해서 자세히 설명한다. 4장에서는 본 논문에서 제안하는 EMQT와 기존의 다른 알고리즘을 이론적인 분석과 실험적인 분석을 통해서 성능을 비교하고, 5장에서는 결론을 내린다.

Ⅱ. 관련 연구

2.1. 슬롯 알로하 (Slotted Aloha: SA)

SA에서는 시간을 고정된 몇 개의 슬롯(slot)으로 나누고 태그들이 임의로 하나의 슬롯을 선택한 후에 해당 슬롯에서 자신의 고유 태그 ID를 전송하는 방식이다. 이 방식에서는 오직 하나의 태그만이 자신의 ID를 슬롯 내에 충돌 없이 리더에게 전송할 경우에만 태그를 인식할 수 있다. SA 중에서 가장 대표적인 방식은 프레임 슬롯 알로하 프로토콜(Framed Slotted Aloha Protocol: FSA)[12]이다. 그림 2는 FSA가 동작하는 과정을 보여 준다. 하향링크(Downlink)는 리더가 사용하는 채널이고, 상향링크(Uplink) 태그들이 사용하는 채널이다. 그리고, 하나의 프레임(frame)은 세 개의 슬롯으로 구성된다. 리더가 첫 번째 프레임이 시작할 때 요청 메시지를 전송하면, 첫 번째 슬롯에서 태그1과 태그3이, 두 번째 슬롯에서는 태그2와 태그5가 동시에 응답하여 충돌이 발생하고, 세 번째 슬롯에서는 태그4 하나만 응답하여 리더에 의해 인식된다. 두 번째 프레임에서는 인식된 태그4를 제외한 네 개의 태그들이 같은 방법으로 슬롯을 임의로 선택하여 응답한다. 두 번째 프레임에서는 첫 번째 슬롯에서 태그2와 태그3 충돌이 발생하고 태그1과 태그5가 인식된다. 이 방법은 본질적으로 확률의 불확실성을 가지고 있기 때문에 리더의 전송범위 내에 있는 태그들을 100% 인식하는 것을 보장 못한다는 단점을 가지고 있다.

2.2. 쿼리 트리 (Query Tree: QT)

		1	2	3		1	2	3
Downlink	Request				Request			
Uplink		Collision	Collision	Identification		Collision	Identification	Identification
Tag1(0001)		~~0001~~					**0001**	
Tag2(0010)			~~0010~~			~~0010~~		
Tag3(0011)		~~0011~~				~~0011~~		
Tag4(1110)				**1110**				
Tag5(1111)			~~1111~~					**1111**

Frame

그림 2. 프레임 슬롯 알로하 프로토콜의 태그 식별과정
Fig. 2. Tag Identification in Framed Slotted Aloha Protocol

QT[5]는 결정적인(deterministic) 방식으로 리더의 전송 범위 내에 존재하는 모든 태그들을 인식하는 것을 보장한다. QT의 식별 과정은 리더가 질의(query)하고 태그가 응답(response)하는 질의-응답 방식으로 진행된다. 리더는 비트 스트링(string)으로 이루어진 프리픽스(prefix)를 질의-큐(query-queue)에서 꺼내서 태그들에게 질의한다. 이 질의를 수신한 태그는 프리픽스가 자신의 ID의 앞부분과 일치하면 자신의 ID를 가지고 리더에게 응답한다. 만약, 두 개 이상의 태그들이 응답하여 충돌이 발생하면, 리더는 질의한 프리픽스에 '0'과 '1'하나의 비트를 추가하여 질의-큐에 넣는다. 리더는 다시 질의-큐에서 프리픽스를 꺼내어 질의-응답 과정을 반복한다. 이 과정은 질의-큐에 더 이상 프리픽스가 존재하지 않을 때까지 계속된다. 처음에 질의-큐에는 ε으로 초기화되어 있고, ε을 수신한 태그들은 프리픽스 비교 없이 자신의 ID로 리더에게 응답한다.

이러한 식별 과정은 그림 3과 같이 트리 형태로 표현될 수 있다. 그림 3은 5개의 태그(0001, 0010, 0011, 1110, 111)들이 식별되는 과정을 보여 준다. 트리를 구성하는 노드들은 각각 하나의 질의-응답과정을 나타낸다. 각 노드는 질의-응답의 결과에 따라 충돌 노드(collision node), 확인 노드(identification node), 유휴 노드(idle node) 로 구분된다. 충돌 노드는 리더의 질의에 하나 이상의 태그들이 응답했을 때를 나타내고, 확인 노드는 오직 하나의 태그만 응답했을 때, 유휴 노드는 아무 태그도 응답하지 않을 경우를 나타낸다. 노드 안에 표시되어 있는 숫자는 질의-응답의 순서를 나타낸다. 만약, 리더의 전송범위 내에 똑같은 태그들이 존재한다면 항상 태그 식별 과정은 같은 모양의 트리를 갖는다. 그림 3에서의 질의-응답 횟수는 15번이다.

QT는 리더의 전송범위 내에 존재하는 모든 태그들의 식별을 보장하지만, 한 번에 하나의 비트만 식별할 수 있고, 응답 메시지가 충돌한 경우에 충돌한 태그들의 정보를 확인할 수 없기 때문에 불필요한

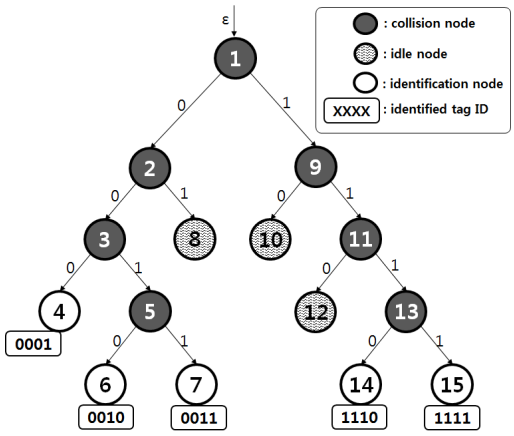

그림 3. QT의 태그 식별 과정
Fig. 3. Tag Identification in QT

질의-응답 과정인 유휴 노드가 추가되어 식별하는 상당히 시간이 길어질 수 있다.

2.3. 맨체스터 코딩 (Manchester Coding)

QT에서는 두 개 이상의 태그들이 응답할 경우에 충돌 여부만 알 수 있지 충돌을 발생시킨 태그들의 정보는 전혀 얻을 수 없다. 디지털 신호를 인코딩하는 방식 중에 하나인 맨체스터 코딩[1]을 이용하여 충돌을 발생시킨 태그들의 정보를 얻을 수 있다. 맨체스터 코딩 방식에서 하나의 비트는 전압의 값(value)가 아니라 전압의 전이(transition)으로 표현된다. '0' 비트는 양의 전이(positive transition)으로 표현되고, '1' 비트는 음의 전이(negative transition)으로 표현된다. 만약 두 개 이상의 태그들이 동시에 응답한다면, 양의 전이와 음의 전이는 서로 상쇄되어 맨체스터 코딩에서는 허용되지 않는 무전이(no transition) 상태가 되어 에러가 발생한 것을 알 수 있다. 이러한 현상은 맨체스터 코딩을 이용해서 어느 비트에 충돌이 발생했는지를 추적할 수 있다. 그림 4는 두 개의 태그 Tag1과 Tag2가 동시에 응답했을 때, 리더가 수신하게 되는 응답 메시지가 어떻게 생성되는지를 보여 준다. 양의 전이와 음의 전이가 서로 상쇄되어 무 전이를 발생시켜 어떤 비트에 충돌이 발생했는지에 대한 정보를 알 수 있다.

2.4. 충돌 트리 (Collision Tree: CT)

QT을 기반으로 하는 CT[9]은 맨체스터 코딩 기법으로부터 얻은 최초 충돌 비트(first collided bit) 정보를 이용하여 충돌노드 개수를 줄이고 유휴노드를 제거해서 QT의 식별시간을 획기적으로 감소시켰다. 태그들이 응답할 때 자신의 태그 ID 전체로 응답하는 QT과 달리, 리더가 전송한 프리픽스 부분을

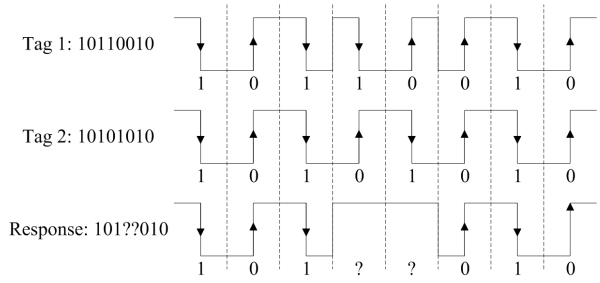

그림 4. 맨체스터 코드 적용 시 태그 응답 및 충돌
Fig. 4. Collision behaviors for Manchester code

제외한 나머지 태그 ID로만 응답한다. 이 때, 맨체스터 코딩을 이용하여 태그들이 전송한 응답에서 어떤 비트에 충돌이 발생했는지 여부를 알 수 있다. 때문에, 여러 태그가 동시에 응답했을 경우, 리더가 수신하는 응답은 '0', '1' 그리고 충돌을 의미하는 '*'로 비트들을 나타낼 수 있다. 충돌 발생 시 리더가 질의한 프리픽스에 '0'과 '1'을 추가하여 두 개의 새로운 질의를 질의-큐에 넣는 QT과 달리, CT에서는 리더가 태그들의 응답 메시지에서 최초 충돌 비트 이전까지의 비트들을 질의의 프리픽스에 추가한 다음, '0'과 '1'을 붙인 두 개의 새로운 질의를 질의-큐에 넣는다.

CT도 그림 5처럼 태그 식별 과정을 트리 형태로 나타낼 수 있다. 그림 5의 태그들은 그림 3의 QT의 과정과 똑같은 태그들로 이루어져 있다. 그림 5에서의 질의-응답 횟수는 9번이다.

2.5. *M*-ary 쿼리 트리 (*M*-ary QT: MQT)

MQT[10]도 역시 QT를 기반으로 한다. QT는 하나의 질의-응답 과정에 하나의 비트만 식별하는 1-

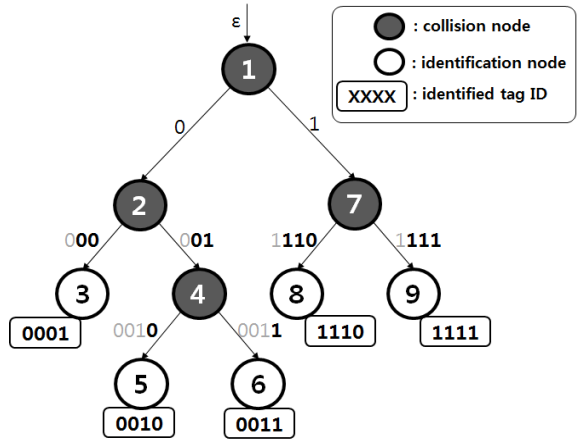

그림 5. CT의 태그 식별 과정
Fig. 5. Tag Identification in CT

비트인식(1-bit arbitration) 방법을 사용한다. 반면,

Algoritm 1. Mapping Function
Require: m-bit string $B=b_{m-1}\|\|b_{m-2}\|\|\cdots\|\|b_0$
(b_i: a binary string)
Ensure: M-bit string $P=p_{M-1}\|\|p_{M-2}\|\|\cdots\|\|p_0$
(M: 2^M, p_i: a binary value)
1. Initialize an integer $I = 0$ and p such that $p_i = 0$ for all
2: **for** $j = 0$ to $m-1$ do
3: $I \leftarrow I + b_j * 2^j$
4: **end** for
5: $p_I \leftarrow 1$

그림 6. MQT 의 매핑함수
Fig. 6. Mapping Function of MQT

MQT에는 1-비트인식 대신에 맨체스터 코딩과 매핑 함수(mapping function)를 이용한 *M*-ary 트리 검색 기법으로 *m*-비트인식(*m*-bit arbitration) 방법을 제공한다(단, $m=\log_2 M$). 그림 6은 *m*비트에 대한 매핑함수 알고리즘을 나타낸다. 매핑함수에 의해서 *m*비트 스트링은 *M*비트 스트링으로 변환된다. 한 번의 질의-응답 과정은 다음과 같이 진행된다. 리더가 질의-큐에서 하나의 질의를 꺼내어 태그들한테 브로드캐스팅방식으로 전송한다. CT와 마찬가지로 태그들이 응답할 때 리더가 전송한 프리픽스 부분을 제외한 나머지 태그 ID로만 응답한다. 이 때, 태그가 응답하는 태그 ID 비트들 중에서 최상위 비트부터 *m*비트를 매핑함수을 이용하여 *M*비트의 매핑비트로 변환하여 태그 ID를 전송한다. 태그들의 응답을 받은 리더는 매핑함수를 이용하여 응답 메시지의 비트들 중에서 최상위로부터 *M*비트의 매핑비트를 역변환하여 태그들의 ID 정보들을 추출하여 실제로 존재하는 태그들에 대한 질의들만을 생성하여 질의-큐에 추가한 다음 질의-응답 과정을 반복한다. MQT는 유휴노드의 개수를 없애며, *m*-비트인식 방식을 이용하여 충돌 노드의 개수를 상당히 줄일 수 있다.

그림 7은 그림 3와 그림 5과 같은 태그들을 MQT를 이용하여 인식하는 과정을 나타낸다. 여기서 점선으로 표시된 화살표는 리더가 전송하지 않는 질의를 하지 않는 것을 나타낸다. 그리고 그림 7에서의 질의-응답 횟수는 8번이다. 그림 7에서 노드 2는 리더가 00의 질의를 전송하면 세 개의 태그 아이디(0001, 0010, 0011)가 프리픽스(00)를 제외한 나머지 2 비트를 매핑함수를 이용하여 각각 0010, 0100, 1000 로 변환하여 응답하는 과정이다. 리더가 ***0를 수신하여 세 개의 질의(0001, 0010, 0011)를 질의-큐에 추가한 후, 세 번의 질의-응답(노드 3, 노드 4, 노드 5)을 거쳐 세 개의 태그 ID를 인식하게 된다. 이 때, 리더는 수신된 응답 메시지(***0)를 이용하여 세 개의 태그 ID를 예측할 수 있기 때문에 확인 노드를 제거하여 전체 식별 시간을 상당히 줄일 수 있다.

SA는 확률이라는 본질적인 특성 때문에 태그 식별을 보장하지 못하는 반면, QT, CT, MQT 방식은 트리 기반 프로토콜을 사용하여 태그 식별을 보장한다. QT는 태그 식별 과정이 태그 ID의 최상위 비트부터 한 비트씩 인식되고, 이 과정은 이 진 트리로 표현된다. 이 때, 태그 식별하는데 걸리는 시간은 그림 3, 그림 5, 그림 7에서 알 수 있듯이 트리의 노드 개수이다. 트리의 노드 개수를 줄이기 위해서, CT는 이진 트리 방식을 유지하면서 최초 충돌 정보를 이용하였고, MQT 방식은 이진 트리 방식을 매핑함수를 이용하여 *M*-ary 트리 방식으로 변경하였다.

CT, MQT 이외에도 맨체스터 코딩 기법에 의한 충돌 비트 정보를 이용한 기법 I4QTA[13]와 BQT[14]가 있다. I4QTA[13]는 최초 충돌 비트뿐 아니라 충돌 비트의 연속성에 대한 정보를 이용하여 빠르게 태그를 인식한다. 두 개의 연속적인 충돌 비트가 발생하면 관련된 태그는 MQT에서 *m*이 2일 때의 매핑 함수와 유사한 '비트 변환 모드'를 이용하여 응답 메시지의 비트들을 변환한다. I4QTA는 2-비트인식만 제공하는 반면, MQT는 *m*-비트인식을 제공하고 최적의 *m*값도 제공한다. BQT는 충돌 비트가 0개, 1개, 2개 이상인 경우에 따라서 0일 때는 하나의 태그를 인식하고, 1개일 때는 2개의 태그를 인식하고, 2개 이상인 경우에는 최초 충돌이 발생한 비트 정보를 가지고 구체적인 질의를 예측하여 식별 시간을 줄인다. BQT는 여러 개의 충돌 비트가 발생할 경우에 하나의 비트만 추정하는 반면 MQT는 *m*개의 비트 정보를 $M(=2^m)$로 분산시켜 분석함으로써 *m* 비트의 태그 ID를 추정할 수 있다.

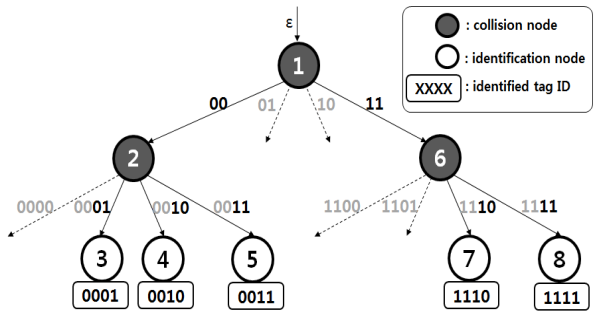

그림 7. MQT의 태그 식별 과정

Ⅲ. *M*-ary 쿼리 트리의 개선 방안 (Enhanced *M*-ary QT : EMQT)

본 논문에서는 MQT의 *m*-비트인식 방식에 예측 기법을 적용하여 태그 식별 시간을 줄이는 *M*-ary 쿼리 트리 프로토콜의 개선 방안(Enhanced *M*-ary Query Tree Protocol: EMQT)을 제안한다. SA뿐만 아니라, QT, CT, MQT 방식에서 태그를 인식하는 판단의 기준은 요구 메시지 또는 질의에 대해서 오직 하나의 태그가 응답했을 때이다. EMQT는 하나의 태그가 응답했을 때가 아니라 여러 태그들로부터 수신한 응답 메시지를 분석하여 태그들의 ID를 예측하여 식별한다. 예측을 통한 태그 ID를 식별하는 방법은 MQT에서 사용되는 매핑함수와 맨체스터 코딩기법에 의해서 가능하다. 기법은 상당히 단순하지만 EMQT 기법을 통하여 트리의 충돌노드와 유휴노드의 개수를 감소시킬 뿐만 아니라 확인 노드의 개수를 감소시켜 식별시간을 현저히 단축시키는 효과를 얻을 수 있다. 이는 트리의 관점에서는 트리의 깊이를 하나만큼 줄이는 것과 같다. 트리의 깊이(depth)를 하나를 감소하면 *M*-ary 트리일 경우에는 최대 1/*M*만큼의 트리 노드 개수를 줄일 수 있다. 특히, 물류 유통 과정에 있는 같은 회사 유사한 분류의 제품들이 함께 모여 있는 제품 창고나 쇼핑센터에서처럼 태그가 부착된 제품들의 태그 ID들이 순서대로 하나씩 증가하는 패턴일 경우에 그 감소 효과가 극대화된다.

표 1과 표 2는 각각 EMQT에서 이루어지는 리더와 태그의 알고리즘을 나타낸다. 알고리즘에서 사용되는 매핑함수는 그림 6의 알고리즘에 의해서 결정되고, 이 매핑함수에 의해서 매핑테이블이 만들어지는데 표 3-(a)와 (b)는 각각 *m*이 2일 때와 *m*이 3일 때의 매핑테이블의 예를 보여 주고 있다.

표 1과 표 2에 의해서 태그 식별되고 이 과정은 여러 개의 질의-응답으로 이루어진다. 하나의 질의-응답은 다음과 같이 수행된다. 최초에 리더는 질의 *q*에서 질의를 꺼내어 전송한다. 만약, 질의가 ε라면 모든 태그들이 응답하고, 이외에는 자신의 태그 ID의 프리픽스와 일치할 때만 응답한다. 이 때, 태그들은 ε의 경우에는 최상위 비트에서 *m*비트만큼을 매핑함수(또는 매핑테이블)를 이용하여 *M*비트로 변환하고, 그 외의 질의에 대해서는 프리픽스를 제외한 비트들에서 최상위 비트부터 m비트를 매핑함수를 이용하여 $M(=2^m)$비트로 변환하여 나머지 태그

표 1. EMQT에서 리더의 알고리즘
Table 1. Reader Algorithm in EMQT

<Reader>
1) The reader pops a query *q* from query-queue and sends out it including a bit string called prefix. The query of the longest size is first selected. If queries with the same size exist, the query with the smallest value is chosen. Initially, it starts with an empty string ε.
2) Three possible cases can arise based on the tags' response. Mapping bits are the $M(=2^m)$ bits from the significant bit. **2-1)** Only single '1' is in the mapping bits and no '*' is in the entire tags' response string: One unique tag is identified. **2-2)** The size of tag's response string is *M* and the number of '*' in the mapping bits is *t*: *t* tags are identified. **2-3)** Otherwise: Let the number of '*' in the mapping bits is *t*, then $r_1,\cdots, r_t$ are the strings that inversely mapped from the mapping function or mapping table. Queries $q\|r_1, \cdots, q\|r_t$ are prepared in the next step.
3) Repeat steps 1) and 2) until the query-queue is empty.

ID와 함께 전송한다. 만약 리더가 충돌 비트가 없는 응답을 수신할 경우에는 하나의 태그 ID 인식을 성공한 것이고, 충돌 비트가 존재하고 응답 메시지의 길이가 *M*비트일 경우에는 응답 메시지 내의 충돌 비트의 개수(*t*)의 태그 ID를 인식한 것이다. 이

표 2. EMQT에서 태그의 알고리즘
Table. 2. Tag Algorithm in EMQT

<Tag>
1) When the tag receives a query different from its own ID, it does not respond.
2) Otherwise, it responds with the rest part of except the prefix part, At this time, *m* bits from the MSB of the rest part are mapped into $M(=2^m)$ bits by the mapping function or mapping table for *m* bits.

표 3. 매핑테이블 예
Table 3. Example for Mapping Table

(*a*) 2(=*m*) 비트에 대한 매핑 함수
(*a*) Mapping function for 2(=*m*) bit

2 bit	4(=2^2) bit
00	000**1**
01	00**1**0
10	0**1**00
11	**1**000

(*b*) 3(=*m*) 비트에 대한 매핑 함수
(b) Mapping function for 3(=*m*) bit

3 bit	8(=2^3) bit
000	0000 000**1**
001	0000 00**1**0
010	0000 0**1**00
011	0000 **1**000
100	0000 0000
101	0000 0000
110	0000 0000
111	0000 0000

외의 경우, 즉 충돌 비트가 존재하고 응답 메시지가 *M*비트보다 클 경우에는 *M*비트에서 충돌 비트의 위치에 해당하는 질의를 매핑함수 또는 매핑 테이블을 이용하여 역변환하여 생성하고, 이번 질의-응답 과정의 질의에 덧붙여 질의 *q*에 추가한다. 그리고 다시 리더가 질의 *q*에서 질의를 꺼내어 전송하면서 질의-응답 과정을 수행하고, 이 과정은 질의 *q*에 더 이상 질의가 없을 때까지 계속된다.

표 4 와 그림 8은 EMQT의 태그 식별 과정을 보여 준다. 태그들은 그림 3, 그림 5 그리고 그림 7 과 동일한 구성으로 이루어져 있다.이 때, 사용되는 매핑함수는 *m*이 2일 때이며, 사용되는 매핑테이블은 표 3-(*a*)에 예시되어 있다. 다음에 식별과정을 자세히 서술한다.

태그들은 Tag1(0001), Tag2(0010), Tag3(0011), Tag4(1110), Tag5(1111) 의 다섯 개로 구성되어 있다. 처음에는 질의-큐에는 비어 있는 질의인 ε 질의로 초기화되어 있다. 첫 번째 질의-응답 과정(<1>)에서 리더는 질의-큐에서 ε 질의를 꺼내어 브로드캐스팅 방식으로 태그들에게 전송한다. ε 를 수신한 다섯개 모든 태그들은 최상위 비트로부터 2비트를 매핑함수를 이용하여 Tag1은 00을 0001로, Tag2는 00을 0001로, Tag3은 00을 0001로, Tag4는 11을 1000, Tag5는 11을 1000으로 4(=2^2) 의 매핑비트로 각각 변환한다. 그리고 Tag1은 0001 01을, Tag2는 0001 10을, Tag3는 0001 11을, Tag4는 1000 10을, Tag5는 1000 11을 각각 전송한다. 리더는 다섯 개 모두의 태그들의 응답을 수신하면 *00* **을 응답 메시지로 받는다. 여기서 매핑비트를 매핑함수를 이용하여 역변환하면 00과 11의 두 개의 태그 ID 정보를 추출하고 이 두 개를 질의-큐에 추가한다. 두 번째 질의-응답 과정(<2>)에서는 질의-큐에서 00이라는 질의를 꺼내어 브로드캐스팅 방식으로 태그들에게 전송한다. 태그 ID의 프리픽스가 00인 Tag1, Tag2, Tag3이 자신의 태그 ID에서 00을 제외하고 최상위 비트로부터 2개의 비트를 매핑함수를 이용하여 Tag1은 01을 0010로, Tag2는 10을 0100으로, Tag3은 11을 1000으로 4(=2^2) 의 매핑비트로 각각 변환한다. 그리고 Tag1은 0010을, Tag2는 0100을, Tag3는 1000을 각각 전송한다. 리더는 세 개의 태그들의 응답을 수신하면 ***0의 응답 메시지를 받는다. 리더가 수신한 응답메시지의 길이가 4(=*M*=2^2)이기 때문에 표 1의 리더의 알고리즘 2-2)에 의해서 '*'의 개수인 3개의 태그를 식별할 수 있다. 응답한 태그들은 질의 00를 프리픽스로 가지고 있고, 응답 메시지로부터 01, 10, 11이라는 태그 ID 정보를 추출할 수 있기 때문에 0001, 0010, 0011이라는 세 개의 태그 ID를 식별할 수 있다. 세 번째 질의-응답 과정(<3>)은 질의-응답 과정(<2>)와 매우 유사하다. 질의-큐에서 11이라는 질

표 4. 5개의 태그에 대한 EMQT 동작 과정
Table 4. EMQT Tag Identification for 5 tags

query-response (cycle)	query	response: mapping part	response: remaining part	queue	identification
				{ε}	
<1>	ε	*00***		{00, 11}	
<2>	00	***0		{11}	0001 0010 0011
<3>	11	**00		{ }	1110 1111

표 5. QT/CT/MQT/EMQT의 질의-응답 횟수의 수학적 분석
Table 5. Analysis for query-response cycles of QT/CT/MQT/EMQT

	query-response (# of tags: *n*, length of tag ID: *b*, *m*-bit identification) $I_M(n) = \begin{cases} 1, & if\ n \bmod M = 1 \\ 0, & otherwise \end{cases}$
QT	$N_{QT}(n) = \sum_{k=0}^{b} \left\lceil \frac{n}{2^k} \right\rceil - I_2(n)e_2(n-1)$
CT [9]	$N_{CT}(n) = 2n - 1$
MQT	$N_{MQT}(n) = \sum_{k=0}^{b/m} \left\lceil \frac{n}{2^{mk}} \right\rceil - I_{2^m}(n)e_{2^m}(n-1)$
EMQT	$N_{EMQT}(n) = \sum_{k=1}^{b/m} \left\lceil \frac{n}{2^{mk}} \right\rceil - I_{2^m}(n)e_{2^m}(n-1)$

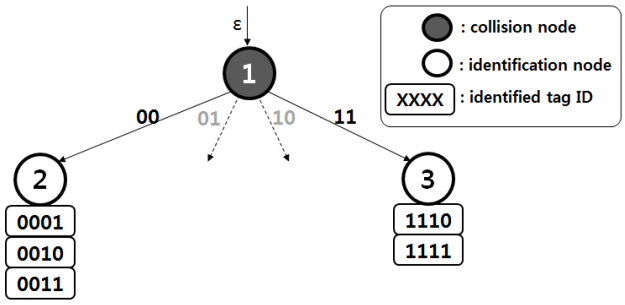

그림 8. EMQT 의 태그 식별 과정
Table 8. Tag Identification in EMQT

의를 꺼내어 브로드캐스팅 방식으로 태그들에게 전송한다. 태그 ID의 프리픽스가 11인 Tag4와 Tag5가 자신의 태그 ID에서 11을 제외하고 최상위 비트로부터 2개의 비트를 매핑함수를 이용하여 Tag4은 10을 0100로, Tag5는 11을 1000으로 4(=2^2) 의 매핑비트로 각각 변환한다. 그리고 Tag4는 0100을, Tag5는 1000을 각각 전송한다. 리더는 세 개의 태 $e_M(n)$ is the exponent of a factor M when factoring n.그 들 의 응답을 수신하면 **00의 응답 메시지를 최종적으로 수신한다. 리더가 수신한 응답메시지의 길이가 4(=M=2^2)이기 때문에 표 1의 리더의 알고리즘 2-2)에 의해서 '*'의 개수인 2개의 태그를 식별할 수 있다. 응답한 태그들은 질의 11를 프리픽스로 가지고 있고, 응답 메시지로부터 10, 11이라는 태그 ID 정보를 추출할 수 있기 때문에 1110, 1111이라는 두 개의 태그 ID를 식별할 수 있다.

그림 8은 표 4의 EMQT 태그 식별 과정을 트리 형태로 보여 주고 있다. 그림 8은 그림 3의 QT, 그림 5의 CT, 그림 7의 MQT에서와 같은 구성의 태그들로 이루어져 있기 때문에 QT, CT, MQT 그리고 제안된 방안인 EMQT와 직관적으로 태그 식별 시간을 비교할 수 있다. 질의-응답 횟수를 보면 각각 QT는 15번, CT는 9번, MQT는 8번, 그리고 EMQT는 3번이다.

Ⅳ. 성능 평가

본 논문에서는 제안하는 EMQT의 성능을 평가하기 위해서 수학적 분석과 시뮬레이션을 통한 실험적 분석을 수행하였다. 수학적 분석은 태그들의 수에 따라 성능을 수학적인 수식으로 유도하였고, 실험적 분석은 이벤트 기반 방식으로 RFID 시뮬레이터를 작성하여, 트리 기반 프로토콜들(QT, CT, MQT)과 비교하였다.

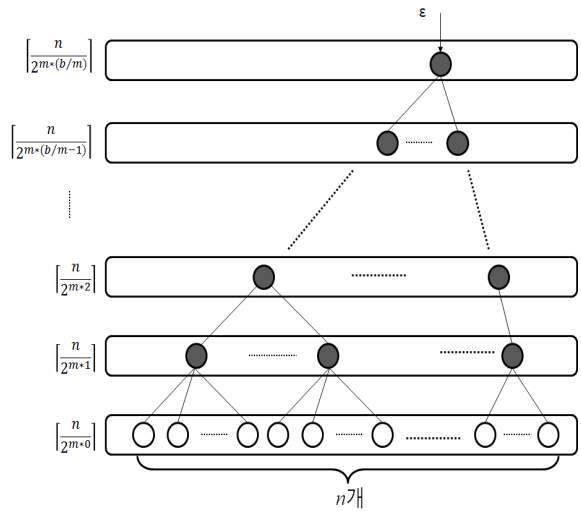

그림 9. MQT의 식별 시간 분석
Fig. 9. Analysis on Identification Time of MQT

성능 평가를 공정하게 수행하기 위해서 QT의 알고리즘을 일부 변경하였다. QT에서는 리더가 전송한 질의와 일치하는 프리픽스를 가진 태그들이 자신의 태그 ID 로 응답하였다. 그런데, 공정한 평가를 위해서 응답 시, 자신의 태그 ID 전부가 아니라 프리픽스를 제외한 태그 ID 일부로 응답하는 방식으로 알고리즘은 일부 수정한다.

또한 본 논문의 성능 평가에서는 QT는 다른 세 개의 기법보다 성능이 크게 떨어지기 때문에 QT의 측정치는 일부 그래프에서는 제외되고 표와 본문에 수치를 언급한다. 본 논문에서 수행된 시뮬레이션은 무선 환경, 하드웨어 및 소프트웨어적인 요소 등을 고려하지 않고 알고리즘의 동작 및 수학적인 분석의 실효성을 확인하는 절차이다. 때문에 수학적 분석과 실험적 분석이 일치하여 그래프에는 하나만 표시한다. 본 논문은 트리 기반의 충돌 방지 기법으로 이러한 트리기반의 충돌방지 기법을 비교는 태그 수와 크기에 따라 트리의 노드의 개수(식별 시간)를 얼마나 줄이느냐가 가장 중요한 성능평가 요소이다. 그래서 수학적인 분석을 통해서 정량적으로 비교할 수 있는 가에 비중을 두어, 다른 트리 기반 프로토콜과 정량적으로 성능 평가를 수행하였다.

4.1. 성능 평가 요소

성능 평가는 다음의 세 개의 평가 요소를 기준으로 이루어진다. 첫 번째로 식별시간(identification time)은 충돌 방지 기법에서 가장 중요한 태그가 식별되는데 걸리는 시간으로 본 논문에서는 '하나의 태그를 식별하는데 평균적으로 소요되는 질의-응답

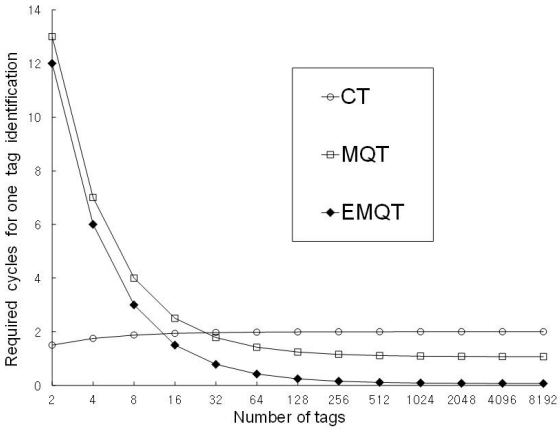

그림 10. CT, MQT, EMQT의 식별시간
Fig. 10. Identification Time of CT, MQT and EMQT

횟수'로 식별시간을 정의한다. 두 번째로 식별효율(identification efficiency)은 한 번의 질의-응답 과정으로 평균 몇 개의 태그가 식별되는가에 대한 성능을 나타낸 것으로, '리더의 전송범위 내에 존재하는 태그의 개수를 질의-응답 횟수로 나눈 수치'로 정의한다. 끝으로 통신 비용(communication overhead)은 '하나의 태그들을 식별하는데 전송되는 평균 비트수'로 나타낸다. 이 평가 요소는 에너지 소모와 관련이 있다.

4.2. 비교 분석

성능 평가 환경은 하나의 리더가 전송범위 내에 존재하는 하나 이상의 태그들을 식별하는 것을 가정한다. 태그들의 개수는 2개에서 8192개까지 변한다. 태그 ID는 현재 가장 많이 적용되어 있는 96비트를 가정한다. 물류 또는 유통 시스템 과정에 사용되는 태그들을 고려하여 리더의 전송범위 내에 있는 태그들의 ID의 번호는 순서대로 하나씩 증가한다. 태그 ID의 추가적인 필드들은 ISO-14233[1][15]을 따른다. 96비트의 태그 ID에 명령어 필드(command field)는 5비트이고, 유효 필드(valid field)는 8비트가 별도로 추가된다. MQT와 EMQT에서 사용되는 m은 [10]로부터 태그 ID의 길이가 96비트일 때 최적의 값인 4로 설정하고, 16(=24)-ary 검색 트리를 이용하여 태그 식별 과정을 수행한다.

QT, CT, MQT, EMQT의 질의-응답 횟수를 수학적으로 분석하여 비교하였다. 태그의 수는 n, 태그 ID의 비트 길이는 b이고, MQT와 EMQT의 경우 m-비트인식 방식한다. 연속적인 태그 ID들에 대해 분석하기 위해서 태그 ID는 00…00부터 하나씩 증가하여 n개 태그로 구성된 환경을 가정한다.

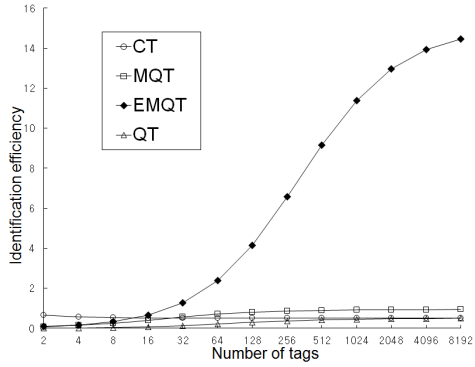

그림 11. QT, CT, MQT, EMQT의 식별효율
Fig. 11. Identification Efficiency of QT, CT, MQT and EMQT

CT의 식별시간은 태그의 수가 n이면 식별시간은 $2n$-1이다[9]. MQT의 식별시간은 트리의 충돌노드 및 확인 노드 개수들의 합이다. 태그 ID가 0…00부터 하나씩 증가하기 때문에 식별 트리의 그림 9처럼 잎(leaf) 노드를 맨 왼쪽부터 n개를 가지고, 이들이 모두 연결되어 있는 트리 모양을 갖는다. 이 때, 각 레벨의 노드의 개수는 그림 9와 같고 이를 합하면 식별시간이 된다. 단, 노드의 개수가 $M(=2^m)$배수일 때는, n의 인수 M의 지수만큼 노드의 개수를 감해야 한다. QT는 MQT에서 m=1인 경우이고, EMQT는 확인 노드의 개수를 제외하면 식별시간을 구할 수 있다.

4.2.1. 식별시간(identification time)

그림 10은 CT, MQT, EMQT의 식별시간을 보여 준다. 이 그래프는 표 5의 질의-응답 횟수를 태그의 개수로 나누어 얻을 수 있다. CT는 태그의 개수가 증가하더라도 태그 하나를 식별하는데 평균 2번 정도의 질의-응답 과정으로 일정하다. 반면, MQT와 EMQT, 그리고 그래프에 포함되어 있지 않는 QT는 태그의 개수가 증가하면 질의-응답 횟수가 감소한다. 평균적으로 하나의 태그를 식별하는데 필요한 질의-응답 과정을 살펴보면, QT는 평균 6.16번, CT는 평균 1.98번, MQT는 평균 1.59번, EMQT는 평균 0.59번의 질의-응답 과정을 수행해야 한다. QT, CT, MQT 충돌 방지 기법은 리더의 질의에 오직 하나의 태그만 응답할 경우에만 태그 ID를 식별하기 때문에 하나의 태그를 식별하기 위해 요구되는 질의-응답 횟수는 반드시 1이상이어야 한다. 그러나 EMQT는 맨체스터 코딩을 적용한 매핑 함수 기반의 *M*-ary 검색 트리에 예측 방식을 적용하기 때문에 하나의 태그를 식별하기 위한 질의-응

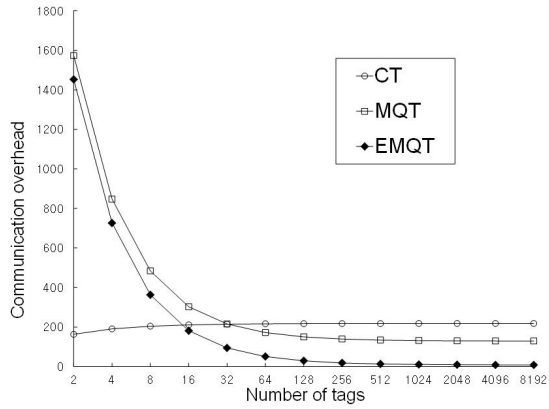

그림 12. CT, MQT, EMQT의 통신비용
Fig. 12. Communication Overhead of CT, MQT and EMQT

답 횟수가 1번보다 상당히 작은 0.59번 만에 태그 ID를 식별할 수 있다.

4.2.2. 식별효율(identification efficiency)

그림 11는 QT, CT, MQT, EMQT의 식별효율을 보여 준다. 식별효율은 태그의 개수를 표 5의 질의-응답 횟수로 나누어 얻는다. 식별효율은 리더의 질의에 오직 하나의 태그만 응답할 경우에만 태그 ID를 인식한다면, 절대로 1을 넘을 수 없다. 반면, 예측 방법을 적용한 EMQT는 1을 넘어 하나의 질의-응답 과정을 통해서 평균 7.02 개의 태그 ID를 인식할 수 있다. CT는 평균 0.51개를, QT는 평균 0.32개를, MQT는 평균 0.75개의 태그 ID를 하나의 질의-응답 과정을 통해 식별한다.

4.2.3. 통신비용(communication overhead)

그림 12는 CT, MQT, EMQT의 통신비용을 보여 준다. 통신비용은 리더와 태그들 사이에 교환되는 총 비트수를 태그의 개수로 나누어 얻을 수 있다. CT가 일정한 성능을 보여 태그의 개수가 작을 때는 좋은 성능을 보인다. 반면, QT, MQT, EMQT는 태그 수가 증가할수록 점차 통신비용이 줄어드는 것을 볼 수 있다. QT는 평균 671비트를, CT는 평균 216비트를, MQT는 평균 193비트를, EMQT는 평균 72비트를 전송한다.

V. 결 론

본 논문에서는 *m*-비트인식 및 예측 기반의 알고리즘을 이용하여 연속적인 태그 ID들을 식별할 때, 불필요한 질의-응답 횟수를 효과적으로 줄일 수 있는 EMQT 기법을 제안하였다. 그리고 이론적 추정을 통한 수학적 분석과 시뮬레이션을 통한 실험적 분석을 통해 여러 트리 기반 프로토콜들과 비교 및 분석을 통하여 EMQT가 다른 기법들보다 식별시간, 식별효율, 통신비용 관점에서 훨씬 성능이 우수하다는 것을 보여준다. 기존의 트리 기반 프로토콜에서는 태그 식별 과정에서 형성되는 트리에서 충돌노드나 유휴노드의 개수를 줄이는 것에 초점을 맞추어 연구를 진행해 왔다. 반면, EMQT는 *M*-ary 트리 검색 방식에 예측 방법을 적용하여 충돌노드나 유휴노드의 수 뿐만 아니라 확인 노드의 수를 감소시켜 현저히 성능을 개선하였다는 데에 그 의의가 있다.

References

[1] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, John Wiley & Sons, pp. 206-219, 2003.

[2] H. Vogt, "Efficient object identification with passive RFID tags," in *Proc. Int. Conf. Pervasive Comput.*, pp. 98-113, Zurich, Switzerland, Aug. 2002.

[3] M. A. Bonuccelli, F. Lonetti, and F. Martelli, "Instant collision resolution for tag identification in RFID networks," *Ad Hoc Networks*, vol. 5, no. 8, pp. 1220-1232, Nov. 2007.

[4] G. Bagnato, G. Maselli, C. Etrioli, and C. Vicari, "Performance analysis of anti-collision protocols for RFID systems," in *Proc. IEEE 69th Veh. Technol. Conf.*, pp.1-5, Barcelona, Spain, Apr. 2009.

[5] C. Law, K. Lee, and K. Y. Siu, "Efficient memoryless protocol for tag identification," in *Proc. ACM DIAL-M '00*, pp. 75-84, Boston, U.S.A., Aug. 2000.

[6] J. Myung, W. Lee, J. Srivastava, and T. K. Shih, "Tag-splitting: adaptive collision arbitration protocols for RFID tag identification," *IEEE Trans. Parallel Distributed Syst.*, vol. 18, no. 6, pp. 763-775, June 2007.

[7] J. H. Choi, D. Lee, and H. Lee, "Query

tree-based reservation for efficient RFID tag anti-collision," *IEEE Commun. Lett.,* vol. 11, no. 1, pp. 85-87, Jan. 2007.

[8] P. Hawkes, "Anti-collision and transponder selection methods for grouped vicinity cards and RFID tags," *IEE Colloq RFID Technol.,* pp. 7/1-7/12, 1999.

[9] X. Jia, Q. Feng, and C. Ma, "An efficient anti-collision protocol for RFID tag identification," *IEEE Commun. Lett.*, vol. 14, no. 11, pp. 1014-1016, Nov. 2010.

[10] J. Shin and D. Yang, "Multiple RFID tags identification with M-ary query tree scheme," *IEEE Commun. Lett.*, vol. 17, no. 3, pp. 604-607, Mar. 2013.

[11] EPC Tag Data Standard 1.6, [Online], Available: http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf, Sep. 2011.

[12] J.-R. Cha and J.-H. Kim, "Dynamic framed slotted ALOHA algorithms using fast tag estimation method for RFID system," in *Proc. IEEE Consumer Commun. Networking Conf. 2006*, vol. 2, pp. 768-772, Las Vegas, U.S.A., Jan. 2006.

[13] Y. Kim, S. Kim, S. Lee, and K Ahn, "Improved 4-ary query tree algorithm for anti-collision in RFID system," in *Proc. IEEE AINA 2009*, Bradford, U.K., pp. 699-704, May 2009.

[14] H. Gou and Y. Yoo, "Bit collision detection based query tree protocol for anti-collision in an RFID system," *Int. J. Innovative Comput., Inform. Control*, vol. 8, no. 5, pp. 3081-3102, May 2012.

[15] EPC Tag Data Standards Version 1.1 Rev.1.24, [Online], Available: http://read.pudn.com/downloads46/sourcecode/others/155143/EPCTagDataSpecification11rev124.pdf, Apr. 2004.

양 동 민 (Dongmin Yang)

2000년 8월 POSTECH 컴퓨터공학과 학사
2003년 2월 POSTECH 컴퓨터공학과 석사
2011년 2월 POSTECH 컴퓨터공학과 박사
<관심분야> 무선이동통신, 인지 라디오 네트워크

신 종 민 (Jongmin Shin)

2003년 2월 홍익대학교 컴퓨터공학과 학사
2005년 2월 POSTECH 컴퓨터공학과 석사
2011년 2월 POSTECH 컴퓨터공학과 박사
<관심분야> 무선이동통신, 유비쿼터스 컴퓨팅