RFID 시스템에서 중지신호를 이용한 쿼리 트리 프로토콜

임 인 택*

* 부산외국어대학교 컴퓨터공학부

A Query Tree Protocol with Stop Signal in RFID Systems

In-Taek Lim*

* Division of Computer Engineering, Pusan University of Foreign Studies

E-mail : itlim@pufs.ac.kr

요 약

본 논문에서는 RFID 시스템에서 식별영역 내에 있는 태그들을 식별하기 위하여 무기억 특성을 갖는 QT 프로토콜을 개선한 QT_ss 프로토콜을 제안한다. QT_ss 프로토콜에서는 질의 문자열이 식별코드의 처음 비트들과 일치하는 태그는 자신의 식별코드로 응답한다. 태그가 응답하는 동안 리더가 충돌을 감지하면 리더는 중지 신호를 방송한다. 한편 태그들이 응답을 전송하는 도중에 중지 신호를 수신하면 전송을 중단한다. 성능 분석의 결과, 본 논문에서 제안한 QT_ss 프로토콜이 QT 프로토콜에 비하여 태그의 응답 비트 수가 월등히 적음을 알 수 있었다.

ABSTRACT

In this paper, a QT_ss protocol is proposed for identifying all the tags within the identification range. The proposed QT_ss protocol revises the QT protocol, which has a memoryless property. In the QT_ss protocol, the tag will send all the bits of their identification codes when the query string matches the first bits of their identification codes. While the tags are sending their identification codes, if the reader detects a collision bit, it will send a signal to the tags to stop sending. According to the simulation results, the QT_ss protocol outperforms the QT protocol in terms of the number of response bits.

I. 서 론

RFID 시스템에서 태그 충돌을 해결하기 위한 방법은 크게 태그-주도 방식과 리더-주도 방식으로 나눌 수 있다. 태그-주도 방식은 리더가 데이터 전달을 제어할 수 없기 때문에 비동기적으로 동작한다. 따라서 태그-주도 방식은 인식속도가 느린 단점이 있어서 대부분의 적용분야에서 리더-주도 방식을 사용한다. 리더-주도 방식은 모든 태그의 응답이 리더에 의해 동시에 제어되어 동기적으로 동작하기 때문에 대부분의 적용분야에서 이 방식을 사용한다[1][2][3].

리더-주도 방식의 태그 식별 프로토콜 중에서 QT(Query-Tree) 프로토콜은 Auto-ID센터에서 제안한 무기억형(Memoryless) 프로토콜로서 모든 태그들은 태그 식별과정에서 일어난 이전 질의의 이력을 기억할 필요가 없다[3]. 리더가 질의한 프리픽스가 자신의 식별코드 중에서 처음 몇 비트들과 일치하는 태그는 자신의 식별코드로 응답한다. QT 프로토콜인 경우, 태그들로부터 식별코드를 수신하는 동안에 충돌이 발생하여도 이를 태그들에게 알리는 방법이 없다. 따라서 태그는 자신이 응답한 식별코드가 충돌이 발생하였는지를 알 수 없고, 이로 인하여 태그가 전송하는 비트의 수가 많은 문제점이 있다. 따라서 본 논문에서는 이를 개선하여 리더가 식별코드를 수신하는 도중에 충돌을 감지하면 태그들에게 전송을 중지하도록 하는 신호를 보내는 QT_ss (QT with stop signal) 프로토콜을 제안한다.

본 논문의 구성은 다음과 같다. 서론에 이어 II장에서는 본 논문에서 비교 분석하고자 하는 무기억 프로토콜인 QT 프로토콜을 설명하고, III장에서는 태그들의 응답 비트 수를 줄이기 위한 QT_ss 프로토콜을 설명하고, IV장에서는 이들 프로토콜의 성능분석 결과를 기술하고, 마지막으로 결론을 맺는다.

II. QT 프로토콜

QT 프로토콜에서는 매 질의마다 몇 비트로 구성된 프리픽스를 질의 문자열로 하여 전송한다.

수신한 질의 문자열이 자신의 식별코드 중에서 처음 몇 비트들과 일치하는 태그는 자신의 전체 식별코드를 전송하고 리더로부터 다음의 질의를 기다린다. 반면 질의 문자열이 일치하지 않는 태그는 STAND-BY 상태로 천이하여 하나의 태그가 완전히 식별되어 다음 사이클이 시작될 때까지 리더의 질의에 응답하지 않는다[4].

전송한 질의 문자열에 대하여 둘 이상의 태그가 응답하면 충돌이 발생한다. 이 경우 리더는 이전의 질의 문자열에 비트 '0'과 '1'을 각각 연장하여 새로운 프리픽스를 구성하여 다음 단계를 반복한다. 반면 리더가 충돌을 감지하지 않으면 하나의 태그를 완전히 식별한 경우이다. 이 경우, 리더는 프리픽스를 갱신하고 다음 태그를 식별하기 위하여 위의 사이클을 반복한다.

QT 알고리즘의 동작은 다음과 같다. 먼저 태그의 식별코드 길이를 k 비트로 가정한다. A를 최대 길이가 k 비트인 이진 문자열의 집합이라 하고, w를 태그의 식별코드 문자열이라 하면, 집합 A와 문자열 w는 다음과 같이 각각 정의된다.

$$A = \bigcup_{i=0}^{k} \{0,1\}^i \quad (1)$$

$$w \in \{0,1\}^i \quad (2)$$

한편 리더의 상태는 A의 문자열로 구성된 일련의 문자열인 큐 Q와 A의 원소들로 구성된 문자열의 집합인 메모리 M으로 구성된다. 여기서 큐 Q는 리더가 다음에 질의할 질의 문자열 프리픽스를 저장하는 용도로 사용되고, 메모리 M은 이미 식별된 태그의 식별코드를 저장하는 용도로 사용된다. 리더가 방송하는 질의는 A에 있는 문자열 $q(q \in A)$정의하고, 태그의 응답은 태그의 식별코드 문자열인 w로 정의한다. 초기 상태에서 리더의 큐 Q와 메모리 M은 비어있다. 먼저 리더의 알고리즘은 다음과 같다.

1) 큐를 $Q =< q_1, q_2, ..., q_l >$로 둔다.
2) 질의 q_1을 큐에서 꺼내서 방송한다.
3) 큐를 $Q =< q_2, ..., q_l >$로 갱신한다.
4) 태그로부터 수신한 응답에 대하여,
   ① 응답이 w이면 태그 w를 식별한 것으로 표시하고, w를 메모리 M에 삽입한다.
   ② 충돌이면, 큐를 $Q =< q_2, ..., q_l, q_1 0, q_1 1 >$로 갱신한다.
   4) 큐 Q가 빌 때까지 위의 과정을 반복한다.

태그의 알고리즘은 다음과 같다.

1) 태그의 식별코드 w를 $w = w_1 w_2 ... w_k$라 한다.
2) 리더로부터 수신한 질의를 q라 하고, q의 길이를 $|q|$라 한다.
3) $q = \varepsilon$ 또는 $q = w_1 w_2 ... w_{|q|}$이면, 태그는 w로 응답한다. 즉 질의 q가 자신의 식별코드 일부분과 일치하면 전체 식별코드를 리더로 보낸다.

그림 1은 위의 태그 알고리즘을 기반으로 하는 QT 프로토콜에서 태그의 상태 천이도를 나타낸 것이다. STAND-BY 상태에 있는 태그가 리더로부터 질의 문자열을 수신하면 RECEIVING 상태로 천이한다. RECEIVING 상태에서 수신하는 질의 문자열이 자신의 식별코드의 처음 비트들과 일치하지 않으면 STAND-BY 상태로 천이하여 다음 사이클의 질의를 기다린다. 반면 식별코드의 처음 비트들이 질의 문자열과 일치하는 태그는 SENDING 상태로 천이하여 식별코드의 전체 비트를 리더로 전송한 후 STAND-BY 상태로 천이하여 다음 사이클의 질의를 기다린다.

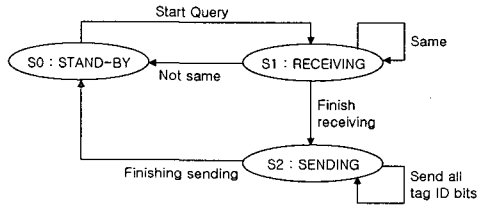

그림 1. QT 프로토콜의 태그 상태 천이도

III. QT_ss 프로토콜

QT 프로토콜에서는 태그들로부터 응답을 수신하는 도중에 충돌이 감지되어도 식별코드의 전체 비트를 모두 수신할 때까지 다음의 질의를 하지 않는다. 따라서 식별코드의 길이가 길어지고, 충돌이 빈번하게 발생하면 태그가 전송하는 데이터량이 증가하고, 이는 태그의 전력 소모량을 증가시키는 결과를 초래한다. 이러한 문제점을 해결하기 위하여 본 논문에서는 QR_ss 프로토콜을 제안한다.

본 논문에서 제안하는 QT_ss 프로토콜에서는 QT 프로토콜에서와 같이 매 질의마다 몇 비트로 구성된 질의 문자열 프리픽스를 전송한다. 수신한 질의 문자열이 자신의 식별코드 중에서 처음 비트들과 일치하는 태그는 자신의 전체 식별코드로 응답하고 리더로부터 다음의 질의를 기다린다. 반면 일치하지 않는 태그는 STAND-BY 상태로 천이하여 하나의 태그가 완전히 식별되어 다음 사이클이 시작될 때까지 리더의 질의에 응답하지 않는다.

전송한 질의 문자열에 대하여 둘 이상의 태그가 응답하면 충돌이 발생한다. 이 경우 리더는 이전의 질의 문자열에 비트 '0', '1'을 각각 연장하여 새로운 질의 문자열을 구성하여 다음 단계를 반복한다. 태그로부터 응답을 수신하는 도중에 충돌을 감지하면 리더는 중지 신호를 태그들에게 방송한다. 반면 리더가 충돌을 감지하지 않으면 하나의 태그를 완전히 식별한 경우이다. 이 경우, 리더는 질의 문자열을 갱신하고 다음 태그를 식별하기 위하여 위의 사이클을 반복한다.

QT_ss 프로토콜의 동작은 다음과 같다. 리더의 질의 문자열을 $q(q \in A)$정의하고, $|q|$를 질의 문자열의 길이로 정의한다. w를 태그의 식별코드 문자열이라 정의한다. 먼저 리더의 알고리즘은 다음과 같다.

1) 큐를 $Q=<q_1,q_2,...,q_l>$로 둔다.
2) 질의 q_1을 큐에서 꺼내서 방송한다.
3) 큐를 $Q=<q_2,...,q_l>$로 갱신한다.
4) 태그로부터 수신한 응답에 대하여,
   ① 태그로부터의 응답이 w이면, 즉 충돌이 아니면 태그 w를 식별한 것으로 표시하고 메모리에 삽입한다.
   ② 응답을 수신하는 도중에 충돌이 발생하면, 중지 신호를 방송하고, 큐를 $Q=<q_2,...,q_l,q_10,q_11>$로 갱신한다.
5) 큐 Q가 빌 때까지 위의 과정을 반복한다.

태그의 알고리즘은 다음과 같다.

1) 태그의 식별코드 w를 $w=w_1w_2...w_k$라 한다.
2) 리더로부터 수신한 질의를 q라 하고, q의 길이를 $|q|$라 한다.
3) $q=\varepsilon$ 또는 $q=w_1w_2...w_{|q|}$이면, 태그는 w로 응답한다. 즉 질의 q가 자신의 식별코드 처음 비트들 일치하면 식별코드의 전체 비트들을 리더로 보낸다.
4) 응답하는 도중에 전송 신호를 수신하면 전송을 중단한다.

그림 3은 위의 태그 알고리즘을 기반으로 하는 QT_ss 프로토콜에서 태그의 상태 천이도를 나타낸 것이다. STAND-BY 상태에 있는 태그가 리더로부터 질의 문자열을 수신하고, 수신한 질의 문자열이 자신의 식별코드 처음 비트들과 일치하는 태그는 SENDING 상태로 천이하여 자신의 전체 식별코드를 전송한다. SENDING 상태에 있는 태그가 리더로부터 중지 신호를 수신하는 경우에는 즉시 전송을 중지하고 STAND-BY 상태로 천이한다. 한편 태그가 응답하는 도중에 중지 신호를 감지하지 못하면 식별코드의 전체 비트들을 전송한 후 STAND-BY 상태로 천이한다.

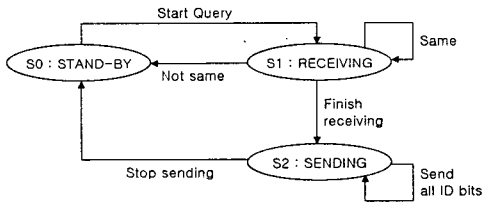

그림 3. QT_ss 프로토콜의 태그 상태 천이도

IV. 성능 분석

본 논문에서는 시뮬레이션을 통하여 QT 프로토콜과 QT_ss 프로토콜의 성능을 비교하였다. 시뮬레이션을 위하여 태그의 식별코드 길이는 64비트로 가정하였다. 태그의 식별코드가 무작위인 경우와 순차적인 경우 각각에 대하여 식별영역 내의 모든 태그를 식별하기 위하여 태그의 수에 따른 모든 태그가 보낸 비트 수를 성능평가 매개변수로 하였다. 식별코드가 순차적인 경우는 식별코드의 최하위 비트부터 순차적으로 증가하는 것으로 가정하였다.

그림 3과 4는 태그의 식별코드를 무작위로 한 경우와 순차적으로 한 경우, 모든 태그를 식별할 때까지 태그들이 응답한 총 비트의 수를 각각 나타낸 것이다. 태그의 식별코드가 무작위이든 순차적이든 상관없이 본 논문에서 제안한 QT_ss 프로토콜이 QT 프로토콜에 비하여 태그가 응답하는 비트의 수가 적다. 이는 QT 프로토콜인 경우, 리더가 태그로부터 응답을 수신하는 동안 충돌이 발생하였음을 감지하여도 이를 태그들에게 알려주지 않아서 매 질의마다 태그들은 식별코드의 전체 비트를 모두 전송하는 반면, 본 논문에서 제안한 QT_ss 프로토콜인 경우에는 리더가 충돌을 감지하는 즉시, 태그들에게 중지 신호를 전송하여 태그들로 하여금 응답 전송을 중지하도록 하게 때문이다. 태그의 식별코드가 순차적인 경우에는 QT 프로토콜은 본 논문에서 제안한 QT_ss 프로토콜에 비하여 약 1.07배 더 많은 반면, 무작위일 때는 QT 프로토콜이 QT_ss 프로토콜에 비하여 약 5.2 배 더 많다.

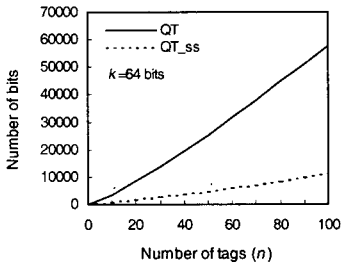

그림 3. 총 응답 비트의 수(무작위 식별코드)

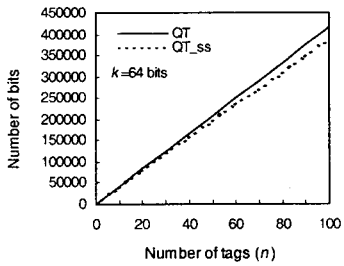

그림 4. 총 응답 비트의 수(순차적 식별코드)

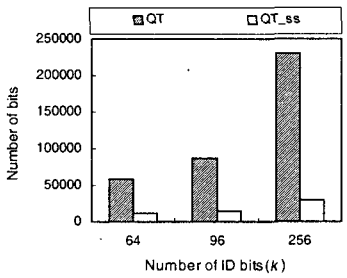

그림 5. 식별코드 길이에 따른 총 응답 비트의 수(무작위 식별코드)

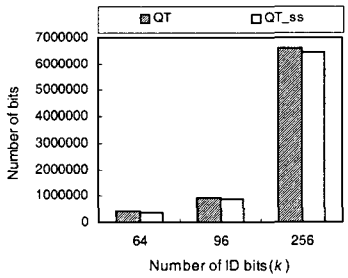

그림 6. 식별코드 길이에 따른 총 응답 비트의 수(순차적 식별코드)

Auto-ID 센터의 클래스 0 RFID 시스템에서는 세 가지 형태의 64비트와 한 가지 형태의 96비트, 및 세 가지 형태의 256비트를 갖는 EPC (Electronic Product Code)를 정의하고 있다[5]. 일반적으로 식별코드의 길이가 길면 길수록 태그들이 응답해야 하는 비트의 수는 많아질 것으로 예상된다. 따라서 본 논문에서는 식별코드의 길이가 AutoID 센터에서 정의한 64, 96, 및 256비트일 때 태그 식별 프로토콜의 성능을 분석하였다.

그림 5와 6은 식별코드의 길이에 따른 태그들의 총 응답 비트 수를 나타낸 것이다. 그림 4의 결과에서처럼 식별코드가 순차적인 경우에는 그림 6에서 나타낸 바와 같이 두 프로토콜의 총 응답 비트 수는 그다지 차이가 없다.

그림 5에서 나타낸 바와 같이 식별코드가 무작위인 경우, 길이가 64비트에서 256비트로 증가하면 QT 프로토콜의 응답 비트 수는 약 4.0배로 증가하는 반면, 본 논문에서 제안한 QT_ss 프로토콜은 약 2.7배만 증가한다. 또한 식별코드의 길이가 64비트일 경우에 태그들의 총 응답 비트 수는 QT 프로토콜이 QT_ss 프로토콜에 비하여 약 5.2배 더 많은 반면, 256비트인 경우에는 약 7.6배가 더 많다. 따라서 식별영역 내에 있는 태그들의 식별코드가 무작위인 RFID 응용에서는 본 논문에서 제안한 QT_ss 프로토콜이 QT 프로토콜에 비하여 월등히 우수한 성능을 나타낼 수 있음을 알 수 있다.

V. 결 론

본 논문에서는 리더-주도 방식의 무기억 태그 식별 프로토콜인 QT 프로토콜을 개선한 QT_ss 프로토콜을 제안하고, 이에 대한 성능을 분석하였다. 무기억 태그 식별 프로토콜인 경우, 모든 태그들은 태그 식별과정에서 일어난 이전의 질의 이력을 기억할 필요가 없는 반면, 리더는 매 질의마다 몇 비트의 질의 문자열 프리픽스를 방송한다. QT 프로토콜인 경우, 태그가 응답하는 도중에 충돌이 발생하였음에도 불구하고 태그는 이를 감지할 수 없으므로 자신의 식별코드 전체를 전송해야한다. 따라서 식별코드의 길이가 길어지면 태그가 전송하는 데이터 량이 증가하고, 이에 따라 태그의 전력 소모량을 증가하는 문제점이 있다.

반면 본 논문에서 제안한 QT_ss 프로토콜인 경우, 리더가 충돌을 감지하면 중지 신호를 태그들에게 방송하고, 이를 감지한 태그들은 즉시 전송을 중지한다. 이로 인하여 태그들이 전송하는 비트의 수가 QT 프로토콜에 비하여 월등히 적음을 시뮬레이션을 통하여 알 수 있었다.

참고문헌

[1] H. Vogt, "Efficient Object Idetification with Passive RFID Tags," *First International Conf. on Pervasive Computing, LNCS*, vol.2414, pp.99-113, Springer-Verlag, 2002.

[2] M. Jacomet, A. Ehrsam, and U. Gehrigm "Contactless Identification Device with Anticollision Algorithm," *Proc. IEEE CSCC'99*, Athenes Ltaly, July 1999.

[3] C. Law, L. Lee, and K. Y. Siu, "Efficient Memoryless Protocol for Tag Identification," Auto-ID Center, *MIT-AUTOID-TR-003*, Oct. 2000.

[4] I. Papadimitriou, and M. Paterakis, "Energy-Conserving Access Protocol for Transmitting Data in Unicast and Broadcast Mode," *Proc. IEEE PIMRC2000*, vol.2, pp.984-988, Sept. 2000.

[5] Auto-ID Center, "860MHz-930MHz Class 0 Radio Frequency Identification Tag Protocol Specification Candidate Recommendation, Version 1.0.0," June 2003.