IE Interfaces
Vol. 22, No. 3, pp. 278-286, September 2009.

# RFID 시스템에서 다중 태그 인식을 위한 하이브리드 충돌방지 알고리즘의 개선 및 성능 분석

최태정[1] · 서재준[2] · 백장현[3†]

[1]전북자동차부품산업혁신센터 / [2]한밭대학교 산업경영공학과 / [3]전북대학교 산업정보시스템공학과

# Improvement and Performance Analysis of Hybrid Anti-Collision Algorithm for Object Identification of Multi-Tags in RFID Systems

Tae-Jeong Choi[1] · Jae-Joon Seo[2] · Jang-Hyun Baek[3]

[1]Jeonbuk Auto-Parts Industry Innovation Center

[2]Dept. of Industrial and Management Engineering, Hanbat National University

[3]Dept. of Industrial and Information Systems Engineering, Chonbuk National University

The anti-collision algorithms to identify a number of tags in real-time in RFID systems are divided into the anti-collision algorithms based on the Framed slotted ALOHA that randomly select multiple slots to identify the tags, and the anti-collision algorithms based on the Tree-based algorithm that repeat the questions and answer process to identify the tags. In the hybrid algorithm which is combined the advantages of these algorithms, tags are distributed over the frames by selecting one frame among them and then identified by using the Query tree frame by frame. In this hybrid algorithm, however, the time of identifying all tags may increase if many tags are concentrated in a few frames. In this study, to improve the performance of the hybrid algorithm, we suggest an improved algorithm that the tags select a specific group of frames based on the earlier bits of the tag ID so that the tags are distribute equally over the frames. By using the simulation and mathematical analysis, we show that the suggested algorithm outperforms traditional hybrid algorithm from the viewpoint of the number of queries per frame and the time of identifying all tags.

***Keyword:*** RFID, anti-collision, framed query tree

## 1. 서 론

RFID(Radio Frequency IDentification) 시스템은 무선 주파수를 이용하여 시간과 장소의 제약 없이 실시간으로 사물의 정보를 획득, 처리, 활용하는 시스템이다(Finkenzeller, 1999). RFID 시스템은 기존의 바코드보다 다양한 정보를 저장할 수 있고 무선으로 태그에 입력된 정보를 읽을 수 있다는 장점을 가진다. RFID 시스템의 도입으로 상품 인식에 소요되는 시간과 인력을 절감하고 유통물류의 자동화, 고속화 실현을 통해 물류비용 절감을 가능하게 해준다. 또한 제조 기업에서 RFID 시스템의 도입은 기존의 공급망 관리의 효율성을 높이고, 이를 통한 관리비용 절감과 정확한 예측을 통한 생산성 및 마케팅 효율성 향상을 기대할 수 있다. RFID 태그는 기존 바코드의 수천배에 달하는 정보를 입력할 수 있기 때문에 공급망상에서 상품의 유통과정에 대한 다양한 정보를 수집할 수 있으며 이를 제품 기획, 생산관리, 판매 관리 등에 연계하여 보다 정확한 수요 예

†연락저자 : 백장현 교수, 561-756 전북 전주시 덕진동 1가 664-14 전북대학교 산업정보시스템공학과, Fax : 063-270-2333,
E-mail : jbaek@chonbuk.ac.kr
투고일(2009년 07월 17일), 심사일(1차 : 2009년 08월 20일), 게재확정일(2009년 08월 24일).

측이 가능하게 된다(Lee *et al.*, 2004).

일반적으로 RFID 시스템은 하나의 리더와 많은 태그들로 이루어진다. RFID 시스템에서 이용되는 수동형 RFID는 그 능력이 매우 제한적이어서 다른 태그들과 많은 통신을 할 수 없고 단지 리더와만 통신할 수 있다. 리더는 무선 채널을 통하여 태그들과 통신을 하는데, 모든 태그들이 리더가 보낸 신호를 동시에 듣고 리더의 전송요구에 응답하게 된다. 이러한 리더와 태그의 통신과정에서 다수의 태그들이 동시에 데이터를 전송하게 되므로 충돌이 발생하게 되며 충돌이 발생된 데이터의 인식을 위해 재전송 등의 프로세스가 요구된다. 이러한 데이터의 재전송 과정 등이 반복되면, 태그 인식시간의 증가 및 인식률 감소를 야기시켜, RFID 시스템의 효율을 떨어뜨리게 된다. 따라서 효율적인 RFID 시스템을 구성하기 위해서는 이러한 태그 충돌을 최소화하면서, 모든 태그를 가능한 빠른 시간 내에 인식해야 한다. 이를 위한 해결방법이 충돌방지 알고리즘이다(Kim and Lee, 2006).

RFID 시스템의 효율적인 운용을 위한 다양한 충돌방지 알고리즘이 제안되었다(Finkenzeller, 1999; Lee *et al.*, 2004; Kim and Lee, 2006; Shin *et al.*, 2007). 본 연구에서는 확률적 기반의 충돌방지 알고리즘과 결정적 기반의 충돌방지 알고리즘을 결합한 하이브리드 알고리즘(FQT, Framed Query Tree)(Shin *et al.*, 2007)을 대상으로 다수의 태그가 몇몇 프레임에 집중되어 해당 프레임의 태그 인식시간이 길어지는 문제를 해결하기 위하여 개선된 FQT 알고리즘을 제안하고 그 성능을 분석한다. 기존 하이브리드 알고리즘의 성능을 개선하기 위하여, 태그 ID 앞부분의 비트 값에 따라 특정 그룹의 프레임을 선택하도록 유도함으로써 모든 프레임에 태그가 골고루 분산되도록 하는 방법을 제안한다. 시뮬레이션 및 해석적 방법을 이용하여 제안하는 방법이 기존의 방법에 비하여 프레임당 충돌 횟수와 쿼리 수를 감소시켜 줌으로써 모든 태그를 인식하는 데에 소요되는 시간을 감소시키는 것을 보이고자 한다.

본 논문의 구성은 다음과 같다. 서론에 이어 제 2장에서는 충돌방지 알고리즘에 관한 기존의 연구를 소개한다. 제 3장에서는 개선된 하이브리드 알고리즘을 제안한다. 제 4장에서는 시뮬레이션을 이용하여 분석결과를 제시하고, 제 5장에서 결론을 맺는다.

## 2. 관련 연구

다수의 태그를 인식하기 위한 충돌방지 알고리즘은 크게 태그들이 여러 슬롯을 랜덤하게 선택하도록 하여 태그를 인식하는 Framed Slotted ALOHA 기반의 확률적 충돌방지 알고리즘(Schoute, 1980; Wieselthier *et al.*, 1989)과, 질의 응답과정을 반복하여 이진 비트로 표현된 태그의 고유 ID를 순차적으로 인식하는 Tree 기반의 결정적 충돌방지 알고리즘(Finkenzeller, 1999; Quan and Kim, 2003; Lim, 2006)으로 나뉜다.

확률적 충돌방지 알고리즘은 리더가 태그 수를 예측하여 프레임 사이즈(slot 수)를 결정하며 태그는 랜덤하게 이중 하나의 슬롯을 선택한다. 확률적 충돌방지 알고리즘에서는 두 개 이상의 태그가 하나의 슬롯을 선택하면 충돌이 발생한다. 따라서 한 번의 인식과정만으로는 모든 태그를 인식하기가 어려우며 충돌이 발생한 태그들은, 모든 태그를 인식할 때까지 재전송을 수행하는 과정을 반복한다. 이 알고리즘을 사용할 경우 태그 수의 예측이 잘못되면, 많은 충돌이 발생하여 성능이 나빠지고 제 때에 태그를 인식하지 못할 수 있다.

결정적 충돌방지 알고리즘은 리더가 태그의 고유 ID를 질의하면서 태그 그룹을 세분화시켜 하나의 태그가 인식될 때까지 이 과정을 반복한다. 이 방법은 모든 태그를 완벽하게 인식할 수 있지만 태그가 많아지면 충돌이 많아져서 트리가 깊어지기 때문에 결국 태그 인식 시간이 길어질 수 있다.

### 2.1 Slotted ALOHA 기반 알고리즘

Framed Slotted ALOHA(Schoute, 1980; Wieselthier *et al.*, 1989) 알고리즘은 RFID 시스템에서 태그의 충돌문제를 해결하기 위해 18000-6 A(ISO/IEC, 2004a), EPCglobal Gen2(EPCglobal, 2005), 18000-7(ISO/IEC, 2004b), EPC CI(Auto-ID Center, 2003)에서 실제로 사용되고 있는 대표적인 알고리즘이다.

Framed Slotted ALOHA의 태그 인식 과정은 다음과 같다. 리더는 태그에게 RF 신호를 전송하고 신호를 전송받은 태그는 리더의 명령을 수행하기 위해 Ready 상태로 전환한다. 동시에 리더는 태그에게 초기 프레임 사이즈를 전송하면 프레임 사이즈를 수신한 태그들은 Active 상태로 전환한 후 Random number generator를 이용하여 자신의 ID를 전송할 슬롯을 선택한 후 리더의 명령을 기다린다. 리더가 슬롯 number를 1씩 증가시키면서 태그에게 전송을 요구하면, 태그들은 리더가 보내온 슬롯 number와 자신의 Random number를 비교한 후 일치하면 자신의 ID를 전송한다. 이 때 하나의 태그만 존재하면 충돌 없이 전송을 할 수 있다. ID 전송이 완료되면 리더에게서 Ack를 받은 후 다음 프레임에서 리더의 명령에 반응하는 것을 막기 위해 태그의 상태를 Quiet로 전환시킨다. 만약 두 개 이상의 태그가 전송을 시도하면 충돌이 발생하여 전송을 완료하지 못한다. 이 때 태그들은 다음 리더의 명령을 기다린다. 리더는 프레임 사이즈와 슬롯 number가 같아질 때까지 이러한 과정을 반복하면서 태그를 인식하고 한 프레임이 끝나면 리더는 태그의 수를 추정하여 다음 프레임 사이즈를 결정하고 모든 태그를 인식할 때 까지 위의 과정을 반복한다(Schoute, 2002; Vogt, 2002).

<그림 1>은 5개의 태그를 대상으로 Framed Slotted ALOHA 알고리즘의 동작과정을 나타내고 있다. 리더는 태그에 ID 전송요구와 함께 프레임 사이즈 4를 보내면 태그는 자신이 선택할 슬롯을 선택하여 자신의 정보를 전송한다. 첫 번째 프레임에서는 2, 3번 슬롯을 태그 2와 태그 3이 선택하여 태그 정보를 전송하고 Quiet 상태로 전환한다. 하지만 1번 슬롯에서는 태그

1, 4, 5가 충돌을 발생하여 ID 전송이 중단되고, 태그들은 다음 프레임의 명령을 기다린다. 리더는 4번 슬롯까지 확인을 마치면 ID 전송을 하지 못한 태그들에게 프레임 사이즈를 전송하여 태그 인식 과정을 다시 시작한다. 두 번째 프레임에서는 모든 태그가 충돌 없이 자신의 정보를 리더에게 전송하였으므로 태그 인식 알고리즘이 종료된다.

| | | Slot 1 | Slot 2 | Slot 3 | Slot 4 | | Slot 1 | Slot 2 | Slot 3 | Slot 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 충돌 | Tag 2 인식 | Tag 3 인식 | 빈 슬롯 | | Tag 5 인식 | Tag 1 인식 | 빈 슬롯 | Tag 4 인식 |
| Tag 1 (1001) | | Tag 1 | | | | | | Tag 1 | | |
| Tag 2 (0101) | Frame size =4 | | Tag 2 | | | Frame size =4 | | | | |
| Tag 3 (1010) | | | | Tag 3 | | | | | | |
| Tag 4 (1101) | | Tag 4 | | | | | | | | Tag 4 |
| Tag 5 (1110) | | Tag 5 | | | | | Tag 5 | | | |

**Figure 1.** Framed Slotted ALOHA 알고리즘에서 태그 인식 과정의 예

프레임 사이즈를 결정할 때 리더는 수신한 슬롯의 상태를 빈슬롯, 식별슬롯, 충돌슬롯으로 구분하고, 슬롯의 상태를 파악한 후 태그 개수를 추정하여 다음 프레임 사이즈를 결정한다(Vogt, 2002).

### 2.2 Query Tree 알고리즘

QT 알고리즘(Law *et al.*, 2000)은 리더와 태그의 질의 응답과정을 반복하여 충돌이 발생하면 질의를 만들어가면서 태그를 인식하는 알고리즘이다. 리더는 질의가 저장된 큐에서 k-bit로 구성된 프리픽스의 질의를 태그에 전송한다. 태그는 리더로부터 전송받은 질의와 자신의 ID를 비교하여 일치하면 리더에 자신의 정보를 전송하고, 일치하지 않으면 전송하지 않는다.

리더의 질의에 대한 응답은 하나의 태그만 응답하는 경우, 응답이 없는 경우, 2개 이상의 태그가 응답하는 경우로 나눌 수 있다. 하나의 태그가 응답할 경우에는 태그 ID를 저장 후 QT 알고리즘을 종료한다. 만약에 2개 이상의 태그가 자신의 정보를 리더에게 전송하면 충돌이 발생한다. 충돌이 일어나면 리더는 이전 프리픽스에 '0'과 '1'을 추가하여 큐에 삽입한 후 새로운 프리픽스에서 질의를 가져와 태그에게 질의 하고, 태그는 리더로부터 받은 질의와 자신의 ID를 비교한 후 자신의 상태와 일치하면 ID를 전송하고 일치하지 않으면 다음 질의를 기다린다. 이러한 과정을 큐가 빌 때까지 반복하면 모든 태그 인식이 완료된다.

<그림 2>는 ID가 '0001', '0010', '1010', '1011'인 4개의 태그에 대하여 QT 알고리즘의 동작과정을 보여준다. 리더는 모든 태그에 태그 정보를 보내라는 명령을 보낸다. 이 때 모든 태그가 동시에 응답을 하기 때문에 충돌이 발생한다. 리더는 충돌을 인식하고 '0'과 '1'을 큐에 저장하여 새로운 프리픽스를 생성한 후 '0'을 꺼내어 태그에게 질의를 하면 ID가 '0'으로 시작하는 Tag1, 2가 응답하게 되어 충돌이 발생한다. 충돌이 발생하였으므로 큐에 '00'과 '01'을 추가하고 다시 '1'을 꺼내어 태그에게 질의를 한다. 큐가 빌 때까지 이러한 과정을 반복하면 리더는 모든 태그를 인식하게 된다. 결국 4개의 태그를 인식하는데 6회의 충돌과 13회의 질의응답 과정을 거치게 된다.

| 태그 | ID |
|---|---|
| 1 | 0001 |
| 2 | 0010 |
| 3 | 1010 |
| 4 | 1011 |

| step | Query Prefix | Response |
|---|---|---|
| 1 | | 충돌 |
| 2 | 0 | 충돌 |
| 3 | 1 | 충돌 |
| 4 | 00 | 충돌 |
| 5 | 01 | 응답없음 |
| 6 | 10 | 충돌 |
| 7 | 11 | 응답없음 |
| 8 | 000 | 0001 인식 |
| 9 | 001 | 0010 인식 |
| 10 | 100 | 응답없음 |
| 11 | 101 | 충돌 |
| 12 | 1010 | 1010 인식 |
| 13 | 1011 | 1011 인식 |

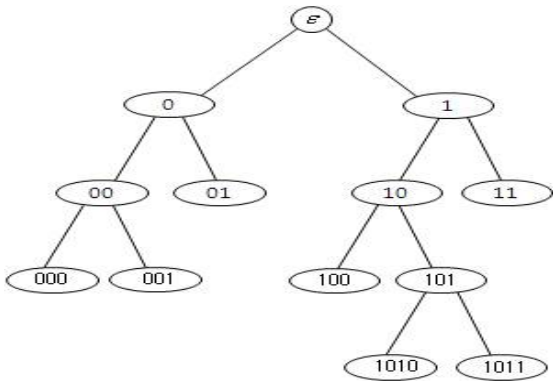


**Figure 2.** QT 알고리즘의 태그 인식과정의 예

### 2.3 하이브리드 알고리즘

Framed Query Tree(FQT) 알고리즘(Shin *et al.*, 2007)은 Frame Slotted ALOHA 알고리즘과 QT 알고리즘을 혼합한 알고리즘이다. FQT 알고리즘은 다수의 태그들이 존재할 때 트리가 깊어지는 것을 방지하기 위하여 태그들이 랜덤하게 프레임을 선택하게 하여 태그들을 분산시키고, QT 알고리즘을 사용하여 프레임을 선택한 태그들을 인식하는 방법을 사용한다.

실제 동작방식은 다음과 같다. 리더가 태그에게 ID 전송요청을 할 때 초기 프레임 사이즈를 전송하면, 태그들은 자신의 ID를 전송할 프레임을 선택한 후 리더의 명령을 기다린다. 리더는 1번 프레임에서 마지막 프레임까지 순차적으로 각 프레임에서 QT 알고리즘을 사용하여 태그를 인식한다.

<표 1>과 <그림 3>은 8개의 태그를 4개의 프레임을 사용하여 인식하는 과정을 보여준다. 각 태그들은 표와 같이 프레임 1(Tag1, 2), 프레임 2(Tag3, 4), 프레임 3(Tag5, 6), 프레임 4(Tag7, 8)를 선택한다. 자신이 참여할 프레임을 선택한 태그들은 리더에게 질의를 받고 자신의 ID를 전송한다. 이 때 QT 알고리즘이 사용된다. QT 알고리즘을 사용한 결과 전체 충돌 횟수는 6회, 쿼리 수는 16회임을 알 수 있다.

**Table 1.** FQT 알고리즘의 태그 인식 과정의 예
(태그의 프레임 선택)

| Frame size = 4 | |
|---|---|
| Frame | 태그 |
| 1 | Tag1(110) |
| | Tag2(100) |
| 2 | Tag3(001) |
| | Tag4(111) |
| 3 | Tag5(011) |
| | Tag6(010) |
| 4 | Tag7(000) |
| | Tag8(101) |

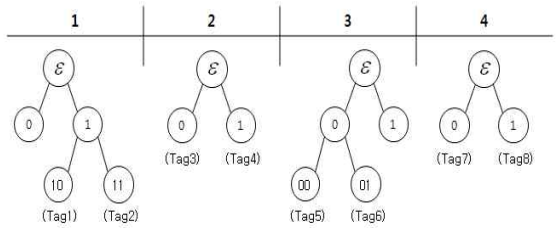


**Figure 3.** FQT 알고리즘의 태그 인식 과정의 예
(각 프레임에서의 인식과정)

FQT 알고리즘을 이용할 경우 적절한 프레임 사이즈를 결정하는 것이 중요하다. QT 알고리즘을 시행했을 때 하나의 프레임에 첫 비트가 각각 '0'과 '1'인 2개의 태그가 할당되는 경우가 가장 좋은 성능을 내기 때문에(Shin *et al.*, 2007) 하나의 프레임에 2개의 태그가 할당되도록 유도한다. 즉 최적의 성능을 이끌어낼 수 있다면 프레임 사이즈는 태그 개수의 1/2로 하는 것이 바람직하다. 그러나 실제 시스템에서는 태그의 개수를 알 수 없기 때문에 처음부터 알맞은 프레임 사이즈를 결정하기는 힘들다. 이 문제를 해결하기 위해 FQT 알고리즘에서는 First Frame Test(FFT)를 사용한다. FFT는 FQT 알고리즘을 수행할 때 첫 번째 프레임에서 충돌이 3회 이상 발생하면 한 프레임에 2개 이상의 태그들이 있다고 판단하여, 태그 인식을 중단하고 프레임 사이즈를 다시 설정하는 것이다(Shin *et al.*, 2007). 그러나 FFT는 태그의 수가 충분히 클 때나 도움이 되며 태그의 수가 적을 때에는 오버헤드로 작용하여 오히려 성능이 나빠질 수도 있다.

## 3. 제안하는 알고리즘(Improved FQT)

FQT 알고리즘에서는 여러 태그들이 랜덤하게 프레임을 선택하기 때문에 적절한 프레임 사이즈를 결정하여도 몇몇 프레임에 집중되어 해당 프레임의 태그 인식시간이 길어지는 경우가 종종 발생한다. 본 연구에서는 기존 FQT 알고리즘의 성능을 개선하기 위하여, 태그 ID 앞부분의 비트 값에 따라 특정 그룹의 프레임을 선택하도록 유도함으로써 모든 프레임에 태그가 골고루 분산되도록 하는 방법을 제안한다.

제안하는 IFQT(Improved FQT) 알고리즘은 태그들의 ID 값을 이용하여 그룹을 구분한 후 각각의 그룹들이 서로 다른 프레임을 선택하게 한다. 리더가 태그들에게 프레임을 랜덤하게 선택하도록 하는 명령을 보낼 때 모든 태그에게 같은 명령을 보내는 것이 아니라 처음 두 비트가 '00' '10'인 태그들과 '01' '11'인 태그들에게 각각 다른 프레임을 선택하도록 하고, 태그들은 자신이 참여할 프레임을 선택한 후 리더의 명령을 기다린다. 각 그룹의 태그들이 프레임을 선택하는 과정은 QT 알고리즘에서 충돌이 발생했을 때 '0'과 '1'로 구분하여 쿼리를 보내는 방법을 응용하여 태그 응답 쿼리 대신에 처음 두 비트가 '00' '10'인 태그들에게는 짝수 프레임을 선택하게 하고, '01' '11'인 태그들은 홀수 프레임을 선택하게 하는 명령을 보낸다.

리더는 태그들에게 이와 같은 명령어를 동시에 보내고, 태그들은 자신의 ID와 비교하여 선택적으로 리더의 명령을 받아 프레임을 선택하기 때문에 기존의 방법과 비교했을 때 전체 태그 응답시간에 영향을 주지 않는다. 이 때 추가 비용 없이 선택적으로 리더의 명령을 받아 수행할 수 있게 태그의 random number generator를 변경할 수 있다.

태그들이 프레임 선택을 마치면 리더는 각 프레임에서 QT 알고리즘을 사용하여 태그를 인식한다. 즉, 리더가 각 프레임에서 태그들에게 ID를 요청할 때 충돌이 발생하면 QT를 사용한다. 이 때 각 프레임에서 태그들의 앞 2 비트를 알고 있기 때문에 프레임 번호가 짝수일 때 초기 질의 문자열은

$$Q = [q_{12}00,\ q_{12}10,\ q_3, \cdots, q_l]$$

이고, 홀수일 때의 초기 문자열은

$$Q = [q_{12}01,\ q_{12}11,\ q_3, \cdots, q_l]$$

로 나타낸다. 이 후 FQT 알고리즘과 동일한 방법으로 모든 프레임에서 태그인식을 완료한다.

<그림 4>에 제안하는 충돌방지 알고리즘의 태그 인식과정을 흐름도로 나타내었다. <표 2>와 <그림 5>는 8개의 태그를 4개의 프레임을 사용하여 인식하는 과정 중 가장 좋은 경우에 대한 예이다. 각 태그들은 표와 같이 프레임 1(Tag1, 2), 프레임 2(Tag3, 4), 프레임 3(Tag5, 6), 프레임 4(Tag7, 8)를 선택한 후 QT 알고리즘을 통해 태그를 인식한다. 이 때 전체 충돌 횟수는 4회, 쿼리 수는 12회임을 알 수 있다.

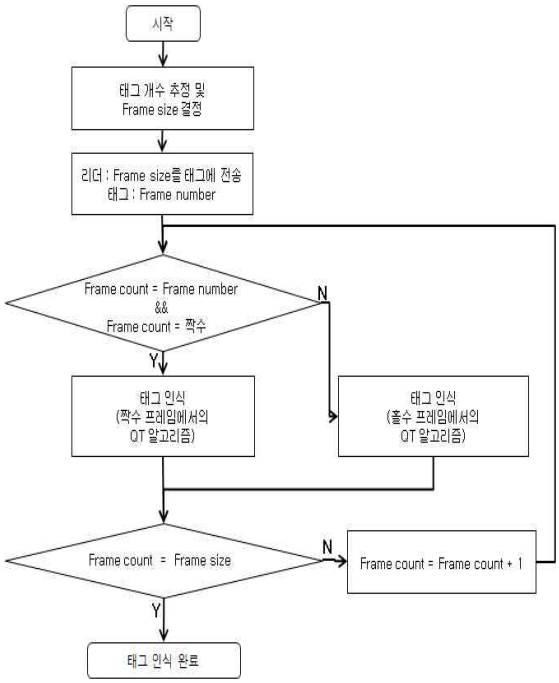


**Figure 4.** IFQT의 태그 인식과정 흐름도

**Table 2.** IFQT의 태그 인식 과정의 예(태그의 프레임선택)

| Frame size=4 | |
|---|---|
| Frame | 태그 |
| 1 | Tag1(010) |
| | Tag2(110) |
| 2 | Tag3(001) |
| | Tag4(101) |
| 3 | Tag5(011) |
| | Tag6(110) |
| 4 | Tag7(000) |
| | Tag8(101) |

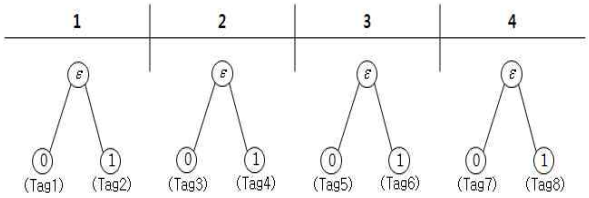


**Figure 5.** IFQT의 태그 인식 과정의 예
(각 프레임에서의 인식과정)

본 연구에서는 기존의 FFT에 해당하는 절차를 대신하여, 태그 ID들의 비트별 합인 SB(Sum of bit)(Ko *et al.*, 2007)를 이용하여 태그 개수를 추정하고 그에 대응하는 프레임 사이즈를 설정한다.

이러한 SB를 이용한 초기 프레임 사이즈 결정방법의 원리를 간단히 설명하면 다음과 같다. 일반적으로 태그 ID는 '0'과 '1'로 구성되어 있고 고유의 ID를 사용하고, 서로 다른 ID를 갖기 때문에 비트별 '0'의 개수도 '1'의 개수만큼 있다고 가정할 수 있다. 결과적으로 비트별 SB의 평균값을 통해 전체 태그 개수를 추정할 수 있다.

| | 태그 ID | | | | |
|---|---|---|---|---|---|
| Tag 1 | 0 | 1 | 0 | 1 | |
| Tag 2 | 0 | 0 | 1 | 1 | |
| Tag 3 | 1 | 0 | 1 | 0 | |
| Tag 4 | 1 | 1 | 0 | 0 | |
| | SB_1=2 | SB_2=2 | SB_3=2 | SB_4=2 | SB_ave=2 |

**Figure 6.** SB를 이용한 태그 개수 추정과정의 예

<그림 6>은 4개의 태그 ID를 가지고 태그 수를 추정하는 예이다. ID가 '0101', '0011', 1010', '1100'인 4개의 태그를 살펴보면, SB_1=2, SB_2=2, SB_3=2, SB_4=2이고 SB의 평균인 SB_ave는 2가 된다. 그 결과로 태그 수는 2×SB_ave=4로 추정한다. 추정된 태그 수를 근거로 프레임 사이즈를 2로 결정한다.

## 4. 분석 모형

본 절에서는 FQT 알고리즘과 IFQT 알고리즘의 시스템의 성능을 비교하고자 한다. 핵심 성능요소인 충돌횟수와 쿼리횟수를 파악하기 위하여 가능한 모든 경우를 고려하여 그 기댓값을 구하고 그에 따른 비용을 산출한다. 먼저 충돌과 쿼리에 의한 시스템 비용은 다음과 같다.

$$CS = TC + TQ \tag{1}$$

여기서 $CS$(Cost of System)는 시스템 비용이고, $TC$(Total Collision), $TQ$(Total Query)는 각각 충돌과 쿼리에 의한 비용을 나타낸다.

하나의 프레임 안에 존재할 수 있는 태그의 수는 0~$N$이다. 즉 프레임 안에 하나의 태그도 없는 경우에서부터 모든 태그가 하나의 프레임 안에 있는 경우까지가 가능하다. 따라서 가능한 경우의 수 $NTF$(Number of Tags in a Frame)는 $\sum_{i=0}^{N}\binom{N}{i}$이다.

하나의 프레임 안에 $i$개의 태그가 존재하는 경우를 모두 고려하여 충돌과 쿼리의 기댓값을 구해보자. $N$개의 태그가 $FS$(Frame Size) 개의 프레임을 나누어 선택하는 경우를 고려할 때, 임의의 $i$번째 경우에 대하여 프레임 안에서 태그들의 변화에 따른 경우의 수 $NC_i$(Number of Case $i$)는 다음과 같다.

$$\binom{N}{f_{i1}} \cdot \binom{N-f_{i1}}{f_{i2}} \cdot \binom{N-(f_{i1}+f_{i2})}{f_{i3}} \cdot \cdots \cdot \binom{N-(f_{i1}+f_{i2}+f_{i3}+\cdots+f_{i,FS-1})}{f_{i,FS}} \tag{2}$$

$$= \prod_{j=1}^{FS} \left( N - \sum_{f_{ij}} f_{i,j-1} \right), \; f_{i1} + f_{i2} + \cdots + f_{i,FS} = N$$

결국 $N$개의 태그가 프레임을 선택하는 경우의 총 가짓수 $TNC$(Total Number of Case)는 다음과 같이 구할 수 있다.

$$TNC = \sum (NC_i) \tag{3}$$

예를 들어 $N$= 4이고, $FS$= 4일 때, 한 프레임에 태그들이 존재할 수 있는 경우의 수는 모두 16가지이므로, 각각의 상황에 따른 충돌과 쿼리의 기댓값을 구할 수 있다. 또한 4개의 태그가 2개의 프레임 안에 들어갈 수 있는 모든 경우는 ({0, 4}, {1, 3}, {2, 2}, {3, 1}, {4, 0})의 5가지이다. 각각의 경우에 대한 $NC_i$는 (1, 4, 6, 4, 1)이고 따라서 태그들이 프레임을 선택하는 모든 경우의 수 $TNC$는 16가지이다. <표 3>은 이 결과를 보여준다.

**Table 3.** FQT 알고리즘에서 $f_{ij}$의 예($N$= 4, $FS$= 2)

| | Frame 1 | Frame 2 | 태그의 변화/프레임 |
|---|---|---|---|
| $\{f_{11}, f_{12}\}$ | 0 | 4 | ${}_4C_0 \cdot {}_4C_4 = 1$ |
| $\{f_{21}, f_{22}\}$ | 1 | 3 | ${}_4C_1 \cdot {}_3C_3 = 4$ |
| $\{f_{31}, f_{32}\}$ | 2 | 2 | ${}_4C_2 \cdot {}_2C_2 = 6$ |
| $\{f_{41}, f_{42}\}$ | 3 | 1 | ${}_4C_3 \cdot {}_1C_1 = 4$ |
| $\{f_{51}, f_{52}\}$ | 4 | 0 | ${}_4C_4 \cdot {}_0C_0 = 1$ |

태그들이 프레임을 선택하는 모든 경우에 대한 충돌과 쿼리의 합을 $TNC$로 나누어 주면 태그 수가 $N$일 때 충돌과 쿼리의 기댓값을 다음과 같이 구할 수가 있다. 식에서 $C_i$는 $i$에서의 충돌 횟수, $f_{ij}$는 $ij$에 속한 태그의 수, $Q_{ij}$는 $ij$에서의 쿼리수를 나타낸다.

$$TC = \frac{\sum_{i=1}^{NC} \sum_{j=1}^{FS} \{C_i \cdot f_{ij} \cdot NC_i\}}{TNC} \tag{4}$$

$$TQ = \frac{\sum_{i=1}^{NC} \sum_{j=1}^{FS} \{Q_{ij} \cdot f_{ij} \cdot NC_i\}}{TNC} \tag{5}$$

## 5. 성능 분석

### 5.1 해석적 방법에 의한 분석 결과

본 절에서는 태그 수가 4, 8일 때 프레임 사이즈를 2, 4로 고정하여 FQT 알고리즘과 IFQT 알고리즘의 성능을 수리적 분석과 시뮬레이션을 이용하여 구하고 이를 비교한다. 태그의 ID가 길어지면 발생하는 상황들이 기하급수적으로 늘어나기 때문에 태그 개수가 4인 경우만을 상세하게 설명한다. 태그 개수가 8일 경우에도 경우의 수가 증가할 뿐, 태그 개수가 4일 경우와 마찬가지로 설명이 가능하다.

편의상 태그 ID를 '00', '01', '10', '11'로 가정하여 $N$=4인 경우를 살펴보자. <표 4>는 $N$= 4일 때, 한 프레임 안에 들어갈 수 있는 태그의 개수와 충돌과 쿼리의 기댓값이다. 또한 <그림 7>은 $N$= 4일 때, 태그들이 프레임을 선택하는 모든 경우를 보여준다. FQT 알고리즘에서 IFQT 알고리즘보다 많은 충돌과 쿼리가 발생하는 것을 볼 수 있다.

**Table 4.** 충돌과 쿼리의 기댓값($N$= 4, $FS$= 2)

| FQT | 충돌 기댓값 | 쿼리 기댓값 |
|---|---|---|
| 0개/Frame | 0.00 | 1.00 |
| 1개/Frame | 0.00 | 1.00 |
| 2개/Frame | 1.33 | 3.67 |
| 3개/Frame | 2.00 | 5.00 |
| 4개/Frame | 3.00 | 7.00 |
| NTF | 16 | |

| IFQT(짝수 프레임) | 충돌 기댓값 | 쿼리 기댓값 |
|---|---|---|
| 2개/Frame | 1.00 | 3.00 |
| NTF | 1 | |
| IFQT(홀수 프레임) | 충돌 기댓값 | 쿼리 기댓값 |
| 2개/Frame | 1.00 | 3.00 |
| NTF | 1 | |

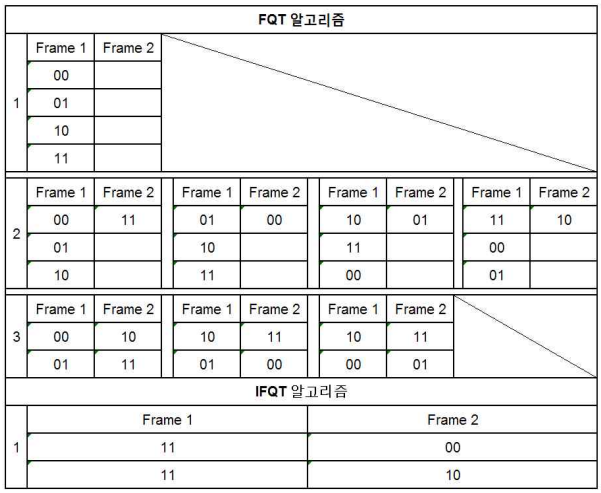


**Figure 7.** 태그가 프레임을 선택하는 모든 경우($N$= 4)

<표 4>와 <그림 7>을 이용하여 가능한 모든 경우의 FQT 알고리즘과 IFQT 알고리즘의 기댓값을 구할 수 있다. <표 5>는 그 결과를 보여준다.

<표 6>은 태그 수가 4이고, 프레임 사이즈가 2일 때의 FQT 알고리즘과 IFQT 알고리즘을 분석한 결과이다. FQT의 성능(충돌과 쿼리의 합)은 9.125이고, IFQT의 성능은 8.000으로 제

안하는 알고리즘이 약 14% 정도 좋은 성능을 보여주는 것을 알 수 있다.

**Table 5.** 충돌과 쿼리의 기댓값($N$= 4, $FS$= 2)

| FQT | | | | | | | |
|---|---|---|---|---|---|---|---|
| 프레임 1 | 프레임 2 | 프레임 1 | | 프레임 2 | | 충돌 | 쿼리 |
| | | 충돌 | 쿼리 | 충돌 | 쿼리 | | |
| 0 | 4 | 0.00 | 1.00 | 3.00 | 7.00 | 3.00 | 8.00 |
| 1 | 3 | 0.00 | 1.00 | 2.00 | 5.00 | 2.00 | 6.00 |
| 2 | 2 | 1.33 | 3.67 | 1.33 | 3.67 | 2.67 | 7.33 |
| 3 | 1 | 2.00 | 5.00 | 0.00 | 1.00 | 2.00 | 6.00 |
| 4 | 0 | 3.00 | 7.00 | 0.00 | 1.00 | 3.00 | 8.00 |
| 평균 | | | | | | 2.375 | 6.750 |

| IFQT | | | | | | | |
|---|---|---|---|---|---|---|---|
| 프레임 1 | 프레임 2 | 프레임 1 | | 프레임 2 | | 충돌 | 쿼리 |
| | | 충돌 | 쿼리 | 충돌 | 쿼리 | | |
| 2 | 2 | 1 | 3 | 1 | 3 | 2 | 6 |

**Table 6.** $N$= 4, $FS$= 2일 때의 성능 비교

| | 성능 비교($N$= 4, $FS$= 2) | | |
|---|---|---|---|
| | 충돌 | 쿼리 | 충돌+쿼리 |
| FQT | 2.375 | 6.750 | 9.125 |
| IFQT | 2.000 | 6.000 | 8.000 |
| 성능비교 | 18.74% | 12.50% | 14.06% |

### 5.2 시뮬레이션 분석 결과

앞서 언급한 바와 같이 태그의 수가 조금만 증가하여도 수리적 분석을 이용한 성능 분석은 현실적으로 불가능하다. 따라서 본 연구에서는 시뮬레이션을 이용하여 FQT 알고리즘과 IFQT 알고리즘의 성능을 비교하였다.

태그 ID는 10비트를 가정하고 최대 태그의 개수를 1024개로 가정하였다. 프레임 사이즈는 8, 16, 32, 64, 128, 256으로 가정하고, 성능 척도는 앞서와 마찬가지로 충돌과 질의응답과정의 횟수, 즉 충돌과 쿼리의 합을 사용했다. 성능평가를 위한 시뮬레이션 환경은 C++언어를 기반으로 모델링하였고, 편의상 각각의 경우에 대하여 100회의 시뮬레이션을 수행하여 얻은 결과의 평균값을 최종 결과로 사용하였다.

**Table 7.** 태그 개수 추정

| 태그 추정 | |
|---|---|
| 실제 태그 개수 | 추정한 태그 개수 |
| 25 | 24.92 |
| 50 | 50.93 |
| 100 | 100.56 |
| 200 | 199.96 |

<표 7>은 SB를 이용하여 태그 개수를 추정한 결과이다. 실제 태그 개수를 25, 50, 100, 200으로 가정하여 SB를 적용한 시뮬레이션 결과이다. 추정한 태그 개수가 실제 태그 개수와 매우 근접하는 것을 알 수 있다. 따라서 모든 실험 초기에 SB를 이용, 태그 개수를 추정하여 프레임 사이즈를 결정하였다.

**Table 8.** 프레임 사이즈에 따른 시스템 비용($N$= 25)

| | FQT | IFQT |
|---|---|---|
| Frame size = 8 | 80.72 | 73.85 |
| Frame size = 16 | 77.77 | 71.56 |
| Frame size = 32 | 85.18 | 14.37 |

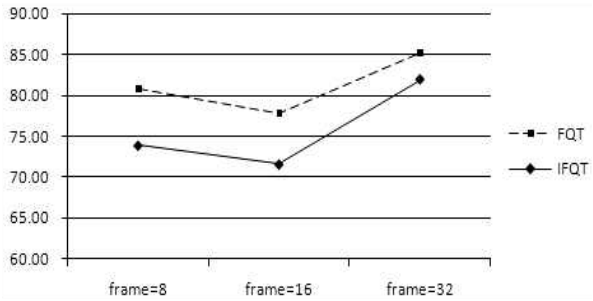


**Table 8.** 프레임 사이즈에 따른 시스템 비용($N$= 25)

태그가 선택할 수 있는 프레임 사이즈는 고정되어 있기 때문에 추정한 태그 수를 토대로 적절한 프레임 사이즈를 결정해야 한다. <표 8>과 <그림 8>은 25개의 태그가 있는 상황을 가정하고 프레임 사이즈를 4에서 32까지 변화시키면서 시스템 비용($CS$)을 분석한 결과를 보여준다. 프레임 사이즈가 태그 수의 1/2인 12.5개 보다 많은 16으로 했을 경우에 충돌과 쿼리 개수의 합이 71.56으로 가장 좋은 시스템 비용을 보여주는 것을 알 수 있다.

**Table 9.** 프레임 사이즈에 따른 시스템 비용($N$= 32)

| | FQT | IFQT |
|---|---|---|
| Frame size = 8 | 105.99 | 98.13 |
| Frame size = 16 | 101.03 | 94.04 |
| Frame size = 32 | 101.91 | 96.78 |

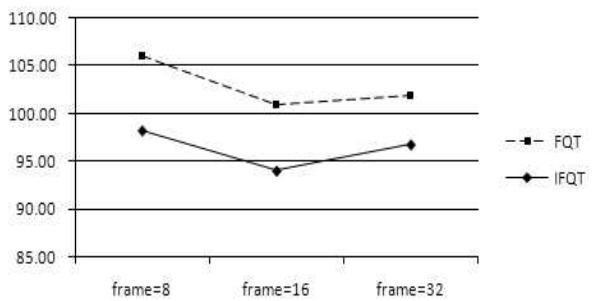


**Figure 9.** 프레임 사이즈에 따른 시스템 비용($N$= 32)

<표 9>와 <그림 9>는 태그 수가 32개이고, 프레임 사이즈가 8, 16, 32일 때를 분석한 결과이다. 프레임 사이즈가 태그 수의 1/2에 가까울 때 좋은 시스템 비용을 보여주는 것을 알 수 있다.

일반적으로 태그 수를 추정한 결과가 $2^n$ 형태일 경우에는 프레임 사이즈를 태그 수의 1/2로 할 경우에 성능이 좋아진다. 태그 수를 추정한 결과가 $2^n$이 아닐 경우에는, 태그 수 추정치의 1/2에 해당하는 프레임 사이즈보다는 더 큰 값으로 프레임의 수를 할당하는 것이 좋은 성능을 나타내므로, 태그 수의 1/2의 프레임 사이즈보다 큰 $2^n$에 해당하는 프레임을 할당한다. 이 결과를 바탕으로 프레임 사이즈를 추정한 태그 수의 1/2보다 많은 프레임 사이즈를 결정한다. 예를 들어 50개의 태그로 추정된다면 프레임 사이즈는 50의 1/2보다 큰 2의 배수인 32를 할당한다.

<표 10>과 <그림 10>은 태그 개수를 16, 32, 64, 128로 변화시키면서 FQT 알고리즘과 IFQT 알고리즘을 비교한 시뮬레이션 결과를 나타낸다. 10비트의 태그 ID를 갖는 16, 32, 64, 128개의 태그를 인식하기 위하여 프레임 사이즈를 8, 16, 32, 64로 실행한 결과 FQT 알고리즘에서는 그 시스템 비용이 47.16, 101.03, 209.07, 429.27이고, IFQT 알고리즘에서는 42.50, 94.04, 198.79, 414.93로 나타나는 것을 볼 수 있다 결국 IFQT의 시스템 비용이 FQT에 비하여 10.96~3.46% 좋아지는 것을 알 수 있다.

**Table 10.** FQT와 IFQT의 시스템 비용

| | FQT | IFQT | 비용비교 |
|---|---|---|---|
| frame = 8(태그 16개) | 47.16 | 42.50 | 10.96% |
| frame = 16(태그 32개) | 101.03 | 94.04 | 7.43% |
| frame = 32(태그 64개) | 209.07 | 198.79 | 5.17% |
| frame = 64(태그 128개) | 429.27 | 414.93 | 3.46% |

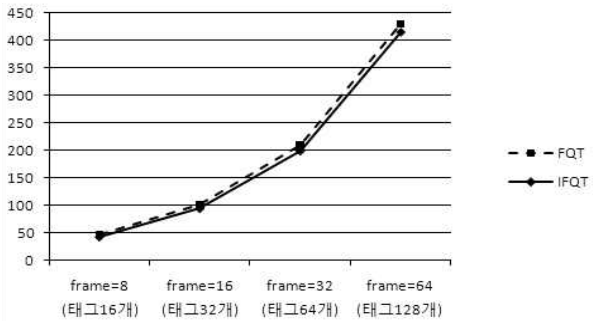


**Figure 10.** FQT와 IFQT의 시스템 비용

<표 10>과 <그림 9>를 살펴보면, 태그 개수가 많아짐에 따라 IFQT 알고리즘의 시스템 비용이 FQT에 비하여 크게 증가하지 않는 것을 볼 수 있는데, 이는 태그의 수와 프레임의 개수가 늘어남에 따라 태그가 프레임을 선택하는 경우의 수도 늘어나게 되면서 각 프레임에 2개 이상의 태그가 할당되는 경우가 FQT와 IFQT 둘 다 증가하게 되면서 IFQT의 장점이 잘 드러나지 않게 되기 때문이다. 결국 태그 개수가 많아지면 개선의 정도가 다소 약해지는 것을 알 수 있으며 이러한 점을 보완한 연구가 추가적으로 필요해 보인다.

## 6. 결 론

본 연구에서 RFID 시스템의 충돌방지를 위한 하이브리드 알고리즘을 대상으로 개선 방법을 제안하고 그 성능을 분석하였다.

각각의 태그가 여러 프레임 중 하나를 선택한 후 각 프레임별로 쿼리트리 방법을 사용하여 태그를 인식하게 되는 하이브리드 알고리즘에서는, 여러 태그가 몇몇 프레임에 집중되어 해당 프레임의 태그 인식시간이 길어지는 경우가 종종 발생할 수 있다. 본 연구에서는 기존 하이브리드 알고리즘의 성능을 개선하기 위하여, 태그 ID 앞부분의 비트 값에 따라 특정 그룹의 프레임을 선택하도록 유도함으로써 모든 프레임에 태그를 골고루 분산시키는 방법을 제안하고 그 성능을 분석하였다. 시뮬레이션 및 해석적 방법을 이용하여 성능을 분석한 결과, 제안하는 방법이 기존의 방법에 비하여 프레임당 쿼리 수를 감소시켜 줌으로써 모든 태그를 인식하는 데에 소요되는 시간을 감소시키는 것을 확인할 수 있었다. 추후 태그 개수가 많아질수록 더 우수한 성능을 보일 수 있는 방법에 대하여 연구를 진행할 것이다. 또한 태그 앞부분의 비트 정보를 3비트 이상으로 늘려 성능을 개선하는 방안도 모색할 예정이다.

## 참고문헌


Finkenzeller, K. (1999), *RFID Handbook*, John Wiley & Sons.

Lee, J-M., Leem, S-H., and Um, W-S. (2004), A Study on the RFID-Based Supply Chain Management, *Proc. 2004 Autumn Conf. of the Korean Institute of Industrial Engineers*, 1-6.

Kim, J-G., and Lee, J-K. (2006), Trend of Anti-Collision Technology for UHF RFID System, *Information and Communications Magazine*, **23**(12), 93-106.

Shin, J-D., Yeo, S-S., and Cho, J-S. (2007), Hybrid Tag Anti-Collision Algorithms in RFID System, *Journal of Korea Information and Communications Society*, **32**(4), 358-364.

Schoute, F-C. (1980), Control of ALOHA Signalling in a Mobile Radio Trunking System, *International Conference on Radio Spectrum Conservation Techniques*, 38-42.

Wieselthier, J-E., Ephremides, A., and Michaels, L-A. (1989), An Exact Analysis and Performance Evaluation of Framed ALOHA with capture, *IEEE Transactions on Communications*, **37**(2), 125-137.

Quan, C-H. and Kim, H-C. (2003), Identification Algorithm of Multiple RFID Tags for EPC Network, *Journal of Korean Society for Internet Information*, **4**(4), 27-37.

Lim, I-T. (2006), Enhanced Query Tree Based Anti-Collision Algorithm for Multiple Tag Identification, *Journal of Korea Multimedia Society*, **9**(3), 307-314.

ISO/IEC (2004a), *Information Technology-Radio Frequency Identification for Item Management-Part 6 : Parameters for Air Interface Communications at 860MHz to 960MHz.*

EPCglobal (2005), *EPCTM Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz~960MHz Version 1.0.9.*

ISO/IEC (2004b), *Information Technology-Radio Frequency Identification for Item Management-Part 7: Parameters for Air Interface Communications at 433MHz.*

Auto-ID Center (2003), *13.56MHz ISM Band Class 1 Radio Frequency Identification Tag Interface Specification, Candidate Recommendation, Ver. 1.0.0.*

Schoute, F-C. (2002), Dynamic Frame Length ALOHA, *IEEE Transactions on Communications*, 31.

Vogt, H. (2002), Multiple Object Identification with Passive RFID Tags, *IEEE International Conference on Systems, Man and Cybernetics.*

Law, C., Lee, L., and Siu, K-Y. (2000), Efficient Memoryless Protocol for Tag Identification, *Proc. of the 4th ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 75-84.

Ko, Y-E., Oh, K-W., and Bang, S-I. (2007), Adaptive Decision Algorithm for an Improvement of RFID Anti-Collision, *IEEK(Institute of Electronics Engineers of Korea) Journal*, 44(4), 1-9.

사진

**최태정**

전북대학교 산업공학과 학사
전북대학교 산업정보시스템공학과 석사
현재 전북자동차부품산업혁신센터 연구원

**서재준**

서울대학교 산업공학 학사
서울대학교 산업공학 석사
POSTECH 산업공학 박사
현재 : 한밭대학교 산업경영공학과 교수
관심분야 : 정보통신, 신뢰성공학, 추계적 과정

**백장현**

서울대학교 산업공학과 학사
서울대학교 산업공학과 석사
서울대학교 산업공학과 박사
현재 : 전북대학교 산업정보시스템공학과 교수
관심분야 : 정보통신, 경영과학, 응용통계