JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 8 : 1517~1530 Aug. 2017

초기 프레임 크기 예측을 통한 DFSA(Dynamic Frame Slotted Aloha) 알고리즘 성능 개선

이강원* · 이문형 · 이현교 · 임경희

Performance Improvements of DFSA(Dynamic Frame Slotted Aloha) Algorithm through Estimation of Intial frame Size

Kang-Won Lee* · Moon-Hyung Lee · Hyun-Kyo Lee · Kyoung-Hee Lim

Department of Industrial and Information System Engineering, Seoul National University of Science & Technology, Seoul 01811, Korea

요 약

기존의 충돌 방지 알고리즘들은 현 프레임내의 충돌 정보를 토대로 다음 프레임내의 슬롯 수를 결정하기 때문에 충돌 정보가 전혀 없는 초기 프레임의 슬롯 수는 결정할 수가 없다. 따라서 이들 알고리즘들은 초기 슬롯 수를 임의의 상수로 설정해 사용하고 있는데 알고리즘의 효율이 초기 슬롯 수에 민감하다는 연구결과를 감안하면 이는 개선 할 필요성이 있다. 본 연구에서는 이를 위해 효율적인 초기 프레임 슬롯 수 결정을 위한 두 가지 방법을 제안하였는데 이를 통해 Dynamic Frame Slotted Aloha Algorithm의 성능 개선을 시도하였다. 본 연구에서 제안한 알고리즘의 성능을 검증하기 위하여 2.4 GHz RFID 시스템을 사용하였으며 성능 측도로 사용한 Throughput과 Delay Time을 유도하기 위해 시뮬레이션 모형을 Java로 구축하였다. 본 연구에서 제안한 방법을 통해 Throughput은 9.6%, Delay Time은 9.8% 개선됨을 확인하였다.

ABSTRACT

Traditional anti-collision algorithms determine slot size of initial frame based on the information of number of collision slots , idle slots, and success slots. Since there is no information about collision at the beginning of tag information collection, traditional anti-collision algorithms can not determine the initial frame size. Considering that performance of anti-collision algorithm is very sensitive to initial slot size traditional anti-collision algorithms need some improvements. In this study two methods are proposed to determine slot size of initial frame efficiently, through which we can improve the performance of dynamic frame slotted aloha algorithm. To verify the performance of proposed algorithms, 2.4GHz RFID system is used. Throughput and delay time are derived through simulation, which is developed using JAVA. We have seen that proposed algorithm improves throughput by 9.6% and delay time by 9.8%.

키워드 : 충돌 방지 알고리즘, 사물인터넷, Schoute, NEDFSA

Key word : Anti-collision algorithm, IoT, Shoute, NEDFSA

Received 04 April 2017, **Revised** 12 April 2017, **Accepted** 04 May 2017

* **Corresponding Author** Kang-Won Lee(E-mail:kwlee@seoultech.ac.kr, Tel:+82-2-970-6476)
Department of Industrial and Information Systems Engineering, Seoul National University of Science & Technology, Seoul 01811, Korea

Open Access https://doi.org/10.6109/jkiice.2017.21.8.1517 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 사물인터넷 기술은 수많은 비즈니스 방식에 변화를 일으키면서 점차 전 산업을 리모델링하고 있다. 초기 사물인터넷 시장은 기술 위주의 작은 성공 사례를 바탕으로 형성되었으나 점차 산업별 대형 서비스 중심으로 변화하고 있다. 산업 리모델링의 주요 추진 영역으로 스마트 팩토리, 스마트 헬스케어, 스마트 카, 스마트 홈, 스마트 에너지, 스마트 팜, 스마트 물류[1-3]등이 있다. 또한 전 세계적으로 사물인터넷 서비스 제공을 위해 다양한 통신 기술이 사용되고 있다. 가장 많이 활용되고 있는 WiFi, ZigBee, Bluetooth, RFID, LTE 기술 외에도 다양한 LPWA(Low Power Wide Area) 사물인터넷 통신 기술이 등장하고 있다. 따라서 이들을 효율적으로 사용하는 것이 무엇보다도 중요해졌다.

사물 인터넷 환경에서는 수많은 무선통신 태그들이 (사물 인터넷 환경에서 센서나 칩, 디바이스 등을 나타내는데 본 연구에서는 이를 통괄적으로 태그로 칭하였다) 전송하는 정보를 실시간으로 수집하는 것이 중요한 이슈이다. 리더들은(사물 인터넷 환경에서 태그들이 전송하는 정보를 수신하는 장치를 통괄적으로 리더라고 칭하였다)‘Shared Wireless Channel'을 통해서 태그들의 정보 전송을 수신하기 때문에 다수개의 태그들이 동시에 정보를 전송 할 때 서로 간 충돌(Collision)은 불가피하다. 따라서 다양한 환경 하에서 매우 많은 태그들이 정보 전송을 원하는 사물 인터넷에서 효율적인 충돌 방지(Anti-collision) 알고리즘 개발은 무엇보다도 중요하다.

현 프레임내의 슬롯수를 매 프레임 마다 동적으로 결정해 나가는 DFSA(Dynamic Frame Slotted Algorithm)은 사물 인터넷에서도 가장 많이 사용되리라 기대되는 충돌 방지 알고리즘이다. 대표적으로 Schoute[4], Eom-Lee[5], Vogt[6], Chen[7] 그리고 Kim[8]등이 있는데 기본적으로 현재 프레임내의 성공 슬롯수와 충돌 슬롯 수 그리고‘idle’슬롯 수를 토대로 다음 프레임내의 슬롯수를 결정 한다. 즉 현 프레임 내에 성공 슬롯 수에 비해 충돌 슬롯 수가 많으면 다름 프레임내의 슬롯 수를 늘리고 반대로 충돌 슬롯 수가 거의 없으면 다음 프레임의 슬롯 수를 줄여서 무선 채널 사용 효율을 극대화 하는 방법이다. 한편 Q-algorithm은 연구[9]에 잘 설명되어 있는데 한 프레임내의 충돌 슬롯 수와 성공 슬롯 수를 토대로 프레임내의 슬롯 수를 2^Q 형태로 조정해 나가는 방법인데 ‘fixed Q-algorithm’과 ‘adjustable Q-algorithm’이 존재한다.

한편 위에서 언급한 충돌 방지 알고리즘들은 현 프레임내의 충돌 정보를 토대로 다음 프레임내의 슬롯 수를 결정하기 때문에 충돌 정보가 전혀 없는 초기 프레임의 슬롯 수는 결정 할 수가 없다. 따라서 이들 알고리즘들은 초기 슬롯 수를 임의의 상수로 설정해 사용하고 있는데 알고리즘의 효율(Throughput) 이 초기 슬롯 수에 민감하다는 연구결과[8]를 감안하면 이는 개선 할 필요성이 있다. 이에 Mota와 Batista[10]는 Q-algorithm을 바탕으로 하여 초기 슬롯 수를 예측하는 방법인 NEDFSA(Novel Estimation Dynamic Frame Slotted Aloha)를 제안하였다. 예측에 사용하는 슬롯 수 i 와 (이는 프레임내의 슬롯 수 보다 매우 작은 값을 갖는다) 슬롯 수의 증감을 나타내는 C, 두 개의 파라메타로 구성되는데 Mota와 Batista는 NEDFSA의 효율이 Q-algorithm에 비해 27% 상승했다는 것을 보였다.

한편 NEDFSA 에서는 최종적으로 프레임 크기(Final Q)를 추정한 후 초기 프레임 크기가 2^{Q-1}과 2^Q 사이에 존재 한다고 가정하여 0.67*Final Q 로 초기 프레임크기를 결정하였다. 이때 사용한 0.67은 어떠한 이론적 근거 없이 직관적인 경험에 바탕을 둔 값이기 때문에 받아들이기 어렵다. 또한 예측에 사용하는 슬롯 수 i 와 슬롯 수의 증감을 나타내는 C를 태그 수가 100개 일 때를 기준으로 각각 3과 1로 설정하여 사용하였다. 그런데 태그의 개수가 동적으로 변하는 상황 하에서 이 값들이 최적의 효율을 보장 한다고 말 할 수 없다.

본 연구에서는 위에서 언급한 NEDFSA의 두 가지 문제점을 보완하는 새로운 방법을 제안하고자 한다. Final Q에서 초기 프레임 크기를 예측 할 때 단순하게 0.67을 곱하는 대신에 마지막 프레임내의 충돌 슬롯 수와 ‘idle’ 슬롯수를 토대로 예측하는 방법을 제안하였다. 그리고 예측에 사용하는 슬롯 수 i 와 슬롯 수의 증감을 나타내는 C를 상수로 사용하는 대신에 프레임내의 충돌 슬롯 수와 ‘idle’슬롯수를 토대로 동적으로 결정해 나가는 새로운 방법을 제안하였다. 본 연구에서 제안한 알고리즘의 성능을 검증하기 위하여 2.4 GHz RFID 시스템을 사용하였으며 성능 측도로 사용한 Throughput과 Delay Time을 유도하기 위해 시뮬레이션 모형을

Java로 구축하였다. 제안한 방법의 우수성을 검증하기 위해 Schoute 방법과 NEDFSA 방법과 비교하였다. 제안한 방법에서 초기 프레임크기 예측을 위해 사용하는 슬롯들은 실제로는 정보 전송이 되지 않는 불필요한 슬롯들이기 때문에 태그 수가 매우 작을 경우 Throughput이 떨어지리라 예측 되는데 이때 가장 단순한 DFSA 알고리즘인 Schoute방법과 비교해 보았다.

서론에 이어 2장에서는 Shoute방법과 NEDFSA 방법을 간단하게 살펴보았으며 본 연구에서 제안한 방법을 설명하였다. 3장에서는 제안한 알고리즘의 성능을 검증하기 위한 2.4 GHz RFID 시스템을 설명하였으며 4장에서는 시뮬레이션 및 결과 분석을 다루었다. 마지막 결론은 5장에 수록 하였다.

Ⅱ. DFSA 성능개선 알고리즘

본장에서는 본 연구에서 제안하는 알고리즘을 설명하였으며 그에 앞서 제안한 알고리즘의 성능의 우수함을 보이기 위해 비교대상으로 선정한 Schoute 알고리즘과 NEDFSA 알고리즘을 간단하게 소개하였다.

2.1. Schoute 알고리즘과 NEDFSA 알고리즘

먼저 Schoute 알고리즘은 DFSA 알고리즘 가운데 가장 간단한 알고리즘이다. 본 연구에서 제안한 방법에서 초기 프레임 크기 예측을 위해 사용하는 슬롯들은 정보 전송이 되지 않는 불필요한 슬롯들이기 때문에 태그 수가 매우 작을 경우 Throughput이 초기 프레임 크기를 예측 하지 않는 다른 DFSA 알고리즘에 비해 떨어지리라 예측된다. 이때 가장 단순한 DFSA 알고리즘인 Schoute와 비교를 통해 태그 수가 매우 작을 경우 Throughput 저하의 심각성 여부를 판단 할 수 있다.

한편 초기 프레임 크기를 예측하는 NEDFSA를 제안한 Mota와 Batista[10]는 대표적인 DFSA 알고리즘의 하나인 Q-algorithm에 비해 Throughput이 27% 향상됨을 보였다. 본 연구에서 제안한 충돌 방지 알고리즘을 NEDFSA와 직접 비교하여 성능 우수성을 검증하였다.

2.1.1. Schoute 알고리즘

Schoute[4]는 통계적 관찰을 기반으로 알고리즘 성능이 최대가 되도록 실제 태그의 수와 한 프레임 내 슬롯의 수가 동일하다는 가정 하에 충돌 슬롯에 있는 태그 개수의 평균값을 2.3922로 추정하였다. 따라서 Schoute의 칩의 수 추정 식을 이용할 경우 다음 프레임의 프레임 크기는 '2.3922*충돌이 발생한 슬롯의 수'가 된다. 그런데 Schoute의 2.3922라는 값은 칩의 수와 프레임의 크기가 동일하다는 가정 하에 유도한 값이기 때문에 프레임의 크기에 비해 칩의 수가 작을 경우에는 칩의 수를 과대 추정하고 반대의 경우에는 칩의 수를 과소 추정하는 단점이 있다.

2.1.2. NEDFSA 알고리즘

선행 연구[8]에서는 최초 타임 슬롯 수의 변화에 따라 성능이 변한다는 사실을 확인하였다. DFSA의 성능은 최초 프레임 크기에 따라 변하기 때문에 최초 프레임 크기를 결정하는 과정은 아주 중요하다. 현 프레임의 충돌 상태 정보(성공 슬롯 수, 충돌 슬롯 수 그리고 'idle'슬롯 수)를 토대로 다음 프레임 크기를 조절하는 과정에 대한 연구는 많이 있지만 초기 슬롯 수를 예측하는 연구는 거의 이루어지지 않았다. 실제로 정보 전송을 원하는 태그의 수를 모르기 때문에 최초 프레임크기를 효율이 가장 극대화 돼는 태그 수와 같게 설정하는 것은 불가능하고 또 정보 전송이 이루어지기 전이라 충돌 상태 정보도 갖고 있지 않아서 이들을 토대로 한 프레임 크기 예측도 불가능하다. Mota와 Batista는 정보 전송에 사용되는 슬롯이 아닌 별도의 슬롯들을 도입하고 이들을 통해 초기 프레임 크기를 결정하는 NEDFSA[10]를 제안 하였다.

NEDFSA는 Q 알고리즘[9]을 기반으로 만들어진 DFSA 알고리즘이다. Q 알고리즘은 프레임 크기를 2^Q의 형태로 조절하는 DFSA이다. NEDFSA 알고리즘이 다른 DFSA와의 차이점은 태그의 정보 수집을 시작하기 전에 먼저 초기 프레임 크기를 예측한다는 것이다. 그림 1은 예측 방법을 나타내는 순서도를 나타내는데 예측 절차에 사용되는 파라미터 값들은 다음과 같다.

- i : 예측 시 사용되는 슬롯 수
- C : 슬롯의 증감

NEDFSA에서 이 값들로 각각 3과 1로 고정하여 사용한다.

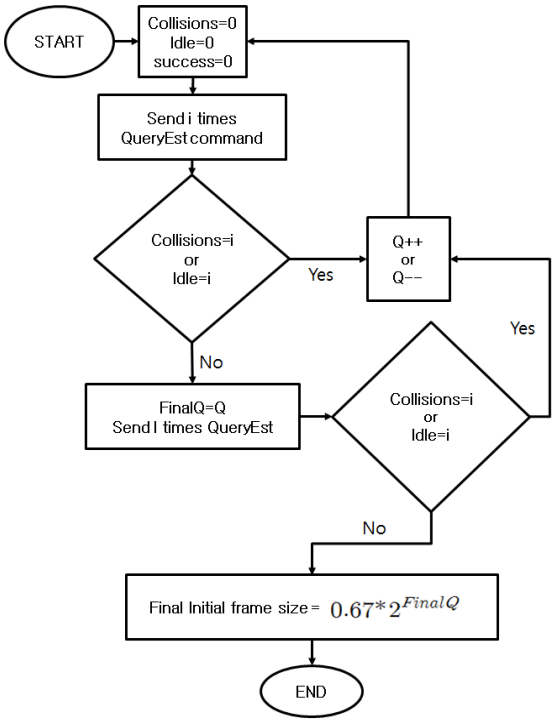


Fig. 1 NEDFSA Algorithm

2.2. 본 연구에서 제안한 알고리즘

본 연구에서는 다음의 두 가지 제안을 하여 이를 Proposal 1과 Proposal 2로 하였다.

2.2.1. Proposal 1

NEDFSA 알고리즘에서는 실제로 수집에 들어가는 초기 프레임 크기를 최종적으로 추정된 프레임 크기(Final Q)에 0.67을 곱해서 결정하였다. 이는 초기 프레임 크기가 2^{Q-1}과 2^{Q} 사이에 존재 한다고 가정하여 이론적 근거 없이 0.67을 사용하고 있는데 이는 직관적 경험에 바탕을 두고 있기 때문에 개선 할 필요성이 있다. 따라서 Final Q에서 프레임의 충돌 상태 정보를 토대로 초기 프레임 크기를 다음 식 1과 같이 결정하였다. 즉 Final Q 프레임 내에 충돌 슬롯수가 많으면 프레임 크기를 2^{Q+1}-2^{Q} 만큼 늘리고 idle 슬롯 수가 많으면 2^{Q}-2^{Q-1} 만큼 프레임 크기를 줄이는 방식이다. 이때 늘리고 줄이는 경우의 가중치는 각각 ‘collision slot # / i’와 ‘idle slot # / i’ 로 주어진다.

$$Initial\ timeslot = 2^{Q} + \frac{(2^{Q+1} - 2^{Q})}{i} \times \text{collision slot \#} - \frac{(2^{Q} - 2^{Q-1})}{i} \times idle\ slot\ \# \quad (1)$$

예를 들어 최종 슬롯 수가 128(Q =7)이고 마지막 프레임의 충돌 상태 정보가 ‘프레임 크기 = 8 슬롯, 충돌 슬롯 수 = 3, idle 슬롯 수 = 2’로 주어졌을 때 위의 식 1에 따라 계산을 해보면 초기 타임 슬롯 수를 160으로 추정하게 된다. 이는 Final Q 프레임 내에서 충돌 슬롯 수가 idle 슬롯 수에 비해 많음에도 불구하고 프레임 크기를 128에서 86으로(= 0.67*128) 줄이는 기존 방식에 비해 훨씬 타당해 보인다.

2.2.2. Proposal 2

Proposal 2는 Proposal 1에 다음 사항을 더 추가하였다.

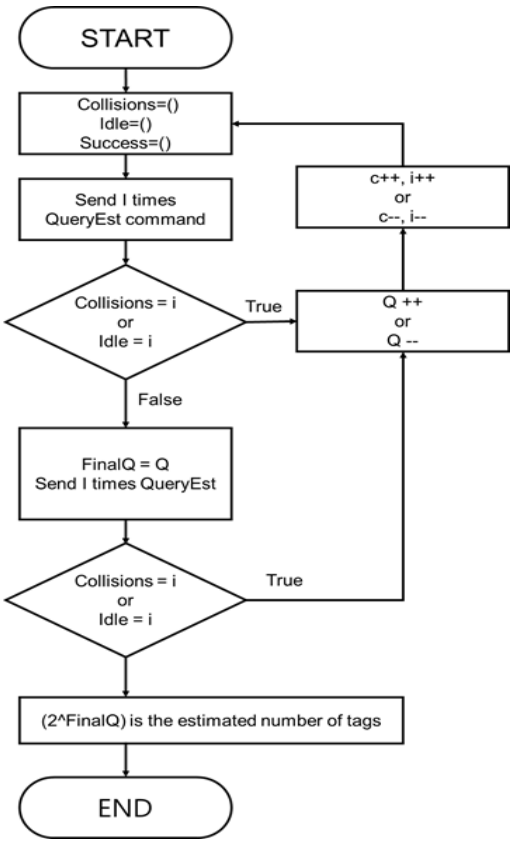


Fig. 2 Proposal 2 Algorithm

NEDFSA 알고리즘에서는 예측 시 사용되는 슬롯 수(i)와 슬롯의 증감(C)을 각각 3과 1로 고정 하였는데 이 값들은 태그수가 100개일 때를 기준으로 선정하였다. 그렇기 때문에 태그의 개수가 다를 때에는 이들 값들이 최적의 성능을 보장할 수 없다. 그러므로 파라미터 값 i와 C를 태그의 개수에 따라 동적으로 변화시키는 과정이 무엇보다도 필요하다. 따라서 본 연구에서는 NEDFSA의 파라미터 값 i와 C를 프레임내의 충돌 상태 정보를 토대로 조정하는 알고리즘을 제안 하였다. 즉 모든 슬롯에 충돌이 발생하면 태그 수가 매우 많이 존재 한다고 예측되므로 i 와 C를 1씩 증가 시키고 반대로 모든 슬롯이 idle 슬롯으로 나타나면 현재 태그 수가 많지 않음으로 판단되어 i 와 C를 1씩 감소시킨다. 다음 그림 2에 Proposal 2의 알고리즘의 순서도를 나타냈다. 이 Proposal 순서도에서 i와 C의 변화하는 과정을 삭제하면 바로 Proposal 1의 순서도가 된다.

Ⅲ. 2.4GHz 능동형 RFID 시스템

본 장에서는 2장에서 제안한 초기 프레임 크기를 예측하는 충돌 방지 알고리즘을 2.4 GHz 능동형 RFID 시스템 하에서 다른 충돌 제어 알고리즘과 성능비교를 통해 우수성을 검증해 보았다.

태그의 전원 장치가 있는 능동형 RFID 시스템은 인식거리, 인식률, 안정성 등에서 우수한 성능을 보이고 있기 때문에 건설 현장과 같이 대규모 인원의 출입 현황을 실시간으로 확인해야 하는 시스템이라든지 공항이나 항만의 팔레트 관리, 컨테이너 관리, 공장의 부품 관리 등 자산 추적 시스템에서 널리 사용되고 있다[11]. 대표적인 능동형 RFID 시스템으로는 433 MHz 대역의 주파수를 사용하는 ISO 18000-7 능동형 RFID 표준[12]이 있는데 27.8Kbps의 낮은 전송 속도와 단일 채널에 따른 간섭 문제 등으로 인해 최근의 다양한 시장 욕구를 충족시키는데 어려움이 있다.

이러한 한계를 극복하기 위하여 2.4 GHz 대역에서 다수개의 채널 및 빠른 데이터 송수신이 가능한 능동형 RFID 시스템이 제안되었다. 현재 2.4 GHz 능동형 RFID 시스템의 확산을 위해 절실히 요구되는 사항은 고속의 다중 태그 인식 속도, 신뢰성 있는 수준의 태그 인식률, 배터리 교체 없이 장기간 사용 할 수 있는 태그 등이다.

선행 논문[8]에서는 태그의 인식 속도와 인식률, 그리고 Throughput 등의 성능을 개선하기 위하여 Query 명령을 이용한 단순화된 Collection과 Ack 절차를 제안하였다. 아래의 그림 3은 태그의 수집 과정을 나타낸다. 태그의 수집 절차는 “Collection command → Tag Response → Collection command & Sleep command to

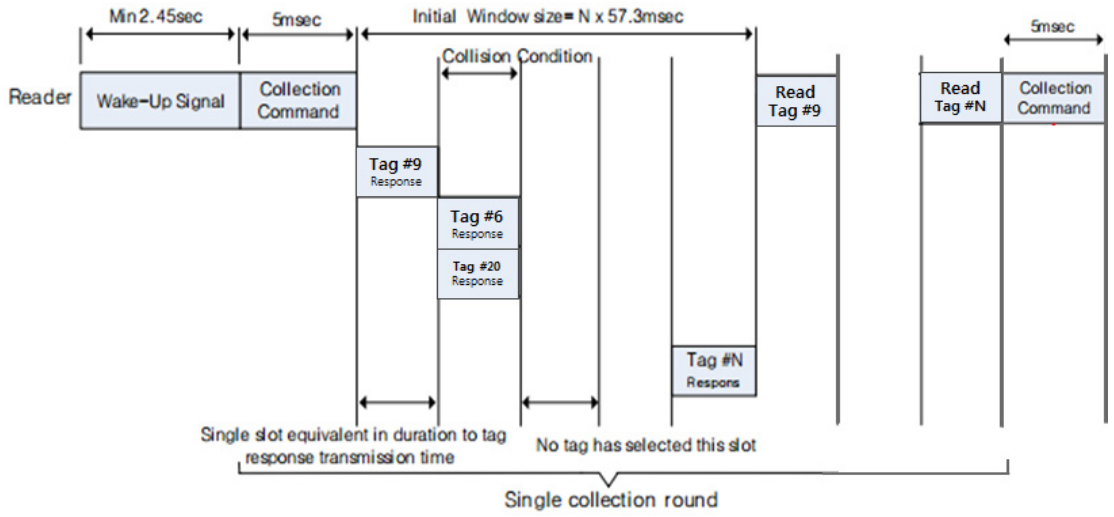


Fig. 3 RFID Tag Collection Process

Identified tag"와 같다. 이 절차는 모든 태그들의 정보가 수집될 때까지 진행된다. 리더가 브로드캐스팅을 통해 태그에게 정보를 요청하는 것을 Collection command 라고 하는데 이는 Query, QueryRepeat, QueryAdjust로 나누어 볼 수 있다. Query는 맨 처음 리더가 태그에게 정보를 요청하는 수집 명령이고 QueryRepeat와 QueryAdjust는 각각 다음 수집 라운드에서 프레임 크기를 이전과 같게 하는 경우와 다르게 하는 경우의 리더의 수집 명령이다. Tag Response는 태그가 선택한 타임 슬롯의 시작부분에서 리더로 응답 메시지를 전송하는 과정이다.

한편 태그의 정보 수집과정에서 현 라운드의 프레임의 출동 상태 정보를 토대로 다음 라운드의 프레임의 크기를 결정하기 위해서 앞의 2장에서 설명한 Schoute 알고리즘을 이용하였다.

Ⅳ. 시뮬레이션 수행 및 결과 분석

시뮬레이션 모형은 Java 언어로 구축하였다. 본 장에서는 시뮬레이션을 위해 사용한 모형 및 입력 자료와 시뮬레이션 결과를 분석하였다. 입력 자료는 시뮬레이션을 위해 필요한 2.4 GHz RFID의 Query 명령과 타임 슬롯 크기 등의 파라미터 값들 등을 포함한다.

4.1. 시뮬레이션 모형

본 연구에서는 2장에서 제안한 초기 프레임 크기를 예측하는 충돌 방지 알고리즘의 성능을 보다 정확하게 분석 할 수 있는 시뮬레이션 방법을 사용하였다. 본 연구에서 사용한 시뮬레이션 모형의 절차를 살펴보면 그림 4와 같다.

Step 1: 정보 전송을 원하는 Active 태그의 개수를 임의의 구간(Small, Medium, Large)으로 나누었다. Small 구간은 1~100개, Medium 구간은 100~1,000개, Large 구간은 1,000~2,000개의 태그의 개수를 나타낸다.

Step 2: 시뮬레이션 반복 횟수는 10,000회로 한다(K = 10,000).

Step 3: Active 칩의 개수는 특정 평균을 갖는 지수 분포에서 난수 L을 발생시켜 사용한다.

Step 4: Query command를 사용한 Collection과 Ack 절차의 간소화 방법과 충돌 확률을 토대로 4 가지 알고리즘 (Schoute, NEDFSA, Proposal 1, 2) 하에서 N개 태그들의 정보 전송과정을 시뮬레이션 한다.

Step 5: 시뮬레이션을 K = 10,000회 반복한 결과 값들로부터 각 알고리즘들의 평균 Throughput과 Delay Time을 계산한다. 이들에 대한 정의는 4.3절에 구체적으로 기술 하였다.

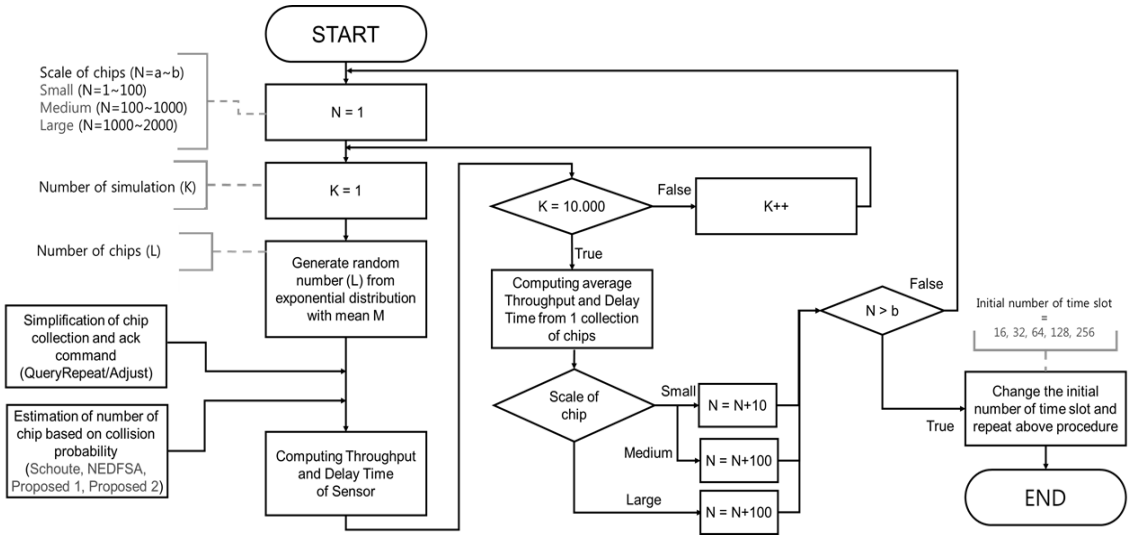


Fig. 4 Flow of simulation model

Step 6: 초기 타임 슬롯 수를 16, 32, 64, 128, 256으로 변화시켜 나가면서 Throughput과 Delay Time을 계산한다.

본 연구의 목적중 하나가 초기 타임 슬롯 수를 예측하는 것과 그렇지 않은 경우에 대하여 성능을 비교해보는 것이다. 따라서 초기 타임 슬롯 수를 예측하지 않는 Schoute 알고리즘과 예측하는 NEDFSA, 본 연구의 제안 방법인 Proposal 1, 2와 비교해보았다. 이 때 태그 수집 과정에서 다음프레임 크기 예측은 모두 Schoute 알고리즘을 사용하였다.

4.2. 입력 파라미터

본 시뮬레이션 모델에서 사용한 입력 자료들은 기본적으로 ISO 18000-7을 토대로 하였다. 다만 433 MHz 대역의 전송 속도가 아닌 2.4 GHz 대역의 전송 속도에 맞도록 일부 데이터를 수정하였다.

제안하는 프로토콜의 명령어 포맷과 기본적인 파라미터 값은 표 1과 같다. 다만 상황에 따른 추가적인 데이터 전송이 필요한 QueryRepeat과 QueryAdjust 명령은 가변적인 파라미터 값을 갖는다. 아래 표의 QueryRepeat와 QueryAdjust의 v는 한 라운드에서 정보 전송에 성공한 태그의 수에 Slot-bit(s)/8을 더한 값이다. QueryAdjust의 경우에는 새로운 슬롯 수를 나타내는 1 byte의 NewQ가 추가로 더해진다.

4.3. 성능 지표

4.3.1. Throughput

ALOHA 기반의 충돌 방지 알고리즘이 사용하고 있는 RFID 시스템의 처리율은 태그 수집 과정에서 각 라운드의 프레임크기의 합인 총 슬롯 수와 그 중에서 인식이 성공한 슬롯의 비로 정의할 수 있다. 따라서 RFID 시스템의 효율은 다음 식 2와 같이 계산할 수 있다.

$$System\ Efficiency = \frac{Success\ Slot\#}{Idle\ Slot\#+Success\ Slot\#+Collision\ Slot\#} \quad (2)$$

4.3.2. Delay Time

Delay Time은 리더의 최초 Collection 명령으로부터 리더의 수집범위 안에 있는 모든 태그들의 정보를 수집하는데 걸리는 시간으로 정의한다.

4.4. 시뮬레이션 결과

충돌 제어 알고리즘의 4가지 방법(Schoute, NEDFSA, 본 연구에서 제안한 Proposal 1, Proposal 2)에 대하여 초기 타임 슬롯 수 Q0(= 16, 32, 64, 128, 256)를 변화시키면서 Throughput과 Delay Time을 시뮬레이션을 통해 구하였다. 이중 본문에는 Q_0 = 16, 64 그리고 256인

Table. 1 Parameters for query command and tag response

<table>
<tr><td colspan="12">Tag response : Data byte = 17 bytes, Time = 544μs</td></tr>
<tr><td>Reader ID</td><td>Seq. no</td><td>Tag ID</td><td>RN-8</td><td>Battery status</td><td>RSSI (LQI)</td><td>CRC</td></tr>
<tr><td>2 bytes</td><td>1 byte</td><td>8 bytes</td><td>1 byte</td><td>1 byte</td><td>2 bytes</td><td>2 bytes</td></tr>
<tr><td colspan="12">Query : Data byte = 10 bytes, Time = 320μs</td></tr>
<tr><td>CMD</td><td>Reader ID</td><td>Seq. no</td><td>Target</td><td>Gab-time</td><td>Slot-time</td><td>Q</td><td>CRC</td></tr>
<tr><td>0x55</td><td>2 bytes</td><td>1 byte</td><td>1 byte(0x00)</td><td>1 byte</td><td>1 byte</td><td>1 byte</td><td>2 bytes</td></tr>
<tr><td colspan="12">QueryRepaet : Data byte = (7+v) bytes, Time = 32*(7+v)μs</td></tr>
<tr><td rowspan="2">CMD</td><td rowspan="2">Reader ID</td><td rowspan="2">Seq. no</td><td rowspan="2">Length of bit</td><td colspan="4">Slot-bit(s)</td><td colspan="3">Random number(s)</td><td rowspan="2">CRC</td></tr>
<tr><td colspan="4">Change according Q</td><td colspan="3">List as many as “1”</td></tr>
<tr><td rowspan="2">0x66</td><td rowspan="2">2 bytes</td><td rowspan="2">1 byte</td><td rowspan="2">Q</td><td>0</td><td>1</td><td>…</td><td>Q-1</td><td>RN1</td><td>RN2</td><td>…</td><td rowspan="2">2 bytes</td></tr>
<tr><td colspan="12">QueryAdjust : Data byte = (8+v) bytes, Time = 32*(8+v)μs</td></tr>
<tr><td rowspan="2">CMD</td><td rowspan="2">Reader ID</td><td rowspan="2">Seq. no</td><td rowspan="2">New Q</td><td rowspan="2">Length of bit</td><td colspan="4">Slot-bit(s)</td><td colspan="3">Random number(s)</td><td rowspan="2">CRC</td></tr>
<tr><td colspan="4">Change according Q</td><td colspan="3">List as many as “1”</td></tr>
<tr><td rowspan="2">0x77</td><td rowspan="2">2 bytes</td><td rowspan="2">1 byte</td><td rowspan="2">1 byte</td><td rowspan="2">Q</td><td>0</td><td>1</td><td>…</td><td>Q-1</td><td>RN1</td><td>RN2</td><td>…</td><td rowspan="2">2 bytes</td></tr>
</table>

경우만 수록하였다.

4.4.1. 최초 타임 슬롯 수(Q_0) = 16에서의 성능 비교

초기 타임 슬롯 수가 16일 때의 Throughput과 Delay Time을 4가지 알고리즘 별로 구하여 다음 그림 5에 나타냈다. Schoute 방법 하에서는 정보 전송을 위한 초기 프레임 크기를 16으로 시작 하였고 나머지 3가지 방법 (NEDFSA, Proposal 1과 2) 하에서는 초기 프레임 크기 예측을 위한 초기 프레임 크기를 16으로 하고 이로부터 초기 프레임 크기를 예측해 사용하였다. 표 2와 표 3은 각각 각 태그 개수의 구간별(Small, Medium, Large) Throughput 평균값과 Delay Time 평균값을 나타낸다.

그림 5와 표 2, 표 3에서 볼 수 있듯이 태그의 개수가 Small 구간에 있을 때에는 기존의 Schoute 알고리즘이 나머지 알고리즘들 보다 Throughput이 높고 Delay Time은 짧게 나타났다. 이는 태그의 개수가 작을 때에는 초기 타임 슬롯 수를 예측하는 과정에서 슬롯 낭비가 발생했기 때문이다. 하지만 태그의 개수가 Medium과 Large 구간일 때에는 기존의 Schoute 알고리즘 보다 나머지 알고리즘들이 높은 Throughput과 짧은 Delay Time을 가진다.

이 구간에서는 프레임 크기를 예측하는데 사용한 여분의 슬롯들이 제대로 역할을 했다고 볼 수 있다. 초기 프레임크기를 예측하는 3개의 방법 중에서 태그의 규모가 Medium 구간 일 때는 본 연구에서 제안한 Proposal 2의 성능이 Large 구간일 때는 Proposal 1의 성능이 가장 좋게 나타났다.

4.4.2. 최초 타임 슬롯 수(Q_0) = 64에서의 성능 비교

초기 타임 슬롯 수가 64일 때의 Throughput과 Delay Time을 4가지 알고리즘 별로 구하여 다음 그림 6에 나타냈다. Schoute 방법 하에서는 정보 전송을 위한 초기 프레임 크기를 64로 시작 하였고 나머지 3가지 방법(NEDFSA, Proposal 1과 2) 하에서는 초기 프레임 크기 예측을 위한 초기 프레임 크기를 64로 하고 이로부터 초기 프레임 크기를 예측해 사용하였다. 표 4와 표 5는 각각 각 태그 개수의 구간별(Small, Medium, Large) Throughput 평균값과 Delay Time 평균값을 나타낸다.

최초 타임 슬롯 수(Q_0) = 16일 때와는 다르게 그림 6과 표 4, 표 5에서 볼 수 있듯이 태그의 개수가 Small 구간에 있을 때에도 기존의 Schoute 알고리즘이 나머지 알고리즘들 보다 Throughput은 낮고 Delay Time은 크게 나타났다. 이는 태그의 개수가 작음에도 불구하고 초기 타임 슬롯 수를 64로 비교적 크게 설정하여 'idle' 슬롯수가 많이 발생함에 기인한다. 반면에 초기 프레임 크기를 예측하는 방법은 처음에 예측을 위해 다수개의 슬롯을 낭비하더라도 비교적 정확하게 초기 프레임 크기를 예측하여 Throughput을 높이고 Delay Time을 줄일 수 있었다. 초기 프레임크기를 예측하는 3개의 방법 중에서는 Q_0 = 16에서와 마찬가지로 본 연구에서 제안한 Proposal 2의 성능이 Throughput과 Delay Time 모두 가장 좋게 나타났다.

4.4.3. 최초 타임 슬롯 수(Q0) = 256에서의 성능 비교

초기 타임 슬롯 수가 256일 때의 Throughput과 Delay Time을 4가지 알고리즘 별로 구하여 다음 그림 7에 나타냈다. Schoute 방법 하에서는 정보 전송을 위한 초기 프레임 크기를 256으로 시작 하였고 나머지 3가지 방법(NEDFSA, Proposal 1과 2) 하에서는 초기 프레임 크기 예측을 위한 초기 프레임 크기를 256으로 하고 이로부터 초기 프레임 크기를 예측해 사용하였다. 표 6과 표 7은 각각 각 태그 개수의 구간별(Small, Medium, Large) Throughput 평균값과 Delay Time 평균값을 나타낸다.

최초 타임 슬롯 수(Q_0) = 64 일 때와 마찬가지로 그림 7과 표 6, 표 7에서 볼 수 있듯이 태그의 개수가 Small 구간에 있을 때에 기존의 Schoute 알고리즘이 나머지 알고리즘들에 비해 Throughput은 0.1476으로 매우 낮고 Delay Time은 973.94로 가장 크게 나타났다. 이는 태그의 개수가 작음에도 불구하고 초기 타임 슬롯 수를 256으로 크게 설정하여 'idle'슬롯수가 많이 발생함에 기인한다. 반면에 초기 프레임 크기를 예측하는 방법은 처음에 예측을 위해 다수개의 슬롯을 낭비하더라도 비교적 정확하게 초기 프레임 크기를 예측하여 Throughput을 높이고 Delay Time을 줄일 수 있었다. 초기 프레임 크기를 예측하는 방법들 중에서 Throughput은 태그 개수가 Small 구간일 때를 제외하고는 세 방법 모두 비슷하게 나타났다. 태그 개수가 Small 구간일 때 Proposal 1의 Throughput 이 낮게 나타난 이유는 태그의 수가 적을 때 파라미터 값(C, i)을 동적으로

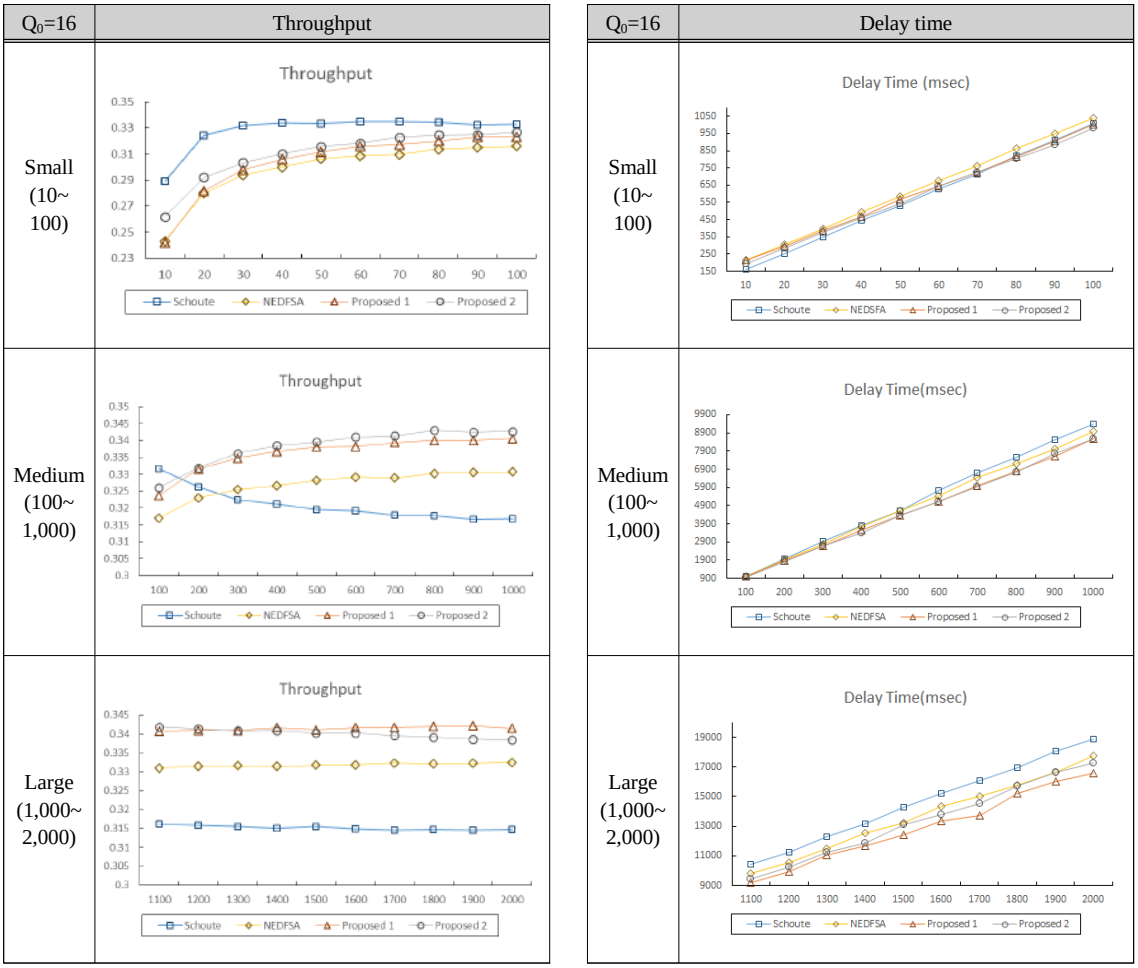


Fig. 5 Performance measures under initial number of time slot (Q_0) = 16

Table. 2 Average throughput according to the number of tags under initial number of time slot (Q_0) = 16

Throughput	Schoute	NEDFSA	Proposal 1	Proposal 2
Small (10~100)	0.3282	0.2986	0.3039	0.3100
Medium (100~ 1,000)	0.3209	0.3270	0.3363	0.3382
Large (1,000~ 2,000)	0.3152	0.3318	0.3414	0.3401

Table. 3 Average delay time according to the number of tags under initial number of time slot (Q_0) = 16

Delay time (ms)	Schoute	NEDFSA	Proposal 1	Proposal 2
Small (10~100)	582.88	629.10	601.98	589.44
Medium (100~ 1,000)	5229.22	5019.37	4752.19	4733.91
Large (1,000~ 2,000)	14656.18	13717.60	12923.49	13369.38

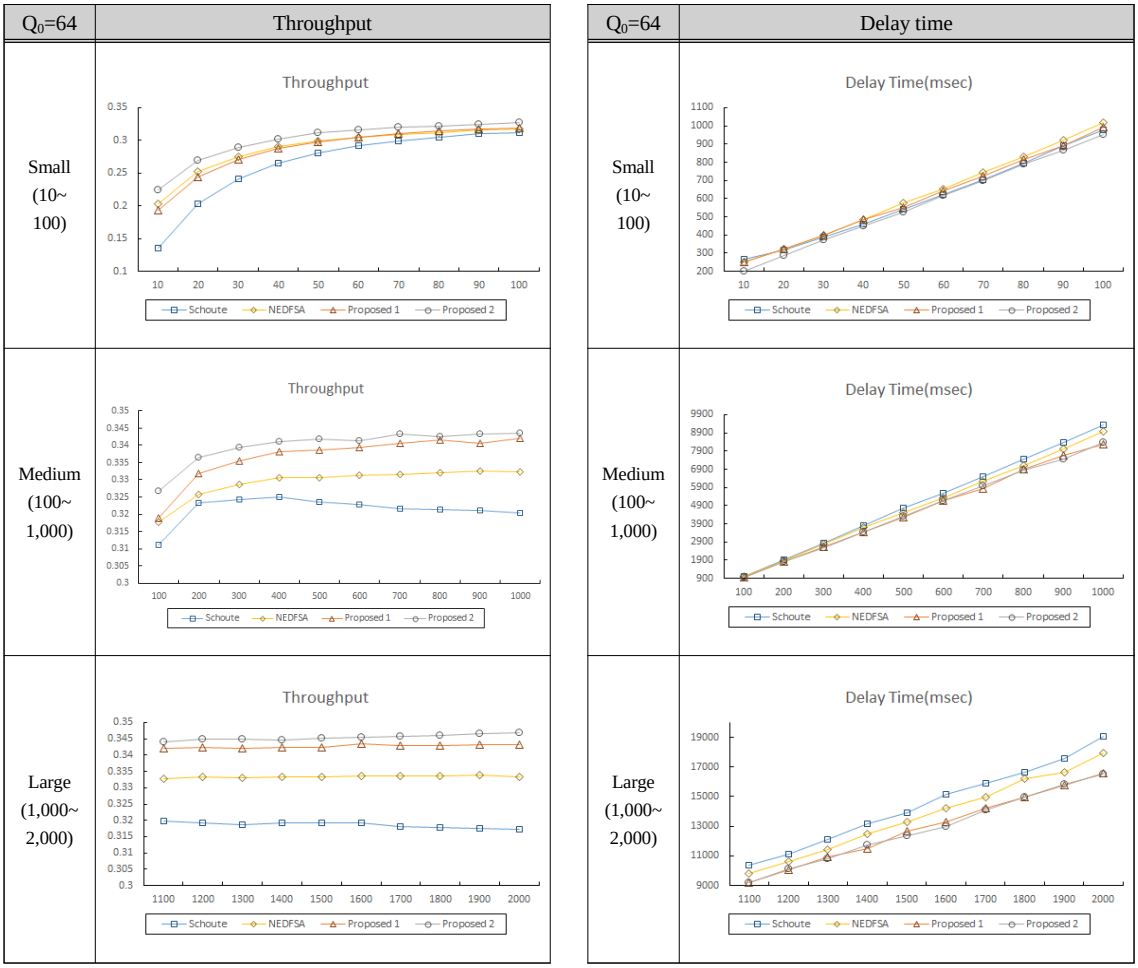


Fig. 6 Performance measures under the initial number of time slot (Q_0) = 64

Table. 4 Average throughput according to the number of tags under initial number of time slot (Q_0) = 64

Throughput	Schoute	NEDFSA	Proposal 1	Proposal 2
Small (10~100)	0.2643	0.2881	0.2860	0.3006
Medium (100 ~1,000)	0.3214	0.3293	0.3367	0.3400
Large (1,000 ~2,000)	0.3186	0.3334	0.327	0.3455

Table. 5 Average delay time according to the number of tags under initial number of time slot (Q_0) = 64

Delay time (ms)	Schoute	NEDFSA	Proposal 1	Proposal 2
Small (10~100)	595.94	620.20	606.55	576.00
Medium (100 ~1,000)	5146.99	4954.25	4697.48	4692.38
Large (1,000 ~2,000)	14496.98	13768.23	12923.15	12860.49

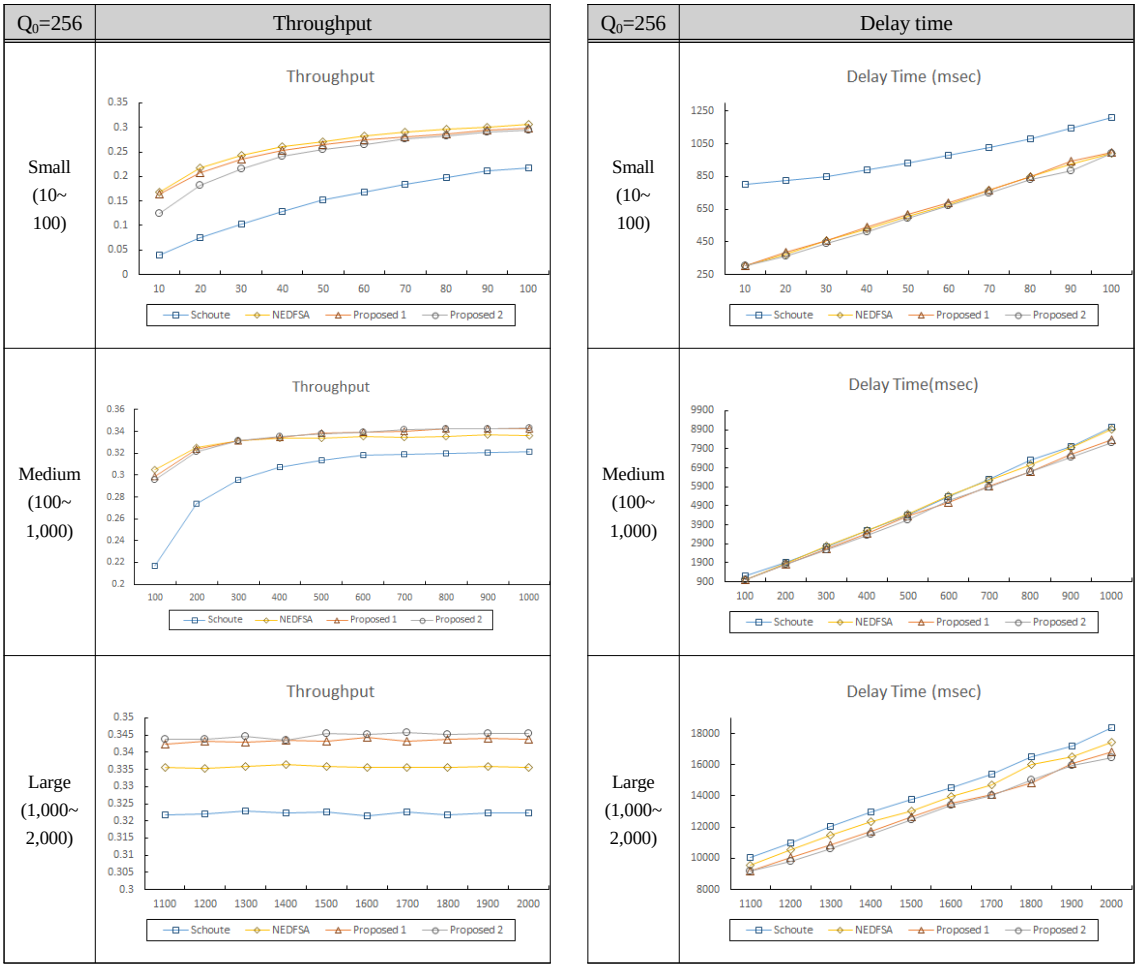


Fig. 7 Performance measures under the initial number of time slot (Q_0) = 256

Table. 6 Average throughput according to the number of tags under initial number of time slot (Q_0) = 256

Throughput	Schoute	NEDFSA	Proposal 1	Proposal 2
Small (10~100)	0.1476	0.2638	0.2557	0.2427
Medium (100 ~1,000)	0.3006	0.3308	0.3334	0.3330
Large (1,000 ~2,000)	0.3222	0.3357	0.3435	0.3448

Table. 7 Average delay time according to the number of tags under initial number of time slot (Q_0) = 256

Delay time (ms)	Schoute	NEDFSA	Proposal 1	Proposal 2
Small (10~100)	973.94	648.38	655.15	634.08
Medium (100 ~1,000)	4995.87	4931.25	4692.24	4636.35
Large (1,000 ~2,000)	14176.57	13568.17	12984.11	12857.88

변화시키는 과정에서 슬롯 낭비가 발생했기 때문이다. Delay Time의 경우에는 태그의 규모에 상관없이 Proposal 2 알고리즘이 가장 우수하게 나타났다.

4.4.4. 최종 결과

시뮬레이션 결과를 다음 표 8에 정리하였는데 태그 개수 구간별로 가장 우수한 Throughput과 Delay Time을 나타내는 알고리즘과 이때의 초기 타임 슬롯 수(Q_0)와 성능 지표를 나타냈다. 태그 개수 구간이 Small일 때에는 Schoute 알고리즘이 가장 좋은 성능을 나타냈다. 이는 초기 프레임 크기를 예측하는 다른 3가지 방법에서는 태그의 개수가 작을 때에 초기 타임 슬롯 수를 예측하는 과정에서 슬롯 낭비가 발생했기 때문이다. 하지만 태그의 규모가 Medium, Large로 커질수록 제안한 Proposal 2의 성능이 우수하게 나타났다. 초기 프레임 크기를 예측하는 방법은 태그의 개수가 많을 경우에 처음에 예측을 위해 다수개의 슬롯을 낭비하더라도 비교적 정확하게 초기 프레임 크기를 예측하여 Throughput을 높이고 Delay Time을 줄일 수 있었다.

실제로는 정보 전송을 원하는 태그의 개수를 모르기 때문에 모든 태그 개수와 모든 초기 타임 슬롯 수에 대하여 Throughput과 Delay Time의 평균값을 구하여 비교 하는 것이 타당하다. 이 결과도 표 8의 마지막 줄에서 보듯이 Proposal 2가 가장 우수하게 나타났다.

Table. 8 Summary of simulation results

	Performance measures (Differences with Schoute (%))	
Number of tags	Throughput	Delay time
Small (10~100)	Schoute, Q_0 = 16 0.3282	Schoute, Q_0 = 16 582.88
Medium (100~1,000)	Proposal 2, Q_0 = 64 0.3399 (+7.5%)	Proposal 2, Q_0 = 256 4692.38 (-9.6%)
Large (1,000~2,000)	Proposal 2, Q_0 = 64 0.3455 (+8.4%)	Proposal 2, Q_0 = 256 12857.88 (-11.1%)
Average	Proposal 2 0.3235 (+9.6%)	Proposal 2 6089.73 (-9.8%)

V. 결 론

초기 프레임의 타임 슬롯의 수의 변화에 따라 DFSA 알고리즘의 성능에 영향을 미친다는 사실이 알려 졌다. 따라서 실제로 태그의 정보 수집에 들어가기 전에 최초 타임 슬롯 수를 예측하는 알고리즘이 필요하다. 본 연구에서는 초기 프레임 크기를 예측하는 NEDFSA 알고리즘을 개선하여 두 개의 알고리즘 Proposal 1 과 Proposal 2를 제안하였다. 이를 2.4 GHz Active RFID 환경에서 검증하고자 시뮬레이션을 통해 제안한 알고리즘들의 Throughput과 Delay Time을 유도 하였다.

태그의 수와 무관하게 좋은 성능을 보이는 알고리즘은 없다. 태그의 개수가 작을 때에는 기존의 Schoute 알고리즘의 성능이 좋게 나타났다. 이는 초기 프레임 크기를 예측하는 다른 방법들 하에서는 초기 슬롯 수 예측을 위해 여분의 슬롯을 사용해야 하는데 태그의 개수가 작을 때에는 그 효과가 제대로 나타나지 않기 때문이다. 그렇지만 태그의 개수가 커질수록 초기 타임 슬롯 수를 추정하는 과정이 효과를 나타내는 것을 확인할 수 있었는데, 이 때 기존의 NEDFSA 알고리즘보다 제안한 알고리즘들의 성능이 좋게 나타났다. 이는 Final Q 프레임 내에 충돌 슬롯수가 많으면 프레임 크기를 늘리고 idle 슬롯 수가 많으면 프레임 크기를 줄이는 방식이 효과적임을 보여준다. 또한 Proposal 1 알고리즘보다 Proposal 2 알고리즘의 성능이 대체적으로 우수하게 나타났다. 이는 NEDFSA의 파라미터 값 i와 C를 프레임내의 충돌 상태 정보를 토대로 조정하는 알고리즘의 효용성을 나타낸다고 볼 수 있다. 한편 모든 태그 수와 모든 초기 타임 슬롯 수에 대해 Throughput과 Delay Time의 평균값도 Proposal 2가 가장 우수하게 나타났다.

ACKNOWLEDGMENTS

This research was supported by Seoul National University of Science and Technology Research funds.

REFERENCES

[1] Y. Quin, O. Z. Sheng, N. J. Falker, S. Dustdar, H. Wang, and A. V. Vasilakos, “When things matter:A survey on data-centric internet of things,” *Journal of Network Computing Applications*, vol. 64, pp. 137-153, February 2016.

[2] D. Gil, A. Ferrandez, H. M. Mora, and J. Peral, “Internet of things:a review of surveys based on context aware intelligent services,” *Sensors*, vol.16, 1069, July 2016.

[3] A. Botta, W. Donato, V. Persico, and A. Pescape, “Integration of cloud computing and internet of things: a survey,” *Future Generation of Computing System*, vol.56. pp.684-700, March 2016.

[4] F. Schoute, “Dynamic frame length ALOHA,” *IEEE Transactions on Communications*, vol. 31, no.4, pp. 235-242, April 1983.

[5] J. B. Eom and T. J. Lee, “Accurate tag estimation for dynamic framed-slotted ALOHA in RFID systems,” *Communication Letters IEEE*, vol. 14, no. 1, pp. 60-62, January 2010.

[6] H. Vogt, “Efficient object identification with passive RFID tags,” in *Proceedings of IEEE International Conference of Systems, Man and Cybernetics*, vol.3, pp.98-113, August 2002.

[7] W. T. Chen, “An accurate tag estimate method for improving the performance of an RFID anti-collision algorithm based on dynamic frame length ALOHA,” *IEEE Transactions on Automation Science Engineering*, vol. 6, no. 1, pp.9-15, January 2009.

[8] J. T. Kim, J. S. Kim, and K. W. Lee, “Study on the performance improvement of active RFID System,” *The Journal of Korean Institute of Communications and Information Sciences*, vol. 40, no. 05, pp. 871-885, May 2015.

[9] V. Namboodiri, M. DeSilva, K. Deegala, and S. Ramamoorthy, “An extensive study of slotted Aloha-based RFID anti-collision protocols,” *Computer Communications*, vol.35, no.16, pp.1955-1966, September 2012.

[10] R. P. B. Mota and D. M. Batista, “A dynamic frame slotted ALOHA anti-collision algorithm for the internet of things,” in *Proceedings of. the 29th Annual ACM Symposium of Applied Computing*, pp. 686-691, March 2014.

[11] J. T. Kim and K. W. Lee, “Study on the low Low Power Consumption of Active RFID Tag System,” *The Journal of Korean Institute of Information and Communication Engineering*, vol. 19, no. 06, pp. 1419-1435, June 2015.

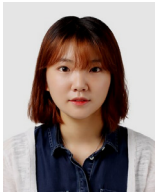
[12] ISO/ISE 18000-7, *Information Technology-radio Frequency Identification for Item Management - Part 7: Parameters for Active Air Interface Communications at 433 MHz*, pp.3-54, 2009.

이강원(Kang-Won Lee)

1980년 서울대학교 산업공학과 학사
1982년 서울대학교 산업공학과 석사
1985년 Kansas State University 산업공학과 박사
1989년~현재 서울과학기술대학교 글로벌융합산업공학과 교수
※관심분야 : 정보통신, 품질 및 신뢰성, O.R., 차세대 이동통신, RFID

이문형(Moon-Hyoung Lee)

서울과학기술대학교 글로벌융합산업공학과 산업정보시스템전공 공학사
※관심분야 : RFID, Smart Factory, SCM, Optimization



임경희(Kyoung-Hee Lim)

서울과학기술대학교 글로벌융합산업공학과 산업정보시스템전공
※관심분야 : RFID, 시뮬레이션

이현교(Hyun-Kyo Lee)

서울과학기술대학교 글로벌융합산업공학과 산업정보시스템전공 공학사
※관심분야 : RFID, 시뮬레이션