DOI: 10.3745/KIPSTC.2009.16-C.3.393

ns-2 시뮬레이터를 이용한 은닉 노드와 CCA 지연 알고리즘이 IEEE 802.15.4 네트워크의 성능에 미치는 영향 분석

현 규 완[†] · 신 연 순[††] · 안 종 석[†††] · 이 강 우[††††]

요 약

본 논문에서는 IEEE 802.15.4의 성능을 정확하게 평가하기 위하여 ns-2에 추가한 두 가지 기능을 소개한다. 첫째는 전체 노드의 수와 은닉 노드의 수가 정해짐에 따라 노드들의 배치를 자동으로 결정하는 은닉 노드 배치 방안과 은닉 노드로 인한 신호 충돌을 구별하는 방안이다. 둘째는 2003 표준만 구현되어 있는 현재의 ns-2 2.33 버전에 2006 표준에 기술된 CCA 지연 처리 방안을 구현하였다. 기능이 확장된 ns-2를 이용하여, 802.15.4 네트워크의 성능에 은닉 노드와 CCA 지연에 대한 처리 방안이 미치는 영향을 정확하게 분석할 수 있게 되었다. 시뮬레이션 결과 은닉 노드가 없을 때에 비하여 은닉 노드가 단 하나 존재할 때 네트워크 처리량이 약 66% 감소하며, 충돌율은 65%에서 90%로 급증한다. CCA 지연 처리 알고리즘의 2003 표준과 2006 표준에 따르면, 충돌 확률은 약 19%까지 차이를 보이고, 처리량은 약 38% 차이를 보이며 2006 버전이 우수한 성능을 보인다

키워드 : 802.15.4, ns-2, 은닉 노드, 전송 충돌, CCA 지연

Analysis of Effects of Hidden Nodes and CCA Deferment Algorithm on IEEE 802.15.4 Performance Using ns-2 Simulator

Gyu-Wan Hyun[†] · Youn-Soon Shin[††] · Jong-Suk Ahn[†††] · Kang-Woo Lee[††††]

ABSTRACT

This paper introduces two functions added to the current version of ns-2 simulator for better accuracy of IEEE 802.15.4 network simulations. The first one is to automatically place hidden nodes over the ring topology in which the coordinator is centered, when the number of hidden nodes and total number of nodes is given. Collisions of signals can be distinguished into the trace file according to the ways of participation of hidden nodes. The second one is the CCA deferment algorithm described in IEEE 802.15.4-2006 standard which is not implemented in the current version of ns-2. Owing to these additional functions, we can carry out the precise analysis of the performance effects of hidden nodes and CCA deferment algorithm on 802.15.4 networks. Simulation results present at least 66% of performance degradation in throughput and drastic increase of collision probability up to 90% from 65% by just a single hidden node. Besides, 2006 standard for CCA deferment algorithm gives 19% lower collision probability and 38% higher performance.

Keywords : IEEE 802.15.4, ns-2, hidden nodes, signal collisions, CCA deferment

1. 서 론

저속 무선 근거리 개인 네트워크(LR-WPANs: Low Rate -Wireless Personal Area Networks)를 지원하는 IEEE 802.15.4 표준[1, 2]은 전원을 교체하기 어려운 무선 센서 네트워크 환경에서 효율적으로 동작될 수 있도록 제정되었다. 또한 센서 노드가 이동성이 거의 없으며 한 셀 안에서는 IEEE 802.11[3] 랜의 접근점(access point)역할을 하는 PAN(Personal Area Network) 조정자(coordinator)를 통해서만 통신이 이루어진다고 가정한다.

802.15.4에서는 에너지를 효율적으로 사용하기 위하여 비콘 프레임(beacon frame)에 의해 동기화되는 슈퍼프레임(superframe) 단위로 동작하도록 규정한다. 슈퍼프레임의 CAP(Contention Access Period)에서는 노드들이 CSMA/CA(Carrier Sense Multiple Access with Collision

※ 본 연구는 지식경제부 및 정보통신연구진흥원의 IT핵심기술개발사업의 일환으로 수행하였음. (2008-F-041-01, 학습 및 진화를 통한 S/W, H/W 적 재설계가 가능한 지능로봇 기술 개발).
※ 본 연구는 지식경제부 및 정보통신연구진흥원의 지원을 받아 수행되었음. (08-기반-13, IT특화연구소 : "유비쿼터스 신기술 연구센터" 설립 및 운영).
† 준 회 원 : 동국대학교 정보통신공학과 석사과정
†† 준 회 원 : 동국대학교 정보통신공학과 박사과정
††† 종신회원 : 동국대학교 컴퓨터공학과 교수
†††† 종신회원 : 동국대학교 정보통신공학과 부교수(주저자)
논문접수 : 2009년 1월 19일
수 정 일 : 2009년 3월 10일
심사완료 : 2009년 3월 10일

Avoidance) 알고리즘을 이용하여 채널의 상태를 감지하여 전송 가능성을 확인한 후 패킷을 경쟁적으로 전송한다. 이 때, 노드들이 서로의 신호를 감지하지 못하는 공간에 위치하고 있는 은닉 노드(HN, Hidden Nodes)에서 패킷을 전송하는 경우에는, 채널이 사용 가능하다고 잘못 판단함에 따라 패킷을 전송하게 되어 예기치 않은 충돌이 발생한다. 이에 패킷 재전송으로 인하여 처리량(throughput)이 저하되고 전력의 추가소모가 발생하는 등 네트워크의 전체적인 성능이 심각하게 저하된다.

하지만 802.15.4에서는 은닉 노드의 영향을 피할 수 있는 어떠한 메커니즘도 제공하지 않는다. 또한 성능 평가에 있어서도 은닉 노드의 영향을 분석한 연구는 그리 많지 않다. 802.15.4에서의 은닉 노드 문제를 해결하기 위해 [4]는 RTS/CTS 메커니즘을 사용하지 않는 그룹화 전략을 제시하였다. 즉, 각 그룹 내의 모든 노드들이 서로에게 은닉이 되지 않도록 노드들을 그룹화 하고 각 그룹에 배타적인 시간 구간을 할당하여 전송 효율과 에너지 절감의 효과를 얻었다. [5]는 802.15.4에서 은닉 노드의 영향을 OPNET[6] 시뮬레이터를 이용하여 분석하였고, 은닉 노드로 인하여 발생하는 문제를 완화시키기 위해 MAC(Medium Access Control) 파라미터를 조정하는 방안을 제시하였다. [7]은 802.15.4에서 포화조건일 때 은닉 노드가 성능에 미치는 영향을 2차원 마코프체인을 이용하여 모델링하고 분석하였다.

이와 같이 은닉 노드에 관련된 소수의 연구 보고서가 있지만 은닉 노드가 전체 네트워크의 성능에 미치는 영향에 대한 연구는 많이 소개되지 않고 있다. 그 이유 중 하나는 현재 사용되고 있는 ns-2[8]가 은닉 노드의 영향을 분석하는데 필요한 핵심적인 기능을 제공하지 않기 때문이다. 즉, ns-2에는 첫째로 주어진 네트워크에서의 은닉 노드의 수를 임의적으로 결정할 수 있는 기능이 없으며, 둘째로, 네트워크 성능을 저하시키는 패킷 충돌에 은닉 노드들이 연관되어 있는지를 규명할 수 있는 방법이 없다.

본 논문에서는, 기존의 ns-2 시뮬레이터에 이러한 기능을 추가하여 은닉 노드가 포함된 네트워크를 정확히 시뮬레이션 할 수 있도록 함과 동시에 트레이스를 이용하여 패킷 충돌의 원인 분석을 가능하게 하였다. 즉, 전체 노드의 수와 은닉 노드의 수가 정해지면, 이에 따라 노드들의 배치를 자동으로 결정할 수 있는 수식을 제안하고 이를 구현하기 위한 코드를 ns-2에 추가하였다. 또한 전송 충돌의 유형을 크게 세 가지로 분류하고 ns-2가 출력하는 트레이스에 이들을 구분하도록 하여 은닉 노드가 미치는 성능저하의 심각성을 증명하였다.

한편 비콘을 기반으로 동기화된 802.15.4 네트워크에서는 노드들이 패킷을 전송하기 전에 현재 슈퍼프레임의 잔여 구간이 패킷을 전송하기에 충분한가를 확인한다. 전송을 하기에 충분한 시간이 남지 않았다면 해당 노드는 전송을 다음 슈퍼프레임으로 미룬다. 이러한 상황을 CCA 지연(Clear Channel Assessment Deferment)이라 한다. 802.15.4에서는 CCA 지연이 발생하는 경우 다음 슈퍼프레임에서 전송을 재개하는 방법을 2003년 표준과 2006년 표준에서 서로 달리 규정하고 있으며 이 둘의 성능 또한 상이하다. 특히 [9]에서는 2003 표준 방식에서 슈퍼프레임이 작은 경우에는 CCA 지연으로 인한 성능 저하가 더욱 심각해진다는 결과를 제시하였다. 그러나 현재의 ns-2 2.33 버전에는 2006 표준에서의 CCA 지연 처리 방법이 구현되어 있지 않다. 이에 본 논문에서는 2006 표준을 따르는 방안을 추가로 구현하여 최신의 표준에 부합하는 방법으로 시뮬레이션을 수행할 수 있도록 하였다.

이러한 기능이 추가된 ns-2를 이용하여 시뮬레이션을 수행한 결과, 노드의 수가 12이고 패킷이 70 Bytes일 때, 은닉 노드가 없는 경우에는 처리량이 약 0.33이다. 하지만 은닉 노드가 단 하나만 있더라도 처리량이 약 0.11로 약 66% 정도가 급격히 감소한다. 아울러 은닉 노드의 수가 증가함에 따라 감소가 지속되며, 은닉 노드가 5개가 되면 처리량이 0에 접근하는 극단적인 상황이 발생한다. 충돌율의 경우에는, 은닉 노드가 없을 때는 약 65% 수준이지만 은닉 노드가 하나라도 존재하는 경우에는 그 값이 90%로 급증하며, 은닉 노드가 5개가 되면 충돌율이 100%에 가깝게 된다. 아울러 CCA 지연과 관련된 충돌 확률을 보면, SO(Superframe Order)의 값이 0이고 노드 수가 12인 경우, 패킷 크기가 70 Bytes일 때의 2003 표준과 2006 표준은 각각 93%와 74%로 19%의 차이가 존재한다. 또한 처리량은 노드 수가 12인 경우 SO가 0일 때 2003 표준에서는 약 0.16이지만, 2006 표준에서는 0.22로 약 38% 증가하고, SO가 1일 때 2003 표준에서는 약 0.27이지만, 2006 표준에서는 0.29로 약 7%의 차이를 보였다. 이와 같이, 본 연구를 통하여 ns-2에 추가된 기능을 이용하여, 802.15.4 네트워크의 성능에 은닉 노드나 CCA 지연에 대한 처리 방안이 미치는 영향을 정확하게 측정하고 분석할 수 있게 되었다.

본 논문의 2장은 IEEE 802.15.4의 표준에 대해 설명하고 3장에서는 ns-2 시뮬레이터가 갖는 은닉 노드와 CCA 지연에 관련된 문제점에 대해 기술한다. 4장에서는 원하는 수만큼의 은닉 노드가 생성될 수 있도록 노드들의 배치를 기하적으로 정하고, 충돌의 원인을 규명하여 은닉 노드로 인한 충돌을 트레이스 파일에서 확인할 수 있도록 구현한 코드를 설명한다. 5장에서 CCA 지연이 발생하였을 때 처리 방안을 OTcl과 C++의 코드에 변경하는 과정을 설명한다. 6장은 수정된 ns-2를 이용하여 시뮬레이션을 수행하여 결과를 분석하고 7장에서는 결론과 향후 연구 과제를 요약하며 본 논문을 마친다.

2. IEEE 802.15.4에서의 성능 문제

IEEE 802.15.4 표준은 LR-WPANs에 대한 물리계층과 MAC 부계층을 정의한다. 비콘의 사용은 선택 사항으로 되어 있으나 802.15.4의 일반적인 특성이 비콘을 사용할 때를 기반으로 정의되므로 본 논문에서는 비콘을 사용하는 환경을 가정한다. 비콘 프레임은 조정자가 노드들에게 주기적으

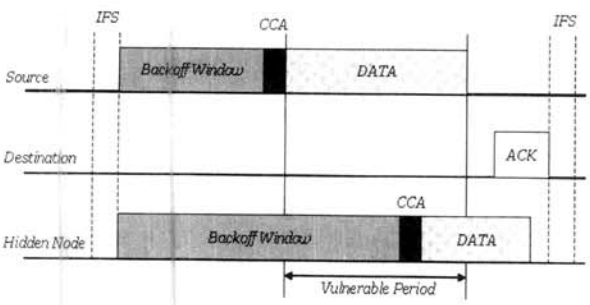
로 전송하고 그 간격은 15ms ~ 245s 이내이며, 센서 노드들의 패킷 전송에 관련된 다양한 정보들을 제공한다. 예를 들어 비콘 프레임의 주기를 결정하는 BO(Beacon Order)와 동작 구간의 길이를 결정하는 SO 등이 비콘 프레임 내에 포함된다.

2.1 802.15.4에서의 은닉 노드 문제

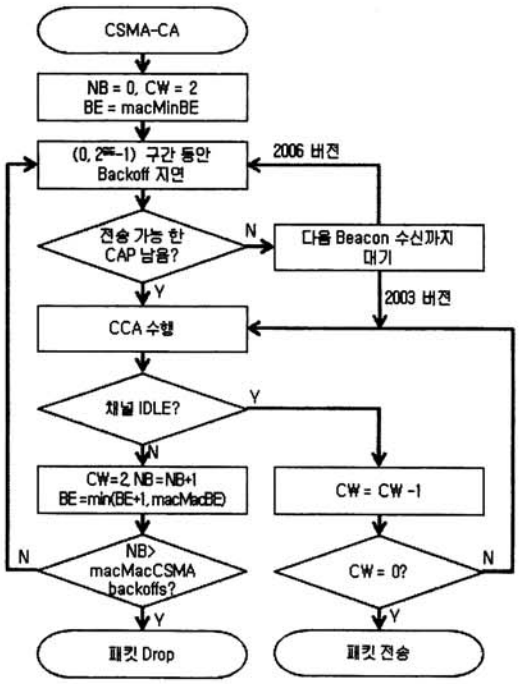
CSMA/CA를 이용할 때 임의의 노드가 패킷을 전송하고 있음에도 불구하고, 거리나 지리적 요인 등에 의하여 은닉 노드들이 전송 신호를 감지할 수 없는 경우가 흔히 발생한다. 여기서는 은닉 노드로 인한 성능 저하를 유발시키는 몇 가지 요소만을 간단히 소개한다. 은닉 노드가 있는 환경에서 전송 충돌을 유발하는 것은 CCA 단계에서의 채널감지 오류 때문이며 이와 같은 채널감지 오류가 발생할 수 있는 시간 구간을 취약 구간(vulnerable period)라고 한다. 취약 구간은 (그림 1)과 같이 은닉 노드가 전송하는 패킷의 길이와 같으므로 전송되는 패킷의 길이가 길면 취약 구간이 길어지며 이에 따라 은닉 노드로 인한 전송 충돌 확률도 증가한다. 또한, 패킷의 길이 외에 전송 충돌에 영향을 주는 다른 주요 요인은 노드의 수와 전송 전의 백오프 지연 시간이라고 할 수 있다. 노드의 수가 많고 백오프 지연시간이 짧은 경우에는 동일한 시간 구간에 패킷을 전송할 확률이 높아지면서 더 많은 충돌이 발생한다. 이러한 빈번한 충돌은 네트워크 처리량을 심각하게 저하시킨다.

2.2 802.15.4에서의 CCA 지연 문제

802.15.4의 slotted CSMA/CA에서 노드들은 (그림 2)와 같이 전송 초기 단계에 $0 \sim (2^{BE}-1)$ 사이에서 임의로 취한 백오프 기간(BP, *aUnitBackoffPeriod*)만큼의 백오프 지연(BD, Backoff Delay)시간을 갖는다. 여기서 BE(Backoff Exponent)는 노드가 채널을 감지하기 전에 어느 정도의 BD를 경과해야 하는지를 나타내는 값이다. 패킷을 전송하고자 하는 노드들은 각자에게 주어진 BD 시간이 경과한 후, 잔여 CAP 구간이 자신의 패킷을 전송하기에 충분한지 확인한다. 만일 잔여 CAP 구간이 패킷 전송에 충분하다면 CCA단계에서 두 차례에 걸쳐 채널 상태를 감지한다. 이때 다른 센서 노드가 패킷을 전송하는 중이라면 채널을 사용(busy) 상태로 판단하고 채널 감지 후에 전송이 지연된 횟수(NB, Number of backoff) 값을 증가시키며, 채널을 감지하는 횟



(그림 1) 취약 구간(Vulnerable Period)



(그림 2) IEEE 802.15.4 표준에 따르는 CCA 지연을 포함한 CSMA/CA 흐름도

수(CW, Contention Window) 값을 초기화 한 후 다시 BD 단계로 진입한다. 한편 첫 번째 CCA에서 채널이 유휴(idle) 상태로 감지되면 CW 값을 감소하고 다시 한 번 채널을 감지하게 된다. 두 번째 CCA에서 채널이 사용 상태로 감지될 경우에는 첫 번째 CCA에서 채널이 사용 상태로 감지되었을 때와 동일한 절차를 따른다. 만일 두 번째 CCA에서도 채널이 유휴 상태로 감지되면 비로소 패킷을 전송한다. 하지만 잔여 CAP 구간이 패킷 전송에 충분하지 않다면 전송을 다음 슈퍼프레임의 CAP 구간으로 미룬다.

CCA 지연이 발생 하였을 때 다음 슈퍼프레임의 CAP 구간에서 지연된 패킷을 처리하는 방식에 대해 802.15.4-2003 표준과 802.15.4-2006 표준에는 차이가 있다. 2003 표준에서는, 지연된 패킷을 전송하고자 하는 노드들은 비콘을 수신한 즉시 두 번의 CCA 단계를 수행한다. 이 경우, 두 개 이상의 노드에 CCA 지연이 발생하였다면 그 노드들이 비콘을 수신한 즉시 CCA 단계를 수행하고 채널이 유휴 상태임을 감지하여 동시에 패킷을 전송하므로 항상 충돌이 발생한다. SO가 작을 경우에는 이와 같은 충돌의 빈도는 더 높아진다. 본 연구에 따르면 CCA 지연으로 인하여 패킷 전송을 다음 슈퍼프레임으로 미루게 되는 확률은, SO가 0이고 네트워크가 포화상태인 경우, 20%를 상회한다. 또한 CCA 지연의 발생 확률은 패킷의 길이가 길어질수록 증가한다. 노드수가 증가하고 패킷의 길이가 증가하면 초기 충돌 확률이 상승하고 이로 인하여 성능 저하는 더욱 심각해진다.

2006 표준에서는 CCA 지연이 발생한 노드들은 다음 슈퍼프레임의 시작 구간에서 CCA 단계를 즉시 수행하지 않고 다시 백오프 절차를 수행하도록 함으로써 이러한 문제를 해결하고자 하였다.

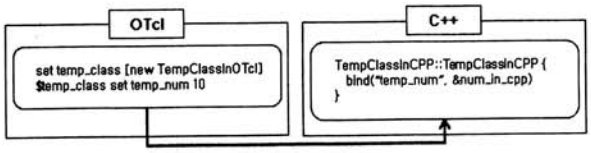
3. ns-2 시뮬레이터

현재 많이 사용되고 있는 네트워크 시뮬레이터로는 ns-2 이외에도 QualNet[10], OPNET 등이 있다. QualNet은 미국 SNT(Scalable Network Technologies)사에서 개발한 유무선 통신망을 위한 네트워크 시뮬레이터이다. 이는 확장이 쉽고 센서 네트워크와 같이 노드가 많은 네트워크에서도 빠른 처리 속도를 보인다. 또한 사용자에게 트래픽의 흐름이나 성능 관련 중요 파라미터에 대한 시각화 기능이 우수하다. OPNET은 MIT에서 미 국방성 프로젝트를 수행하며 개발한 시뮬레이터로써 모델의 구조를 쉽게 이해 할 수 있게 하였고 코드의 통합과 변경이 가능하도록 공개 소스를 제공한다. 또한 실제 벤더의 장비들에 대한 표준 모델 라이브러리를 제공하며 자신만의 모델을 만들 수도 있다. 이들 시뮬레이터들은 모두 강력한 기능들을 제공하지만 모두 라이센스 비용을 지불하여야 하고 최신 기술 동향에 따르는 모델이 적시에 제공되지 않는 등의 단점을 가지고 있다.

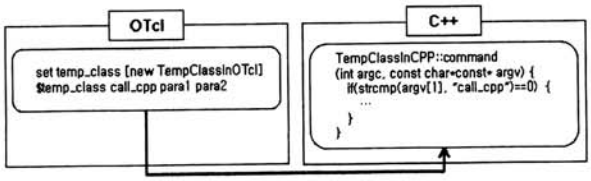
반면 ns-2 시뮬레이터는 공개 소스 기반으로 이용되고 있어 사용에 대한 제약이 없으므로 가장 많이 사용되고 있다. 사용자들은 원하는 프로토콜을 ns-2에서 사용할 수 있도록 모델링하고 개발된 모델을 공유함으로써 최신 기술동향을 빠르게 반영할 수 있다. 또한 연구 목적에 따라 쉽게 코드를 수정할 수 있으므로 기존 모델과 새로운 모델 간의 성능을 비교하기 쉬우며, 이에 따라 새로운 모델을 제시하고자 하는 연구에서 특히 많이 활용된다. ns-2는 객체 지향 기법을 사용하며 C++과 OTcl 두 개의 언어로 구현된 모듈들로 구성되어 있다.

ns-2가 두 언어로 구현되었지만 bind() 함수 또는 command() 함수를 이용하여 서로의 함수나 변수를 쉽게 공유 할 수 있다. 이 두 함수를 이용하여 프로그램을 구현하게 되면 OTcl 변수 값 변경에 따른 재컴파일을 하지 않아도 되는 이점이 있다. (그림 3)과 (그림 4)는 각각 bind()와 command() 함수를 이용하는 간단한 예시이다.

ns-2의 최신 버전 2.33[11]에는 센서 네트워크의 기본적인 MAC 프로토콜이라 할 수 있는 IEEE 802.15.4의 프로토콜이 구현되어 있다. 하지만 다음과 같은 기능들이 구현되어 있지 않아 보다 심도 있는 연구를 수행하기에는 부족한 점들이 있다. 첫 번째로 현재 ns-2 버전 2.33을 이용하여 은닉 노드에 대한 영향을 정확히 분석하기에는 한계가 있다. 연구의 목적에 따라 필요한 수만큼의 은닉 노드를 쉽게 배치할 수 있는 기능이 구현되어 있지 않으며, 아울러 전송된 신호간의 충돌이 발생했을 때, 충돌의 원인을 판단을 할 수 있는 근거를 제공하지 않기 때문이다. 두 번째로는 ns-2 버전 2.33에서 CCA 지연이 발생 했을 때에 처리방식이 802.15.4-2003 표준만이 구현되어 있으며 최근의 802.15.4-2006 표준은 구현되어 있지 않아 이에 대한 정확한 실험이 불가능하다. 이에 다음 절에서는 이들 기능을 추가로 구현하는 방법을 구체적으로 제시하기로 한다.



(그림 3) bind() 함수의 예



(그림 4) command() 함수의 예

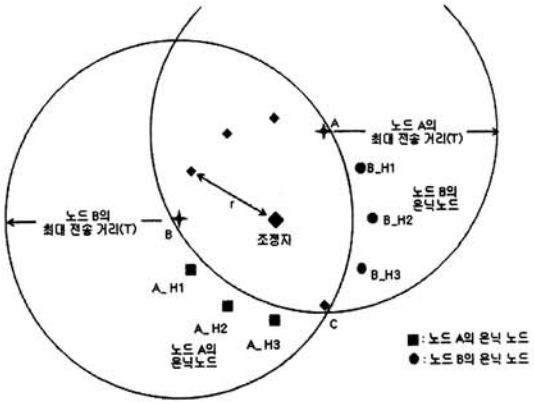
4. 은닉 노드의 임의 설정 기능 구현

4.1 전파 모델

ns-2에서 사용하는 전파 모델은 크게 네 가지로 구분된다. Nakagami 페이딩, 섀도우잉, 자유공간과 2선지면 모델(two-ray ground model)이 그것이다. 이 중에서 본 연구에서는 2선지면 모델을 가정한다. 이 모델은 수신측과 송신측 간에 직접 전송되는 전파와 지면에 반사되어 전달되는 전파를 조합하는 방안이다. 전파의 최대 수신 거리와 최대 감지 거리를 15m로 제한하는 토폴로지를 가정하고 Phy 모듈에서 감지할 수 있는 전파 세기의 한계(CSThresh_)와 수신 할 수 있는 전파 세기의 한계(RXThresh_)를 각각 8.54570e-07로 설정한다.

4.2 은닉 노드 배치 방안

본 연구에서 노드의 배치는 중앙에 하나의 조정자가 있는 스타 토폴로지를 가정한다. (그림 5)는 노드 12개, 은닉 노드 3개를 원형으로 배치했을 때 노드 배치의 예를 보여준다. 조정자가 중심에 있고 나머지 노드들이 균일한 각도를 유지하며 원형을 이루고 있다. 이 가정은 노드들이 다양한 환경에 임의로 배치되는 실제 상황과는 매우 다를 수 있지만 은닉 노드의 다양한 분포 비율에 따른 처리량 변화라는 성능



(그림 5) 반지름(r)과 전송거리(T)를 이용한 원형 배치

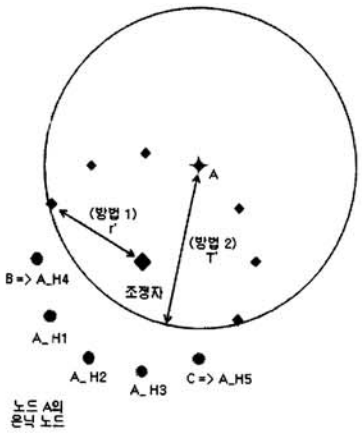
분석상의 편의를 위해 전제된 것이다. 그림에서 노드 A의 은닉 노드는 노드 A_H1, A_H2, A_H3이며 노드 B도 3개의 은닉 노드를 갖는다. 이와 같이 스타 토폴로지를 사용하면 임의의 노드가 n개의 은닉 노드를 갖게 될 때 다른 모든 노드들도 n개씩의 은닉 노드를 갖게 된다.

본 연구에서는 원하는 수만큼의 은닉 노드를 배치하는 방안에만 주안점을 두기로 하며 각 은닉 노드들의 구체적인 위치에 대한 제약 조건은 두지 않기로 한다. (그림 5)에서 조정자와 노드들 간의 반지름은 r이고 각 노드들이 송출하는 신호의 최대 전송 거리는 T로 표기하였다. 이때 은닉 노드를 늘리는 방법에는 두 가지가 있다. 첫 번째 방법은 (그림 6)에서와 같이 조정자와 노드간의 반지름을 변경하는 방법으로 전송 거리는 T로 고정하고 반지름을 r에서 $r'(r'>r)$으로 증가시키는 것이다. 두 번째 방법은 반지름 r은 고정하고 최대 전송 거리를 $T'(T'<T)$으로 줄이는 것이다. 본 논문에서는 전파 세기를 고정시키기 때문에 조정자와 노드 간의 반지름 r을 조정하여 은닉 노드의 수를 변경하는 전자의 방법을 선택하였다.

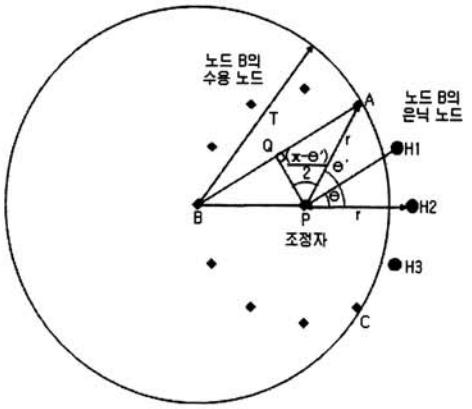
주어진 전체 노드의 수와 원하는 은닉 노드의 수에 따라 r을 결정하고 정해진 반지름을 이용하여 각 노드의 위치를 정하는 기하적인 방식을 소개한다. 이 방법은 조정자를 제외한 전체 노드의 수가 짝수인지 홀수인지에 따라 달라진다. 또한 전체 노드의 수가 짝수이면 은닉 노드의 수는 홀수가 되고 반대의 경우에는 짝수가 된다. 그 이유는 배치 상태가 원형이기에 전송거리 안에 배치할 수 있는 은닉 노드의 수가 전체 노드의 수에 따라 짝수와 홀수로 구분되기 때문이다.

먼저 전체 노드 수가 짝수이며 은닉 노드의 수가 홀수인 경우를 (그림 7)에서와 같이 12개의 노드들에 대해 각 노드에 3개의 은닉 노드가 발생하도록 노드의 위치를 결정하는 예로 설명하겠다. 최대 전송 거리를 T로 할 때 노드 B가 전송하는 신호는 노드 A와 C까지 에서만 감지되고 노드 H1, H2와 H3에서는 감지가 불가능하여 이들이 은닉 노드가 된다.

Θ는 하나의 노드와 인접한 노드와의 간격을 각도로 표현한 것으로써, 다음과 같다.



(그림 6) 반지름 변경 또는 전송거리 변경 방안



(그림 7) 전체 노드 수 짝수, 은닉 노드 수 홀수의 경우의 노드 배치

$$\Theta = \frac{2\pi}{N} \quad (N\text{은 짝수}) \qquad \text{(식 1)}$$

Θ'은 노드 B로부터 조정자를 중심으로 대칭점에 위치한 은닉 노드로부터 노드 B에 대해 은닉이 아니면서 가장 가까이 위치한 노드까지의 각도로써 (식 2)와 같다. 여기서 NH는 은닉 노드의 수이다.

$$\Theta' = \frac{N_H+1}{2}\Theta \quad (N_H\text{는 홀수}) \qquad \text{(식 2)}$$

조정자 P에서 선분 AB에 수선을 그어 만나는 점을 Q라고 하면 △AQP는 직각 삼각형이 된다. 이때 ∠APQ는 (식 3)과 같고 (식 4)를 이용하여 r을 (식 5)와 같이 구할 수 있다.

$$\angle APQ = \frac{\pi-\Theta'}{2} \qquad \text{(식 3)}$$

$$\sin\left(\frac{\pi-\Theta'}{2}\right) = \frac{\frac{T}{2}}{r} \qquad \text{(식 4)}$$

$$r = \frac{\frac{T}{2}}{\sin\left(\frac{\pi-\Theta'}{2}\right)} \qquad \text{(식 5)}$$

r을 이용하여 각 노드의 위치 (n_{ix}, n_{iy})는 (식 6)과 (식 7)로 구할 수 있다. 이때 C_x와 C_y는 조정자의 x 좌표와 y 좌표이다. 또한 i는 i번째 노드를 지칭한다.

$$n_{ix} = r\times\cos\left(\frac{2\pi\times(i-1)}{N}\right) + C_x \qquad \text{(식 6)}$$

$$n_{iy} = r\times\sin\left(\frac{2\pi\times(i-1)}{N}\right) + C_y \qquad \text{(식 7)}$$

마찬가지로 전체 노드 수가 홀수이고 은닉 노드 수는 짝수의 경우에는 (식 1)과 (식 2)에서 N과 N_H가 각각 홀수와 짝수로 제한된다. 전체 노드의 수가 짝수인지 홀수인지에

따라 배치도의 형태는 조금 다르지만 r을 구하는 식은 (식 5)와 동일하며 각 노드의 위치도 (식 6)과 (식 7)로 표현할 수 있다. 마지막으로 두 경우에 있어서 Θ와 Θ'을 다음과 같이 (식 8)로 통합하여 표현할 수 있다.

$$\Theta=\frac{2\pi}{N},\ \Theta'=\frac{N_H+1}{2}\Theta$$

(단, N이 짝수면 N_H는 홀수, N이 홀수면 N_H는 짝수)

(식 8)

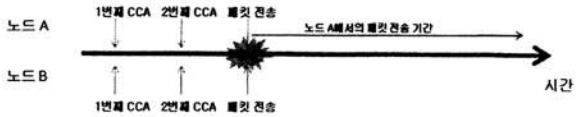
4.2.1.노드 배치 스크립트

[코드 1]은 전체 노드의 수와 은닉 노드의 수가 주어질 때 위의 식을 이용하여 노드들의 위치를 배치하는 Tcl 스크립트 파일이다. ~변수 angle, hidden_angle은 각각 Θ와 Θ'을 나타낸다. 설정 값들을 이용하여 r을 구한 후 줄번호 8~16의 명령어로 조정자와 다른 노드들의 위치를 결정한다. 조정자를 중심으로 한 각 노드의 위치를 순차적으로 배치하도록 for-loop를 이용하였다. 이렇게 만든 Tcl 스크립트 파일을 은닉 노드의 배치가 필요한 Tcl 스크립트 파일 내에서 [코드 2]와 같이 호출하여 이용할 수 있다.

4.3 전송 충돌의 분류

본 절에서는 노드들이 전송한 신호간의 충돌의 종류를 원인별로 세분하고 은닉 노드로 인해 발생할 수 있는 충돌 상황을 쉽게 확인하는 방법에 대해 설명한다.

첫 번째는 일반적인 충돌로써, (그림 8)과 같이 채널이 유휴 상태일 때 두 개 이상의 노드가 동시에 두 번째 CCA를 끝내고 동시에 패킷을 전송함으로써 발생하는 경우이다. 두 번째는 (그림 9)와 같이 은닉 노드로 인하여 발생할 수 있는 충돌이다. 즉, 노드 A가 신호를 전송했을 때 노드 A에 대한 은닉 노드가 동시에 패킷을 전송하는 상황이다. 세 번째는 일반적인 충돌과 은닉 노드로 인한 충돌이 혼합적으로 동시에 발생한 경우이다. 우선 (그림 10)과 같이 2개 이상의 노드들이 동시에 패킷을 전송 하여 충돌이 발생한다. 패킷이 전송되는 시간 중에, 만일 A, B 두 노드에 대한 은닉 노드가 존재한다면 이 시점에 CCA를 수행해서 채널이 유휴하다고 판단하여 패킷을 전송하고 또 다른 충돌이 발생하게 되는 것이다. 이외에도 혼합 충돌의 경우는 한 노드가 패킷을 먼저 전송한 후 그 노드의 은닉 노드들이 동시에 패킷을 보내는 경우와 노드들이 동시에 패킷을 전송하여 충돌이 발생하고 이때 패킷을 전송한 두 개의 노드들에 공통적으로 은닉 상태인 또 다른 두 개의 노드들이 동시에 패킷을 보낼 때, 충돌이 발생하게 된다. 하지만 본 연구에서는 이 경우를 세 번째 경우에 포함하기로 한다.



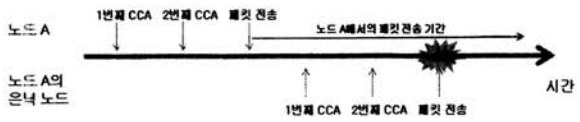
(그림 8) 채널이 유휴 상태일 때 두 개 이상의 노드들이 동시에 신호를 전송할 때의 충돌

줄번호	코드
1	set pi 3.14159265358979323846264338327 95
2	set T 15
3	set nn 12
4	set hidden_nn 3
5	set angle [expr 2 * $pi/$nn]
6	set hidden_angle [expr $angle * ($hidden_nn+1)/2]
7	set r [expr ($T/2)/sin(($pi-$hidden_angle)/2)]
8	$node_(0) set X_ 30
9	$node_(0) set Y_ 30
10	$node_(0) set Z_ 0
11	for {set i 0} {$i < $nn} {incr i} {
12	set sin_num [expr sin(2*$pi/$nn * $i)]
13	set cos_num [expr cos(2*$pi/$nn * $i)]
14	$node_($i) set X_ [expr 30 + $r * $cos_num]
15	$node_($i) set Y_ [expr 30 + $r * $sin_num]
16	$node_($i) set Z_ 0 }

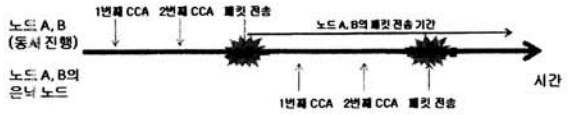
[코드 1] 은닉 노드 배치의 Tcl 스크립트 파일

줄번호	코드
1	source ./hidden.scn

[코드 2] TCL 파일 내 배치 소스 호출



(그림 9) 은닉 노드로 인한 충돌



(그림 10) 혼합 충돌 경우

이상과 같이 패킷들 간의 충돌이 발생하는 원인은 매우 다양하므로 충돌이 많은 네트워크 환경에서 적절한 대응 방안을 찾기 위해서는 신호간의 충돌마다 명확한 원인을 규명할 필요가 있다. 하지만, 현재 사용되고 있는 ns-2 버전에서는 이러한 측면에서의 기능이 매우 제한적이며 이에 본 논문에서는 기존의 ns-2에 다양한 충돌을 구분함은 물론 트레이스 정보에 이와 같은 충돌의 종류를 구분하여 출력할 수 있는 기능을 추가하고자 한다.

4.3.1. 충돌 분류를 위한 코드

충돌의 종류를 구분할 때 패킷이 전송된 시점에 대한 정보가 핵심 단서가 된다. 즉, 두 개 이상의 노드가 동시에 CCA를 수행하여 충돌하는 경우나 특정 노드가 전송한 직후 이 노드의 은닉 노드가 패킷을 전송하여 충돌이 발생하는 경우 등 모든 경우에 있어서 패킷들이 전송된 시점을 알아야 한다. 이를 위하여 p802_15_4Mac.h에 [코드 3]과 같이 전송 시점과 충돌 여부를 저장할 변수를 추가하였다.

전송신호의 충돌 확인은 물리 계층에서 이루어지므로 충돌 시점을 알기 위하여 p802_15_4phy.cc를 [코드 4]와 같이 수정한다. 줄번호 11 이하는 충돌이 발생했을 경우로써, 줄

줄번호	코드
1	class Mac802_15_4 : public Mac {
2	public:
3	...
4	static double time_to_send_data[100];
5	static char is_hidden_col;
6	static char is_col;
7	... };

[코드 3] p802_15_4mac.h 중 추가된 변수들

줄번호	코드
1	void Phy802_15_4::recv(Packet *p, Handler *h) {
2	...
3	switch(ch->direction()) {
4	case hdr_cmn::DOWN:
5	Mac802_15_4::time_to_send_data[index_] = CURRENT_TIME;
6	...
7	default:
8	...
9	if (rxPkt == 0) {
10	...
11	} else {
12	double time_to_send_1 = Mac802_15_4::time_to_send_data[p802_15_4macSA(rxPkt)];
13	double time_to_send_2 = Mac802_15_4::time_to_send_data[p802_15_4macSA(p)];
14	...
15	if (time_to_send_1 == time_to_send_2) {
16	Mac802_15_4::is_col = 1;
17	} else if (time_to_send_1 != time_to_send_2) {
18	Mac802_15_4::is_hidden_col = 1; }
19	if (p->txinfo_.RxPr > rxPkt->txinfo_.RxPr)

[코드 4] p802_15_4phy.cc 중 충돌의 종류를 구분하는 과정

줄번호	코드
1	void Mac802_15_4::recv(Packet *p, Handler *h) {
2	...
3	// drop(p,"LQI");
4	if (Mac802_15_4::is_col == 1 && Mac802_15_4::is_hidden == 1) {
5	drop(p,"H&C");
6	} else if (Mac802_15_4::is_col == 1) {
7	drop(p,"COL");
8	} else if (Mac802_15_4::is_hidden == 1) {
9	drop(p,"HID"); }
10	Mac802_15_4::is_col = 0;
11	Mac802_15_4::is_hidden = 0;
12	return; }
13	... }

[코드 5] p802_15_4mac.cc 중 트레이스 파일 생성 코드

번호 12와 13에서 이전에 수신하던 패킷과 현재 수신 중인 패킷의 전송시간을 받는다. 그리고 줄번호 15 이하에서 두 패킷이 전송된 시간을 비교한다. 또한 패킷 전송이 완료되기 전에 다른 충돌이 발생하면 패킷의 전송시간을 비교하여 충돌의 종류를 다시 확인한다.

4.3.2. 트레이스 파일 생성

충돌로 인하여 패킷을 드롭(drop)하는 과정은 MAC 계층에서 이루어지며 충돌된 마지막 패킷의 전송이 완료된 후 드롭이 발생하고 이 정보가 트레이스 파일에 기록된다. 이때 트레이스에 충돌의 종류를 나타내기 위해 p802_15_4mac.cc를 [코드 5]와 같이 수정하였다. 줄번호 4~11은 충돌의 종류를 확인하기 위해 각 패킷의 충돌 여부를 확인한다. 각각의 경우를 확인하고 drop() 함수를 이용하여 트레이스 파일에 동시 전송으로 일어난 충돌은 "COL", 은닉 노드로 인해 발생한 충돌은 "HID", 두 가지 충돌의 혼합으로 발생하였다면 "H&C"로 각각 출력한다.

(그림 11)은 12개의 노드와 3개의 은닉 노드를 시뮬레이션을 한 경우의 트레이스 파일의 일부분이다. 4번째 줄에서 조정자는 충돌이 발생하여 "D"를 표시하며 패킷이 드롭 된다. 이때 "H&C"는 은닉 노드로 인한 충돌과 일반 충돌의 혼합된 충돌을 의미한다. 마지막 줄에서 노드 1의 패킷 전송은 은닉 노드인 6번의 패킷과 충돌이 발생하여 조정자는 "HID"라는 표시를 남기며 드롭 된다. 따라서 트레이스 파일을 이용하여 충돌 원인별 발생 횟수를 측정할 수 있음은 물론 충돌이 발생한 상황을 분석할 수 있게 되었다. 이와 같이 트레이스 파일을 분석하여 처리량, 일반 충돌 횟수와 비율, 은닉 노드로 인한 충돌 횟수와 비율, 혼합 충돌 횟수와 비율 등을 정밀하게 측정할 수 있다.

5. 802.15.4-2006 CCA 지연 구현

본 연구에서는 CCA 지연을 처리하기 위한 2003 표준과 2006 표준에 대한 처리 방안의 쉬운 변경을 위하여 OTcl의 변수를 이용하기로 한다. 연동을 위한 두 함수 bind()와 command() 중에서 bind() 함수는 변수만을 연결하여 주지만 command() 함수는 변수와 함께 함수를 정의할 수 있으므로 본 연구에서는 이후의 확장에 대한 가능성을 감안하여 command() 함수를 이용하기로 한다.

5.1 OTcl 코드 변경

[코드 6]은 command() 함수를 이용하기 위해 ns_mobilenode.tcl 파일에 CCA_select 함수를 추가한 내용을 보여준다. CCA_select 함수가 호출되면 3번 줄 명령어의 실행으로 CCA_select_in_CPP와 $args를 매개변수로 하여 C++의 MAC 모듈의 command() 함수가 호출된다. [코드 7]은 CCA 지연 처리 방

```
s 360.054912035 _3_ MAC  --- 26 cbr 77 [0 0 3 800] ------- [3:1 0:14 32 0] [1] 0 0
s 360.056192035 _8_ MAC  --- 20 cbr 77 [0 0 8 800] ------- [8:1 0:20 32 0] [0] 0 0
s 360.056192035 _9_ MAC  --- 21 cbr 77 [0 0 9 800] ------- [9:1 0:20 32 0] [0] 0 0
D 360.058848071 _0_ MAC  H&C 20 cbr 77 [0 0 9 800] ------- [9:1 0:20 32 0] [0] 0 0
...
s 360.063872035 _1_ MAC  --- 38 cbr 77 [0 0 1 800] ------- [1:1 0:14 32 0] [2] 0 0
s 360.066112035 _6_ MAC  --- 23 cbr 77 [0 0 6 800] ------- [6:1 0:20 32 0] [0] 0 0
D 360.070688071 _0_ MAC  HID 38 cbr 77 [0 0 6 800] ------- [6:1 0:14 32 0] [2] 0 0
```

(그림 11) 트레이스 파일 예시

줄번호	코드
1	Node/MobileNode instproc CCA_select args {
2	$self instvar mac_
3	eval $mac_(0) CCA_select_in_CPP $args }

[코드 6] ns_mobilenode.tcl에서의 추가 사항

줄번호	코드
1	$node_(0) CCA_select 2003
2	$node_(1) CCA_select 2006

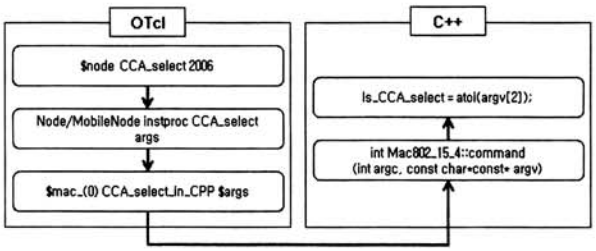
[코드 7] IEEE 802.15.4를 이용하는 Tcl 스크립트 파일 예시

안을 선택하는 Tcl 스크립트 파일의 예시이다. 즉, CCA_select 뒤에 매개변수를 추가하여 CCA 지연 처리 방식을 2003 표준과 2006 표준 중에서 하나를 선택할 수 있게 하였다.

5.2 C++ 코드 변경

5.2.1. OTcl과의 연동

[코드 8]과 같이 p802_15_4mac.cc의 클래스 Mac802_15_4의 public 영역에 CCA 지연 방안을 저장하는 변수를 추가한다. 클래스 CsmaCA802_15_4에서 CCA 지연이 발생하였을 때 클래스 Mac802_15_4의 변수 CCA_type_select를 확인하여 각 표준을 따른다. 변수 CCA_type_select에 값이 적용되는 과정은 [코드 9]를 통해 설명한다. [코드 9]는 5.1절에서의 설명과 같이 [코드 6]의 "Node/MobileNode instproc CCA_select args" 함수가 호출되면 클래스 Mac802_15_4의 command() 함수가 호출된다. 여기서 CCA 지연 처리 방안을 변수에 지정한다.



(그림 12) OTcl과 C++로 구성된 모듈들 간의 변수 연동 과정

이상의 OTcl과 C++로 구현된 모듈들에서의 변수 연동에 대한 과정을 (그림 12)에 나타내었다.

5.2.2. 전송 가능 여부 확인 단계 수정

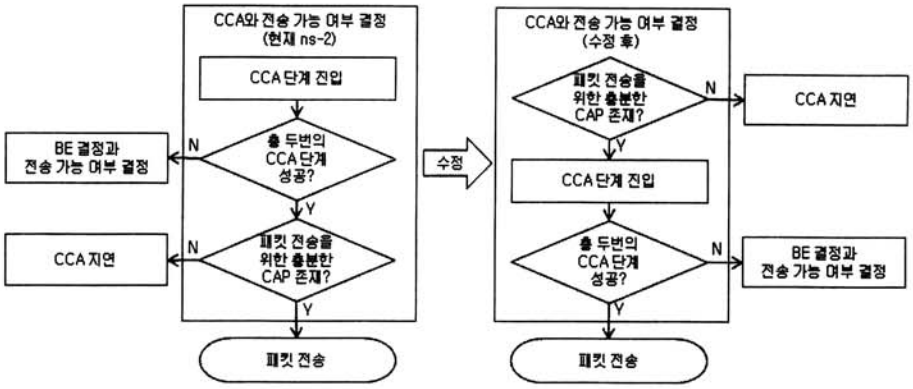
(그림 13)은 ns-2에서 CSMA/CA 알고리즘을 구현할 때 CCA 지연 여부에 따른 진행 과정을 보여준다. 현재 버전의 ns-2에 CSMA/CA 알고리즘이 구현된 내용은 실제 802.15.4 표준과는 CCA 단계 인근에서 차이가 있음을 확인할 수 있

줄번호	코드
1	class Mac802_15_4 : public Mac {
2	…
3	public:
4	…
5	int CCA_type_select;
6	}

[코드 8] p802_15_4mac.h에서의 추가된 변수

줄번호	코드
1	int Mac802_15_4::command(int argc, const char*const* argv) {
2	…
3	if (strcmp(argv[1], "CCA_select_in_CPP") == 0) {
4	CCA_type_select = atoi(argv[2]);
5	return (TCL_OK); }
6	… }

[코드 9] p802_15_4mac.cc의 command() 함수에 추가



(그림 13) ns-2에서 IEEE 802.15.4 CSMA/CA 알고리즘 변경

줄번호	코드
1	void CsmaCA802_15_4::CCA_confirm(PHYenum status) {
2	bool idle;
3	if (CW == 2) {
4	if (!canProceed(0.0)) {
5	if (beaconEnabled) CW = 2;
6	bPeriodsLeft = 0;
7	return; } }
8	…
9	if (CW == 0) {
10	// if (canProceed(0.0, true)) {
11	txPkt = 0;
12	mac->csmacaCallBack(p_IDLE); }
13	// else {
14	// if (beaconEnabled) CW = 2;
15	// bPeriodsLeft = 0;
16	// }
17	} else
18	… } }

[코드 10] p802_15_4csmaca.cc 중 전송 가능성 확인 단계

다. 즉, 표준에서는 CCA 단계로 진입하기 이전에 잔여 CAP 구간이 패킷을 전송하기에 충분히 남았는지를 확인하고 CCA 단계를 수행한다. 하지만 ns-2에서는 두 번의 CCA에서 모두 채널이 유휴 상태로 판단되었을 때에야 비로소 패킷을 전송하고 ACK를 받을 수 있을 만큼 충분한 시간이 남아 있는지를 확인한다. 따라서 두 번의 CCA를 마치고 패킷 전송에 충분한 CAP 구간이 존재하는지 검사하던 기존 방식을 표준에 부합하도록 수정하였다. [코드 10]에서는 변경된 사항을 보여준다. 줄번호 9~17은 CW가 0이 되었을 때 즉, 두 번의 CCA에서 채널이 유휴하다고 감지되었을 때 canProceed() 함수를 통해 잔여 CAP 구간동안 패킷 전송을 완료할 수 있는지 확인하는 단계이다 이 단계를 CCA보다 먼저 수행하기 위해 CCA 지연 결정을 줄번호 4의 위치로 변경한다.

5.2.3. CCA 지연 발생 후 전송 과정

[코드 11]은 각 노드가 새로운 비콘을 수신한 후 이전에 지연되었던 패킷을 전송하는 과정을 보여주는 코드이다. 새로 구현된 2006 표준도 함께 수용하기 위하여 줄번호 15과 16을 삭제하고 줄번호 5~14를 추가함으로써 2003, 2006 표준을 재컴파일 없이 사용할 수 있도록 하였다.

6. 시뮬레이션 결과

6.1 실험 환경

본 절에서는 본 연구를 통하여 기능이 추가된 ns-2를 이용한 실험 결과를 제시한다. <표 1>에서 고정 파라미터는 표준에서 정해져 있는 값으로 실험 시 변경되지 않으며, 유동 파라미터는 실험을 진행하며 상황에 따라 값을 변화시킨다. 노드에서의 패킷 발생은 포화 상황으로 가정한다.

6.2 은닉 노드로 인한 영향

본 연구에서는 은닉 노드의 수를 0, 1, 3, 5로 변경하면서,

줄번호	코드
1	void CsmaCA802_15_4::newBeacon(char trx) {
2	…
3	if (bPeriodsLeft == 0) {
4	wtime = adjustTime(0.0);
5	if (mac->is_CCA_select == 2006) {
6	wtime = (Random::random() % (1<<BE)) * bPeriod;
7	wtime = adjustTime(wtime);
8	} else {
9	wtime = adjustTime(0.0); }
10	if (canProceed(wtime));
11	if (mac->is_CCA_select == 2006) {
12	backoffT->start(wtime);
13	} else {
14	backoffHandler(); }
15	// if (canProceed(wtime));
16	// backoffHandler();
17	} else {
18	… }

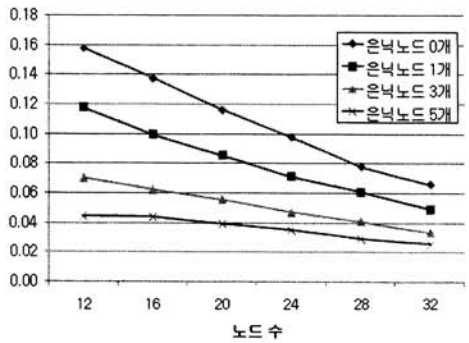
[코드 11] p802_15_4csmaca.cc 중 CCA 지연 발생 후 전송과정

〈표 1〉 시뮬레이션을 위한 파라미터 집합

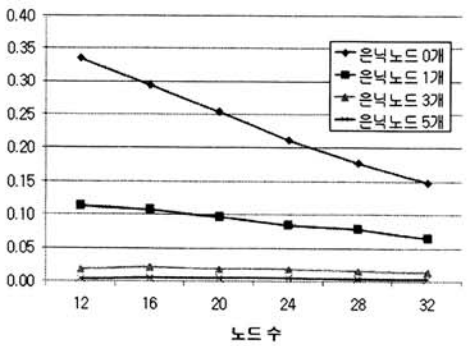
고정 파라미터	값	단위
MAC header	7	Byte
PHY header	6	Byte
ACK	11	Byte
Channel Bit Rate	250	Kbps
SIFS Period	12	symbol
LIFS Period	40	symbol
Ack_timeout	54	symbol
duty cycle	100	%
macMinBE	3	
aMaxBE	5	
macMaxCSMA	4	
유동 파라미터	**값**	**단위**
Packet Payload	20, 70	Byte
Superframe Order	0~3, 10	
Beacon Order	0~3, 10	
Node	12, 16, …, 32	개
Hidden Node	0, 1, 3, 5	개

전체 노드 수와 은닉 노드의 수가 주어지면 노드들을 자동으로 배치하는 스크립트 파일을 이용하여 노드들을 배치할 수 있어서 은닉 노드를 포함하였을 때의 원하는 상황을 정확하게 구현할 수 있도록 하였다.

(그림 14)와 (그림 15)는 SO가 3이고, 패킷 크기가 20, 70 Bytes일 때 은닉 노드의 수에 따른 네트워크 처리량의 변화를 보여준다. 처리량은 포화 조건에서 평균 단위 백오프 슬롯 동안 성공적으로 전송된 데이터 프레임의 평균 양으로 표현할 수 있으며 노드의 수와 은닉 노드의 수가 많을수록 감소한다. 또한 은닉 노드로 인한 영향은 패킷이 70 Bytes일 때가 20 Bytes일 때 보다 크다. 예를 들어 노드의 수가 12이고 패킷이 70 Bytes일 때, 은닉 노드가 없는 경우에는 처리량이 약 0.33이다. 하지만 은닉 노드가 단 하나만 있더라도 처리량이 약 0.11로 약 66% 정도가 급격히 감소한다.



(그림 14) PS=20Bytes일 때 처리량



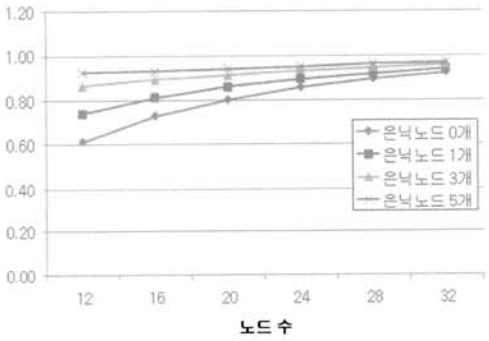
(그림 15) PS=70Bytes일 때 처리량

아울러 은닉 노드의 수가 증가함에 따라 감소가 지속되며, 은닉 노드가 5개가 되면 처리량이 0에 접근하는 극단적인 상황이 발생한다.

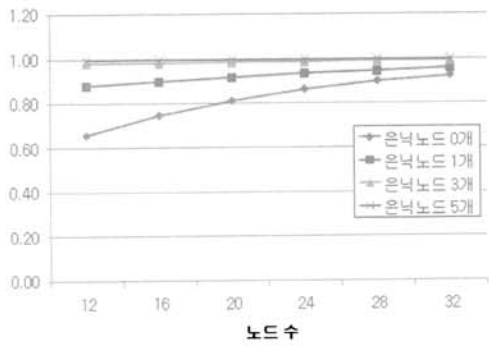
동일한 상황에서 충돌율은 (그림 16), (그림 17)과 같다. 패킷이 크면 은닉 노드의 영향이 커져 충돌 확률이 증가하는데 이는 패킷 크기가 증가할수록 전송 시간이 증가하고 그 기간에 은닉 노드 역시 패킷을 전송할 확률이 증가함으로써 충돌 확률이 증가하는 것이다. 역시 노드 수가 12개, 패킷 크기가 70 Bytes일 때, 은닉 노드가 없을 때의 충돌율은 약 65% 수준이지만 은닉 노드가 하나라도 존재하는 경우에는 그 값이 90%로 급증하며, 은닉 노드가 5개가 되면 충돌율이 100%에 가깝게 된다.

(그림 18)과 (그림 19)는 충돌의 종류 별 초당 충돌 횟수를 보여주고 있다. 은닉 노드의 수가 증가할수록 HID와 H&C의 수가 증가하는 것을 확인할 수 있다. 은닉 노드가 하나 있을 때는 H&C 충돌이 발생하지 않는데 H&C 충돌이 발생하기 위해서는 두개 이상의 공동의 은닉 노드가 있어야 하기 때문이다.

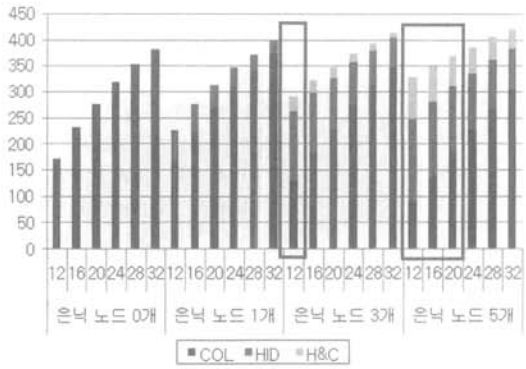
사각형 안에 있는 그래프들에서 볼 수 있는 바와 같이, 패킷의 크기가 20 Bytes일 때, 은닉 노드 수와 전체 노드의 수의 비율이 1:4를 초과하는 경우에 있어서는 은닉 노드가 포함된 충돌이 전체 충돌의 50% 이상을 차지한다. 또한 패킷의 크기가 70 Bytes일 때에는 은닉 노드의 영향이 더욱 커짐을 알 수 있다. 패킷들 간에 충돌이 발생하는 원인은 매우 다양하므로 충돌이 많은 네트워크 환경에서 적절한 대



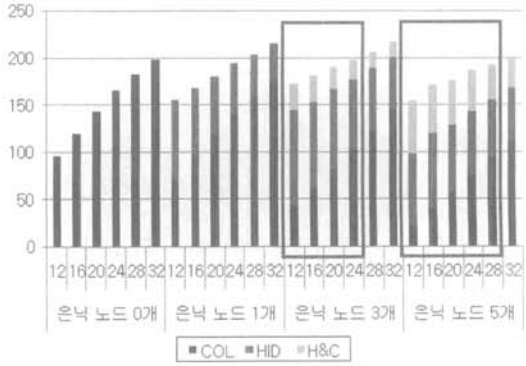
(그림 16) PS=20Bytes일 때 충돌율



(그림 17) PS=70Bytes일 때 충돌율



(그림 18) PS=20Bytes일 때 초당 충돌 횟수



(그림 19) PS=70Bytes일 때 초당 충돌 횟수

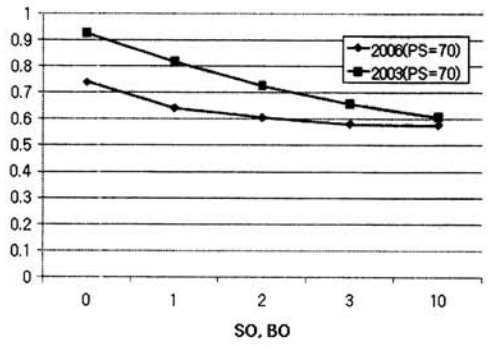
응 방안을 찾기 위해서 신호간의 충돌을 종류별로 분류하는 것은 매우 유용하다고 하겠다.

이와 같이 은닉 노드는 네트워크의 전체 성능에서 대단히 중요하고 때로는 치명적인 영향을 미침에도 불구하고, 이제까지는 은닉 노드를 포함하는 네트워크 상황을 정확하게 시뮬레이션할 수 있는 방안이 없어서 많은 연구 결과가 발표될 수 없었지만, 본 연구에서 제시한 기능들을 추가함으로써 정확하고 면밀한 실험과 분석이 가능하게 되었다.

6.3 CCA 지연에 대한 결과

(그림 20)은 은닉 노드가 없이 전체 노드의 수가 12개, 패킷 크기가 70 Bytes일 때, SO의 변화에 따른 전체 전송 시도에 대한 충돌의 확률을 보여준다. 충돌 확률은 2003 표준과 2006 표준 모두 SO가 증가할수록 감소하는데 이는 SO 값이 작을수록 CCA 지연이 더 빈번히 발생하기 때문이다. 또한 2006 표준에서의 충돌 확률이 2003 표준에 비해 더 작게 나타난다. 예를 들어 SO의 값이 0일 때 2003 표준과 2006 표준은 각각 93%와 74%로 19%의 차이가 존재한다. SO의 값이 3일 때도 7% 차이가 있다. 이는 2006 표준에서는 CCA 지연 이후 다음 슈퍼프레임의 시작 구간에서 백오프를 수행함으로써 전송을 분산시키기 때문이다.

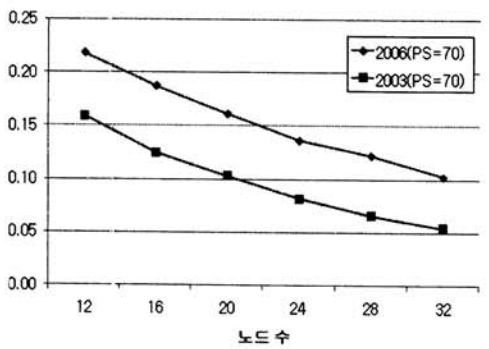
(그림 21)과 (그림 22)는 SO의 값을 각각 0과 1로 고정했을 때 노드 수의 변화에 따른 네트워크 처리량의 변화를 보여준다. 두 그래프를 비교하면 전체적으로 SO가 클수록 처



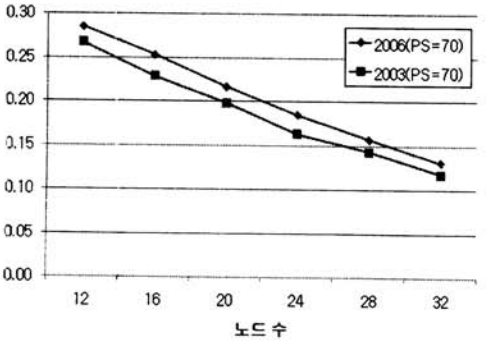
(그림 20) 충돌 확률

리량은 크게 나타나며, SO의 값이 작을 때 2003과 2006 표준의 차이가 더 크게 나타난다. 즉, 노드 수가 12일 때 두 그래프를 비교해 보면 SO가 0일 때는 2003 표준에서는 처리량이 약 0.16이지만, 2006 표준에 따르면 0.22로 약 38%의 차이를 보이고 SO가 1일 때는 2003 표준에서의 처리량이 약 0.27이지만, 2006 표준에 따르면 0.29로 약 7%의 차이를 보인다. 이는 SO가 클수록 CCA 지연의 발생 확률이 작아지기 때문이다.

이상의 결과에서 볼 수 있듯이, IEEE 802.15.4의 2003 표준과 2006 표준의 CCA 지연 확률과 처리량의 차이를 살펴보았을 때, ns-2에서 제공하는 2003 표준만을 사용한다면 2006 표준을 따르는 상황과는 성능에서 분명한 차이가 발생한다는 것을 알 수 있다. 이는 ns-2 버전 2.33의 IEEE 802.15.4를 이용하여 정확한 성능 분석을 하기 위해서는



(그림 21) SO=BO=0 일 때 처리량



(그림 22) SO=BO=1 일 때 처리량

CCA 지연 후 처리 방안에 반드시 2006 표준에 따르는 기능을 추가하여 사용될 수 있어야 함을 의미한다.

7. 결 론

본 논문에서는, 전체 노드의 수와 은닉 노드의 수가 정해지면, 이에 따라 노드들의 배치를 자동으로 결정할 수 있는 수식을 제안하고 이를 구현하는 기능을 ns-2에 추가하였다. 또한 전송 충돌을 은닉 노드가 관련되는 유형별로 구분지어 트레이스 출력하여 패킷 충돌의 원인 분석을 가능하게 하였다. 그리고 CCA 지연 후 처리 방안에 대하여 2003 표준만이 구현되어 있는 현재의 ns-2 2.33 버전에 2006 표준을 따르는 방안을 추가로 구현하였다.

이러한 기능이 추가된 ns-2를 이용하여 시뮬레이션을 수행하여, 일정한 조건 하에서, 네트워크 처리량이 은닉 노드가 없을 때에 비하여 은닉 노드가 단 하나 존재할 때에는 약 66% 정도가 감소하며 은닉 노드의 수가 증가함에 따라 감소가 지속되어 은닉 노드가 5개가 되면 처리량이 0에 접근하는 경우를 보았다. 그리고 은닉 노드가 없을 때는 약 65%이던 충돌율이 은닉 노드가 하나 존재하는 경우에는 90%로 급증하고 은닉 노드가 5개가 되면 100%에 가깝게 된다. 아울러 CCA 지연을 처리하는 방식에 있어서는, 2006 표준이 충돌 확률은 약 19%까지 차이를 보이고, 처리량은 약 38%까지 차이를 보이며 2003 표준에 비하여 우수한 성능을 보인다.

이와 같이, 본 연구를 통하여 ns-2에 추가된 기능을 이용하여, 802.15.4 네트워크의 성능에 은닉 노드나 CCA 지연에 대한 처리 방안이 미치는 영향을 정확하게 측정하고 분석할 수 있게 되었다.

향후 연구로는 다양한 네트워크 구조와 다양한 전파 모델 환경에서도 은닉 노드들을 배치하는 알고리즘을 고안하는 것이다. 또한 IEEE 802.15.4의 수학적 모델을 이용하여 이 논문에서 제시한 시뮬레이션 결과를 검증할 것이다.

참 고 문 헌

[1] Wireless Medium Access Control(MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks(LR-WPANs), IEEE Standard 802.15.4-2003.

[2] Wireless Medium Access Control(MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks(LR-WPANs), IEEE Standard 802.15.4-2006,

[3] IEEE 802.11 WG, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, 1999.

[4] L. Hwang, S. T. Sheu, Y. Y. Shih, and Y. C. Cheng, "Grouping Strategy for Solving Hidden Terminal Problem in IEEE 802.15.4 LR-WPAN". Proc. of the 1st International Conference on Wireless Internet (WICON'05), pp.26-32,

2005.
[5] A. Koubaa, M. Alves, B. Nefzi, and Y. Q. Song, "Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks," Workshop on Real Time Networks RTN'06, 2006.
[6] OPNET homepage, http://www.opnet.com
[7] Youn-Soon Shin, Gyu-Wan Hyun, Jong-Suk Ahn, Hie-Cheol Kim, Kang-Woo Lee, "An Analytical Model for LR-WPAN Performance in the Presence of Hidden Nodes". The KIPS Transactions: Part C, Vol.16-C, No.1, pp.133-142, 2009.
[8] ns-2 official Site, http://www.isi.edu/nsnam/ns
[9] A. Koubaa, M. Alves, and E.Tovar, "A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks," Workshop on Factory Communication Systems WFCS, Torino, Italy, 2006.
[10] QualNet homepage, http://www.scalable-networks.com
[11] ns-allinone-2.33.tar.gz, http://downloads.sourceforge.net/nsnam/ns-allinone-2.33.tar.gz?modtime=1206990403&big_mirror=0

현 규 완
e-mail : redwany@paran.com
2007년 동국대학교 정보통신공학과(학사)
2007년~현 재 동국대학교 정보통신공학과 재학(석사 과정)
관심분야 : 컴퓨터 구조, 컴퓨터 시뮬레이션, 네트워크 시뮬레이션

신 연 순
e-mail : ysshin@dgu.edu
1999년 동국대학교 전산통계학과(학사)
2002년 동국대학교 정보통신공학과(공학석사)
2007년~현 재 동국대학교 정보통신공학과 재학(박사 과정)
관심분야 : 컴퓨터 구조, 임베디드 시스템, 무선 통신, 센서 네트워크 등

안 종 석
e-mail : jahn@dgu.edu
1983년 서울대학교 전자공학과(학사)
1985년 KAIST 전기 및 전자공학과(공학석사)
1985년~1989년 삼성전자 주임연구원
1995년 USC(University of Southern California) Electrical Engineering Department -Systems(공학박사)
1995년~1996년 삼성전자 선임연구원
2001년~2002년 USC/ISI 교환연구원
1996년~현 재 동국대학교 컴퓨터공학과 교수
관심분야 : 네트워크 시뮬레이션, 무선 통신, 라우팅 알고리즘, 센서 네트워크 등

이 강 우
e-mail : klee@dgu.edu
1985년 연세대학교 전자공학과(학사)
1991년 USC(University of Southern California) 컴퓨터공학과(공학석사)
1997년 USC(University of Southern California) Electrical Engineering Department -Systems(공학박사)
1998년~현 재 동국대학교 정보통신공학과 부교수
관심분야 : 컴퓨터 구조, 임베디드 시스템, 센서 네트워크 등