무선 네트워크에서 채널 효율성을 높이기 위한 멀티채널 MAC 프로토콜

정회원 김 영 경*, 유 상 조*

Multi-channel MAC Protocol for Improving Channel Efficiency in Wireless Networks

Young-Kyoung Kim*, Sang-Jo Yoo* *Regular Members*

요 약

본 논문은 무선 네트워크 환경에서 채널 효율성을 높이기 위하여 두 개의 인터페이스를 사용하는 새로운 멀티채널 MAC 프로토콜을 제안하였다. 멀티채널 환경에서 기존에 제안된 대부분의 연구들은 컨트롤 메시지를 교환하기 위하여 공통 컨트롤 채널을 사용하였는데, 이것은 데이터 채널이 증가할 때 공통 컨트롤 채널에서 병목 현상을 야기 시킨다. 본 논문에서는 제안하는 멀티채널 MAC 프로토콜은 송수신용 채널을 분리하여 데이터와 컨트롤 패킷을 동시에 송수신하여 전체 네트워크 처리량을 높이고, 공통 컨트롤 채널이 없기 때문에 병목현상 문제가 발생하지 않는다. 또한 TDMA 방식을 사용하여 데이터 패킷과 컨트롤 패킷의 충돌을 피할 수 있고, 멀티채널환경에서 송수신 인터페이스가 다를 경우 발생하는 CTS 패킷이나 ACK 패킷의 충돌 가능성을 줄였다. 모의실험결과는 제안된 멀티채널 MAC 프로토콜이 기존의 방법과 비교해서 전체 네트워크 처리량과 채널 효율성을 크게 향상시키는 것을 보여준다.

Key Words : Multi-channel MAC, Channel Efficiency, Throughput

ABSTRACT

In this paper, we propose a new multi-channel MAC protocol to improve the channel efficiency by using two interfaces. Most of previous researches that have considered multi-channel wireless network environments use a common control channel to exchange control signals and they have a bottle neck problem at common control channel as increasing the number of data channels. In the proposed MAC protocol, we separate receiving and transmitting channels so that sending and receiving data and control packets at the same time is possible. It increases the total network throughput. Since there is no common control channel, the network does not suffer from the bottle neck problem. By applying a TDMA scheme, we can avoid packet collisions between data packets and control packets and reduce the possibility of CTS or ACK packet collisions. Simulation results show that the proposed multi-channel MAC protocol improves the total network throughput and channel efficiency compared with the existing method.

Ⅰ. 서 론

멀티 주파수 채널 (multi frequency channel)의 사용은 무선 네트워크의 성능 향상을 위해 무한한 가능성을 제공한다. IEEE 802.11[1]과 같은 표준이 멀티 직교 채널 (multiple orthogonal channel)을 제공

※ 본 연구는 2007학년도 학술진흥재단의 지원에 의하여 연구됨 (KRF-2005-202-D00321)
* 인하대학교 정보통신 대학원 멀티미디어 통신망 연구실 (miniwalker@naver.com)
논문번호 : KICS2008-01-007, 접수일자 : 2008년 1월 4일, 최종논문접수일자 : 2008년 5월 7일

하고 있다는 점을 상기해볼 때 이 가능성은 현재 사용되고 있는 표준에서 이미 인식되어 왔다는 것을 알 수 있다. IEEE 802.11b는 기본적으로 2.4GHz의 주파수를 사용하고 11Mbps의 대역폭을 갖는 3개의 무선 채널을 제공하고, IEEE 802.11a의 경우는 5GHz 주파수를 사용하여 54Mbps의 대역폭의 4개 혹은 8개의 무선 채널을 제공한다. 멀티 주파수 채널의 사용은 인접한 이웃 노드 (neighbor node)들의 통신이 다른 채널에서 이루어지는 한, 이웃 노드와 물리적으로 가까이 있어도 충돌에 자유로운 전송을 가능하게 하기 때문에, 간섭 없는 다수의 전송을 동시에 할 수 있으므로 한 채널을 사용하는 것보다 높은 네트워크 처리량 (throughput)을 얻을 수 있다.

본 논문은 채널 효율성 (channel efficiency)과 네트워크 성능 향상을 위해 멀티채널을 이용하는 MAC 프로토콜을 제안한다. 본 논문에서 제안하는 MAC 프로토콜은 노드 당 최소 2개의 인터페이스를 가지고 있다고 가정하고, 하나의 인터페이스에는 데이터의 수신을 위한 수신용 채널 (fixed channel)을, 다른 인터페이스에는 다른 노드에게 데이터를 전송하기 위한 송신용 채널 (switchable channel)을 할당하여 데이터 패킷 (packet)의 송신과 수신을 동시에 가능하게 하여 전체 네트워크 처리량을 향상시켰고, 기존의 멀티채널 MAC 프로토콜[2,3]에서 채널수가 많아질 경우 발생하였던 병목현상(bottle neck problem)을 해결하였다. 또한 TDMA 방법을 사용하여 본 논문과 동일하게 각 노드마다 수신용 채널을 할당하는[4] HMCP (Hybrid Multichannel Protocol)에서 발생할 수 있는 DATA 패킷과 컨트롤 패킷 (control packet)의 충돌문제를 해결하였고, 멀티채널을 사용하고 송수신 인터페이스가 다를 경우 발생하는 CTS 패킷이나 ACK 패킷의 충돌을 줄이는 방법을 제시하였다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 인터페이스 할당 전략에 대하여 간략히 살펴보고, 그 중 혼합할당전략을 사용한 HMCP에 대하여 설명하고 문제점을 제시한다. 제 Ⅲ장에서는 새롭게 제안하는 무선 멀티채널 MAC 프로토콜의 전체 프레임과 데이터 교환 절차를 설명하고, 제 Ⅳ장에서는 모의실험 결과를 통하여 제안된 MAC 프로토콜의 성능을 평가한다. 마지막으로 제 Ⅴ장에서는 본 연구의 결론을 기술한다.

Ⅱ. 관련 연구

이 장에서는 무선 네트워크에서 멀티채널을 이용하기 위하여 인터페이스에 채널을 할당하는 여러 가지 전략에 대해 설명한다.

2.1 인터페이스 할당 전략

인터페이스 할당 전략은 인터페이스에 채널을 얼마 동안 할당하느냐에 따라 정적 할당 (static assignment)과 동적 할당 (dynamic assignment)으로 나타낼 수 있고, 두 방법을 혼합한 혼합할당 (hybrid assignment)으로 분류할 수 있다.

먼저, 정적 할당전략은 각 인터페이스를 채널에 영구적으로 혹은 스위칭 시간 (switching time)에 비해 상대적으로 긴 시간 동안 할당하는 방법으로[5,6] 인터페이스의 스위칭 지연이 클 때 적절하다. 만약 이용 가능한 인터페이스가 이용 가능한 채널수와 같다면 인터페이스 할당은 당연히 정적 할당이 된다. 정적 할당은 (긴 시간의 인터벌이 지나 인터페이스에 다시 채널을 할당할 때를 제외하고) 데이터 통신을 위해 노드사이에 특별한 조정 (coordination)을 요구하지 않는다. 정적 할당에서는 인터페이스 중 하나로 채널을 공유하는 노드들과는 서로 직접적으로 통신할 수 있지만, 반면에 다른 노드들과는 직접적인 통신이 불가능하다. 그래서 정적 채널 할당의 효과는 네트워크 토폴로지 (topology)를 컨트롤하기 위해 어떤 노드들을 서로 직접적으로 통신이 가능하게 할지를 결정하는 가에 따라 달라진다.

동적 할당전략은 인터페이스가 빈번하게 한 채널에서 다른 채널로 스위치 하는 것을 허용하여 인터페이스에게 어떤 채널이든지 할당될 수 있게 하는 방법이다. 이 방법에서 서로 통신이 필요한 두 노드는 어떤 특정 시간에 공통 채널에 있기 위해 조정 메커니즘 (coordination mechanism)을 필요로 한다. 예를 들어 조정 메커니즘은 모든 노드에게 공통 "rendezvous" 채널을 주기적으로 방문하도록 요구 할 수도 있다[7]. 아니면 의사 난수 수열 (pseudo-random sequence)[8]의 사용 등 다른 메커니즘이 필요할 수도 있다. 동적 할당은 인터페이스를 어떤 채널로든 스위치 할 수 있기 때문에 적은 인터페이스를 가지고도 많은 채널을 커버할 수 있는 이점이 있다. 동적 할당은 네트워크에서 노드들 사이에 어떤 채널로 언제 인터페이스를 스위치 하는가의 결정을 조정하는 것이 주요 과제이다.

마지막으로, 혼합 할당전략은 어떤 인터페이스들에 대해서는 정적 할당을 적용하고 다른 인터페이스들에 대해서는 동적 할당을 적용하여서 두 할당 전략을 결합하여 사용하는 방법이다. 혼합 할당 전략은 정적 할당에 적용되는 인터페이스가 공통채널

에 할당되는지 아니면 다양한 채널에 할당되는 지에 따라 자세히 분류될 수 있다. MAC 계층에서 공통채널을 가지고 혼합할당을 하는 예로, [2]에서는 각 노드들의 한 인터페이스에는 공통 "컨트롤"채널을 정적으로 할당하고 그 밖에 다른 인터페이스에는 다른 "데이터"채널 사이에서 자유롭게 스위치 할 수 있도록 하였다. 혼합 할당전략은 동정할당의 유연성을 유지하면서 정적 할당에 의해 지원되는 단순화된 조정 알고리즘을 고려하기 때문에 매력적이라고 할 수 있다.

2.2 HMCP (Hybrid Multi-channel Protocol)

이 장에서는 혼합 인터페이스 할당 전략 중 하나인 HMCP에 대해 설명한다. HMCP (Hybrid Multi-channel Protocol)[4]에서 각 노드는 고정된 인터페이스(fixed interface)와 전환가능 인터페이스(switchable interface)두 종류의 인터페이스를 가진다고 가정한다. 고정된 인터페이스는 이웃 노드로부터 패킷을 수신하기 위해 특정한 채널에 할당되는 용도로 사용되는 인터페이스이다. 그리고 전환가능 인터페이스는 이웃 노드에게 보낼 데이터가 있을 때 자신의 인터페이스를 데이터 수신자의 고정된 인터페이스의 채널과 동일한 채널로 할당하기 위해 사용되는 인터페이스이다. 고정된 인터페이스의 수와 전환가능 인터페이스의 수는 노드에서 이용 가능한 인터페이스의 수에 따라 결정된다. 혼란을 피하기 위해 이 후부터는 고정된 인터페이스를 수신용 인터페이스(fixed interface)라고 하고, 전환가능 인터페이스를 송신용 인터페이스(switchable interface)라고 하겠다.

HMCP에서 인터페이스에 대한 채널 할당은, 각 노드에서 이용 가능한 인터페이스의 수와 채널의 수를 M이라고 했을 때, 각 노드의 수신용 인터페이스 K개($K<M$)에게 K개의 수신용 채널 (fixed channel)을 할당하고 송신용 인터페이스 M-K개에게 M-K개 이상의 채널을 할당하도록 하였다. 만약 총 인터페이스의 수 M=2이고 수신용 인터페이스 수 K=1이라고 했을 때, 수신용 인터페이스가 채널 i로 할당이 된 경우는 그림 1과 같이 나타낼 수 있다. 그림 1에서처럼 송신용 인터페이스는 i를 제외한 모든 채널에 자유롭게 할당될 수 있다.

그림 2는 노드 A가 노드 B를 지나 노드 C에게 보낼 패킷이 있다고 가정했을 때, 이 프로토콜의 동작을 설명한다. 노드 A와 B와 C의 수신용 인터페이스는 각각 채널 1, 2, 3에 할당되어 있고, 송신용 인터페이스는 초기에 채널 3, 1, 2가 할당되어 있다.

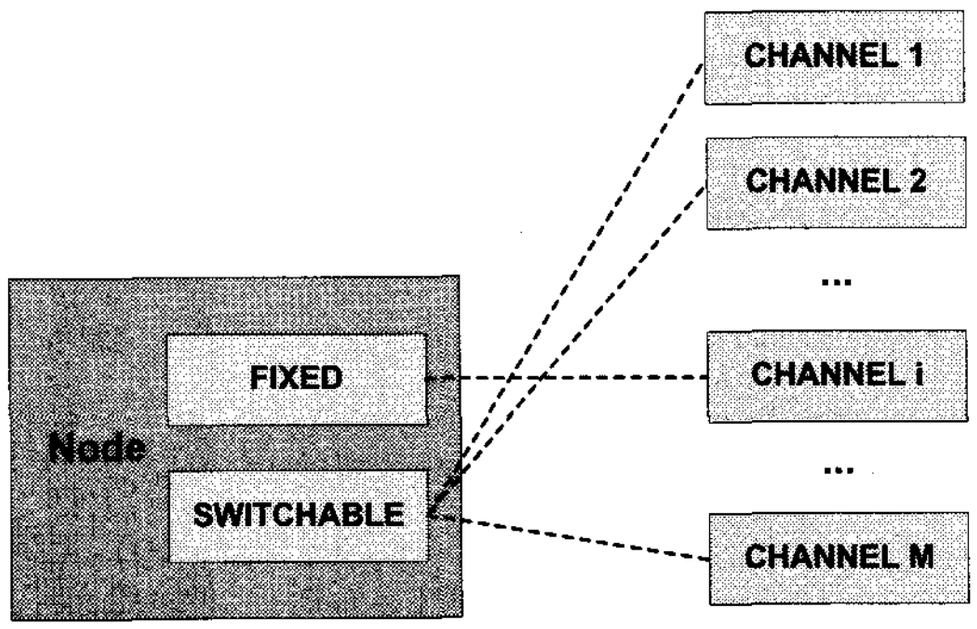


그림 1. 인터페이스 채널 할당 예

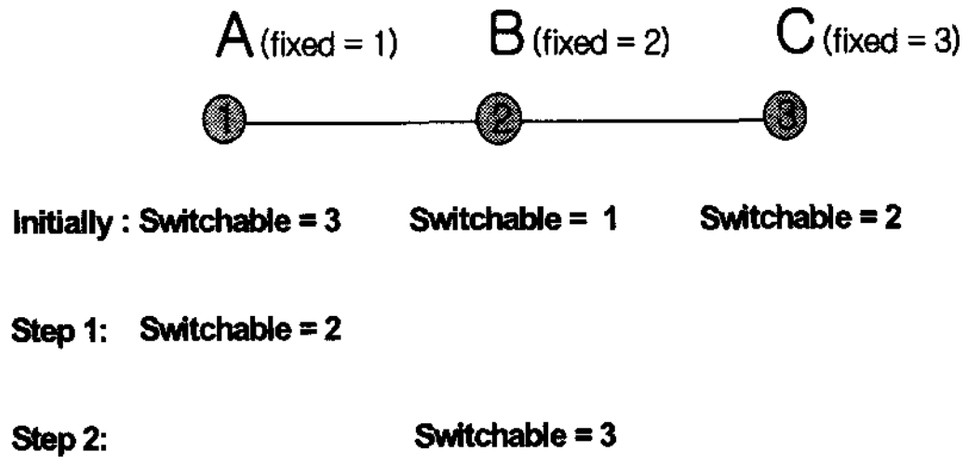


그림 2. HMCP 프로토콜 동작

우선 노드 A는 노드 B에게 패킷을 보내기 위해 패킷을 보내기 전에 노드 자신의 송신용 인터페이스를 채널 3에서 노드 B의 수신용 채널인 채널 2로 변경한다. 다음으로, 노드 B는 패킷을 노드 C에게 보내기 위해 자신의 송신용 인터페이스를 채널 1에서 노드 C의 수신용 채널인 채널 3으로 변경하고 패킷을 전송한다.

HMCP에서 각 노드들은 자신의 수신용 채널을 이웃 노드 (neighbor node)에게 알려야 한다. 그러기 위해 각 노드들은 주기적으로 자신과 이웃의 수신용 채널정보를 포함한 HELLO 패킷을 이웃노드에게 브로드캐스트 (broadcast)하고, HELLO 패킷을 수신한 각 노드는 NeighborTable을 두어 이웃 노드의 수신용 채널 정보를 관리한다. 이 때 HMCP에서의 브로드캐스트는, 노드들이 자신의 수신용 인터페이스에게 할당된 채널 이외의 다른 채널의 패킷은 들을 수 없기 때문에, 브로드캐스트 패킷을 각 채널로 복사해서 모든 채널로 전송하는 로컬 브로드캐스트(local broadcast) 방법으로 구현되었다.

HMCP에서 노드들은 수신용 인터페이스에 서로 다른 수신용 채널을 할당하기 때문에 이웃 노드의 Control 패킷을 수신하지 못한 노드에서 DATA 패킷과 RTS 패킷이 충돌하는 문제가 발생할 수 있다.

예를 들어, 그림 3에서 노드 A는 노드 B에게 데

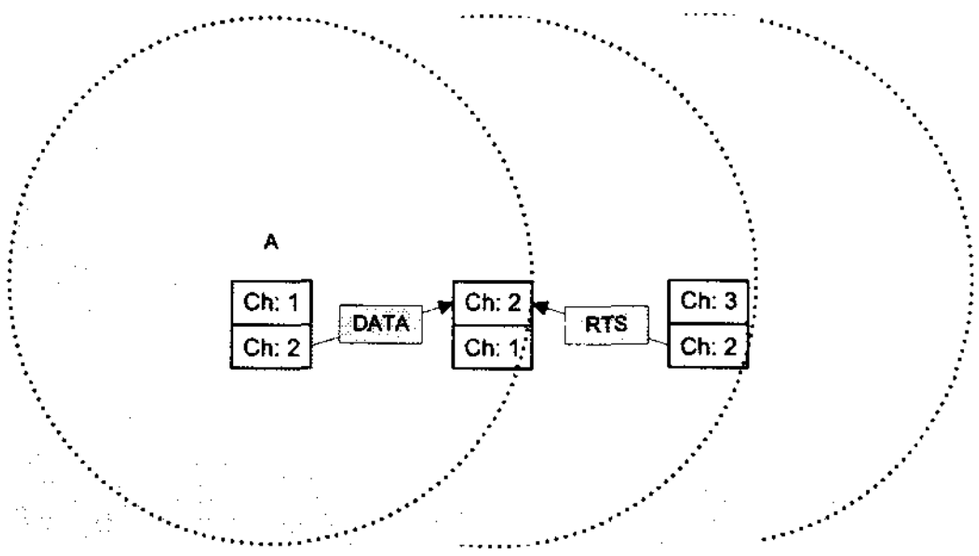


그림 3. 수신 노드에서 패킷 충돌 예

이터 전송을 요청하기 위해 노드 B의 수신용 채널인 채널 2로 RTS 패킷을 보낸다. 그리고 노드 *A*로부터 RTS 패킷을 받은 노드 *B*는 노드 *A*의 수신용 채널인 채널 1로 CTS 패킷을 보낸다. 이 때, 노드 *B*가 CTS 패킷에 NAV를 넣어서 얼마나 채널 2를 점유할 것 인가를 알려주지만, 노드 *C*의 수신용 인터페이스는 채널 1이 아닌 채널 3에 할당되어 있으므로 노드 *C*는 노드 *B*가 보내는 CTS 패킷을 수신하지 못한다. 그로 인해 그림 3에서처럼 노드 *B*의 통신을 모르고 있던 노드 *C*가 채널 2로 노드 *B*에게 RTS 패킷을 보낼 때, 노드 *A*로부터 DATA 패킷을 수신 중이던 노드 B에서 충돌이 발생하게 된다.

Ⅲ. 멀티채널 MAC 프로토콜

본 논문에서 제안하는 멀티채널 MAC 프로토콜은 혼합 인터페이스 할당 전략을 사용하는 프로토콜 중 하나로써 각 노드가 최소 두 개의 인터페이스를 가진다고 가정한다. 모든 노드는 이용 가능한 인터페이스를 두 그룹으로 나눈다. 첫 번째 그룹의 인터페이스는 수신용 인터페이스로써 이웃 노드로부터 패킷을 수신하기 위하여 일정시간 동안 "수신용 채널 (fixed channel)"이라고 부르는 특정 한 채널에 고정하고, 두 번째 그룹의 인터페이스는 송신용 인터페이스로써 각 노드에서 수신용 채널로 사용되는 채널 이외의 다른 채널로 인터페이스의 채널을 자유롭게 전환 (switch) 할 수 있도록 한다. 송신용 인터페이스는 노드가 이웃 노드와 패킷을 교환할 때 이웃 노드의 수신용 채널로 패킷을 보내기 위해 이웃 노드의 수신용 채널로 자신의 송신용 인터페이스의 채널을 전환하기 위해 사용된다. 이 후에는 설명을 간단히 하기 위해서 각 노드가 수신용 인터페이스 한 개와 송신용 인터페이스를 한 개, 총 두 개의 인터페이스를 가지고 있다고 가정하였다.

3.1 멀티채널 MAC 프로토콜 프레임 구조

본 논문에서 제안하는 프로토콜은 TDMA 방식으로 동작한다. 제안 하는 프레임의 구조에서 슈퍼프레임은 그림 4와 같이 한번의 HELLO WINDOW와 한 번 이상의 COMMUNICATION WINDOW로 구성된다.

HELLO WINDOW는 WINDOW의 길이에 따라 그림 5-(a)와 같이 HELLO WINDOW LONG과 그림 5-(b)와 같이 HELLO WINDOW SHORT 두 종류로 나누어진다. HELLO WINDOW LONG은 k개의 HS (Hello Slot)로 구성되고, 각 HS의 길이는 t개의 백오프 미니슬롯 (backoff mini-slot)과 HELLO 패킷 크기를 합한 정도이다. HELLO WINDOW SHORT은 $\frac{k}{2}$개의 HS로 구성된다.

COMMUNICATION WINDOW기간은 그림 6과 같이 RTS WINDOW, CTS WINDOW, DATA WINDOW, ACK WINDOW로 구성된다. RTS WINDOW는 HELLO WINDOW와 유사한 구조를 가지며 x개의 RS (RTS Slot)로 구성되고, 각 RS의 길이는 t개의 백오프 미니슬롯 (backoff mini-slot)과 RTS 패킷 크기를 합한 정도이다. CTS WINDOW와 ACK WINDOW는 y개의 CS (CTS Slot) 또는 AS (ACK Slot)로 구성되며 하나의 CS 또는 AS의 크기는 CTS 패킷 또는 ACK 패킷의 크기와 같다. DATA WINDOW는 크기가 고정되어 있으며 크기는 네트워크의 최대 DATA 패킷의 크기와 같다. 그리고 전체 프레임 구조는 한번의 HELLO WINDOW LONG을 가진 슈퍼프레임과 한번 이상의 HELLO WINDOW SHORT을 가진 슈퍼프레임이 반복된다.

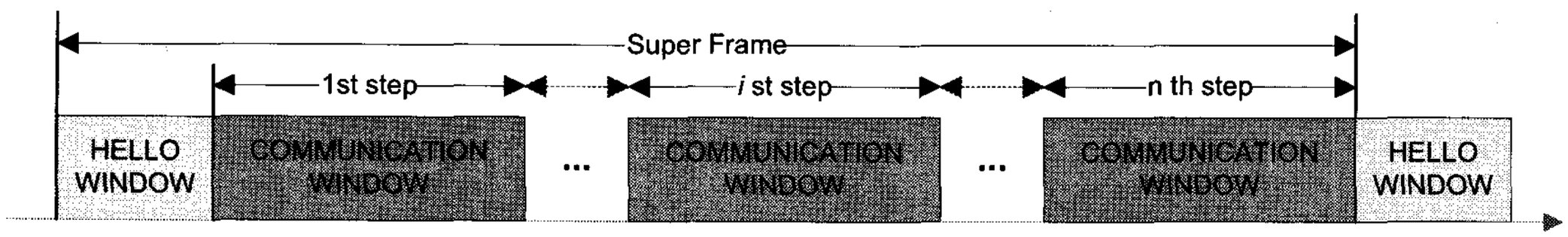
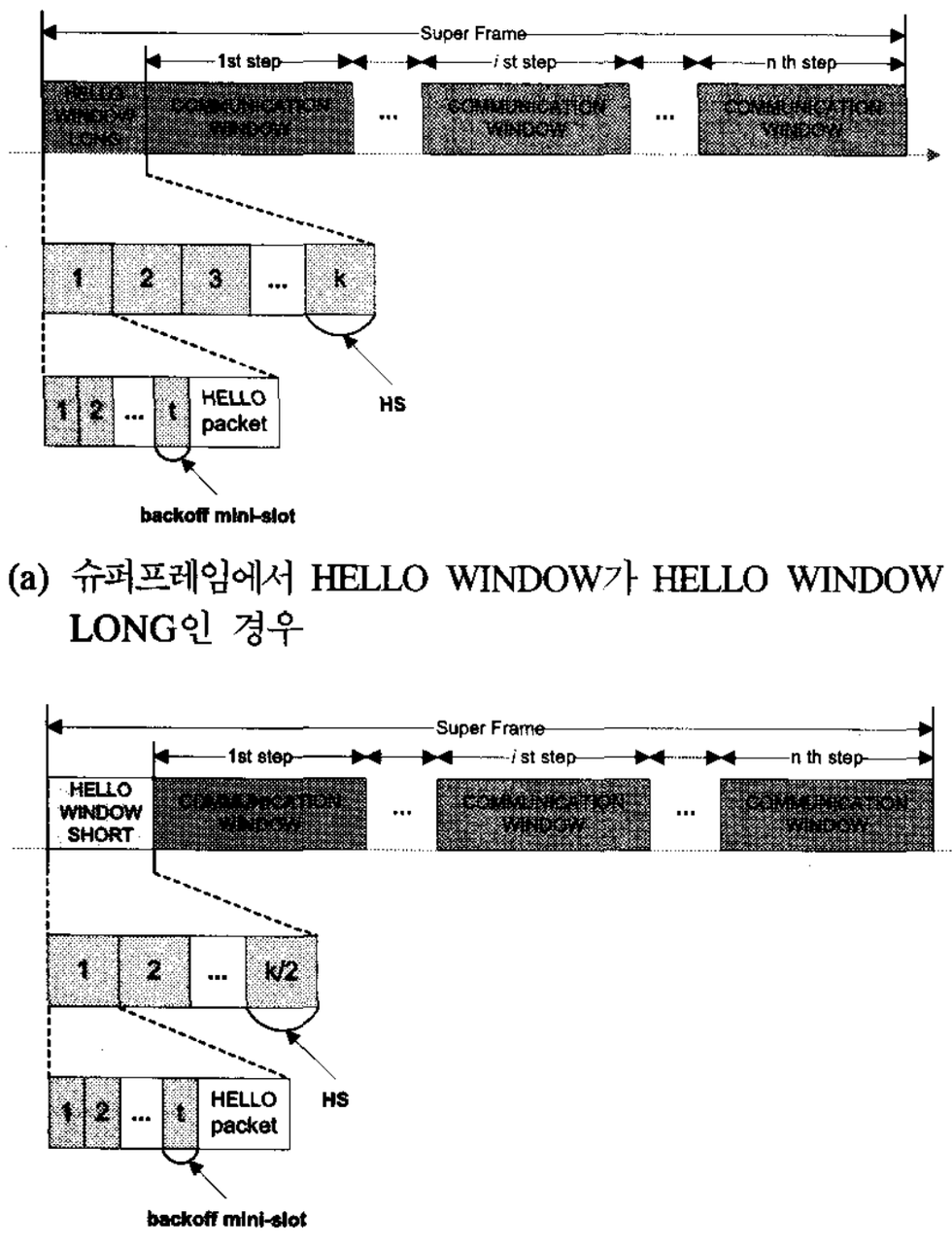


그림 4. 슈퍼프레임 구조



(a) 슈퍼프레임에서 HELLO WINDOW가 HELLO WINDOW LONG인 경우

(b) 슈퍼프레임에서 HELLO WINDOW가 HELLO WINDOW SHORT인 경우

그림 5. 슈퍼프레임에서 HELLO WINDOW 구조

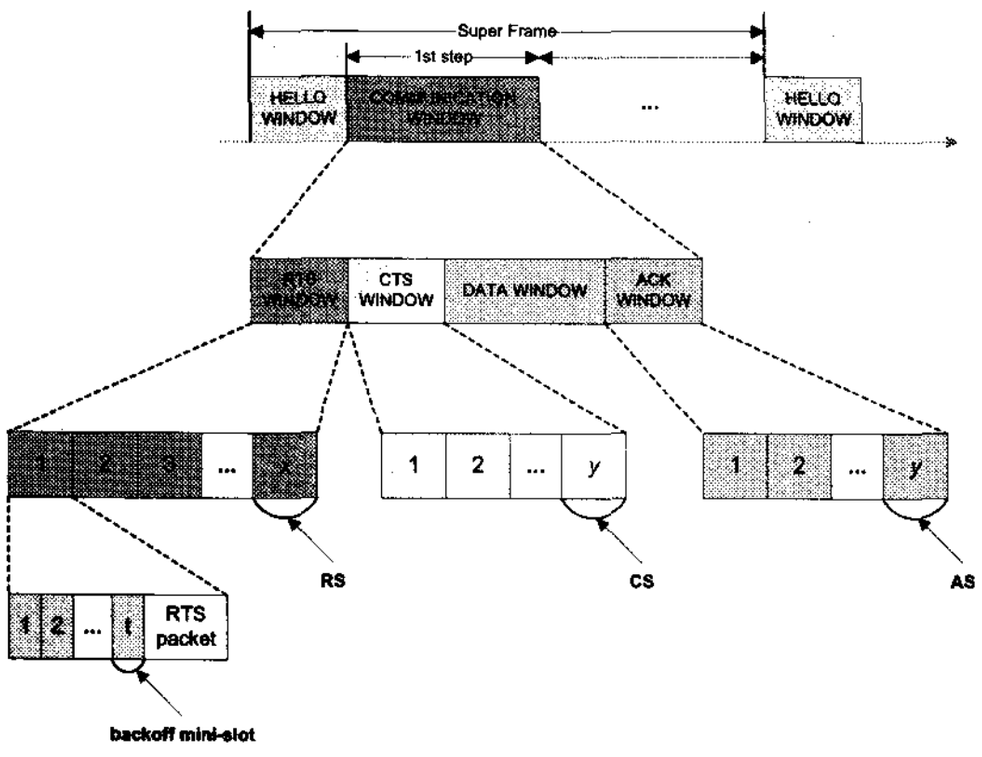


그림 6. COMMUNICATION WINDOW 구조

3.2 HELLO 패킷 브로드캐스트

본 논문에서 노드들은 데이터를 교환하기 전에 이웃 노드들의 수신용 채널정보를 알기 위해 주기적으로 HELLO WINDOW 기간 동안 자신과 이웃의 수신용 채널 정보를 담은 HELLO 패킷을 교환한다. HELLO WINDOW는 WINDOW의 길이에 따라 HELLO WINDOW LONG과 HELLO WINDOW SHORT 두 종류로 나누어지는데, HELLO WINDOW가 HELLO WINDOW LONG인 경우는 모든 노드가 HELLO 패킷 브로드캐스트를 시도하고, HELLO WINDOW SHORT인 경우는 노드 자신의 수신용 채널이 변경되었거나 이웃 노드의 정보를 기록한 NeighborTable에서 이웃 노드들의 수신용 채널 정보가 변경된 경우만 HELLO 패킷을 브로드캐스트 한다. 이는 노드의 평균 이동성이 낮은 경우, 시간이 지날수록 각 노드가 선택한 수신용 채널이 네트워크 전체에 고르게 분포되어 각 노드의 수신용 채널을 변경할 필요가 없어지기 때문이다. 그래서 HELLO WINDOW SHORT를 이용하면 자신의 수신용 채널이 변경된 노드나 NeighborTable에서 이웃 노드들의 수신용 채널 정보가 변경된 노드만 HELLO 패킷을 브로드캐스트하기 때문에 불필요한 전송을 줄일 뿐만 아니라 HELLO WINDOW LONG을 이용하였을 경우 발생할 수 있는 HELLO 패킷의 충돌도 줄일 수 있다. 그래서 본 논문은 HELLO WINDOW LONG 기간의 반복 주기를 노드의 평균 이동성에 따라 다르게 설정할 수 있도록 제안한다.

HELLO 패킷의 교환 절차는 다음과 같은 순서로 이루어진다.

1) Hello Window 기간 동안 모든 노드는 송수신용 인터페이스의 채널을 공통채널로 고정한다.
2) HELLO 패킷을 교환하기 위해, 우선 각 노드는 자신의 HELLO 패킷을 보낼 슬롯을 랜덤하게 선택한다. 슬롯은 HELLO WINDOW 기간의 길이가 HELLO WINDOW LONG인 경우 1에서 k사이에서 선택하고, HELLO WINDOW SHORT인 경우, 1에서 $\frac{k}{2}$사이에서 선택할 수 있다.
3) 다음으로 각 노드는 선택한 슬롯을 이용하여 HELLO 패킷을 전송하기 위해 0에서 t사이의 백오프 미니 슬롯을 랜덤하게 선택하여 미니 슬롯 백오프를 수행한고 채널을 센싱한다.
4) 만약 미니슬롯 백오프 동안 다른 노드의 HELLO 패킷 전송이 없으면 노드는 자신의 HELLO패킷을 전송하고, 다른 노드의 HELLO 패킷 전송이 있는 경우 노드는 자신의 HELLO 패킷의 전송을 포기한다.
5) 4)에서 만약 자신과 같은 미니슬롯을 선택한 이웃노드가 있을 경우, 노드는 센싱 중에 아무것도 감지하지 못했기 때문에 자신의 HELLO

패킷을 전송하지만, 자신과 같은 미니슬롯을 선택한 이웃노드와 동시에 HELLO 패킷을 전송하게 되어 두 노드의 패킷은 충돌을 일으킬 수 있다.

6) 이웃 노드로부터 Hello 패킷을 받은 노드는 자신의 NeighborTable에 이웃 노드의 수신용 채널 정보를 기록하고 다음 HELLO 패킷을 교환하기 전까지 NeighborTable에 기록된 정보를 이용하여 이웃노드와 데이터를 주고받는다.

예를 들면, 그림 7에서 HELLO 패킷을 전송하기 위해 1번 HELLO슬롯을 선택한 노드 *A*가 선택한 2개의 미니슬롯을 백오프하는 동안 다른 경쟁 노드의 전송을 감지하지 못한 경우 노드 *A*는 자신의 HELLO 패킷을 전송하기 위해 첫 번째 슬롯을 이용할 수 있게 된다. 반면에, 1번 HELLO슬롯과 3개의 미니슬롯 백오프를 선택한 노드 *D*는 미니슬롯 백오프를 하는 동안 노드 *A*의 전송을 감지하여 자신의 HELLO 패킷 전송을 포기하게 된다. 그리고 2번 슬롯과 4개의 미니슬롯 백오프를 선택한 노드 *B*와 노드 *C*는 4개의 미니슬롯 백오프 동안 다른 노드의 전송이 없었기 때문에 HELLO 패킷을 전송하지만, 두 노드가 동시에 HELLO 패킷을 전송하게 되어 충돌이 발생하게 된다.

노드는 NeighborTable에 기록된 정보를 이용하여 주기적으로 자신과 같은 수신용 채널을 사용하는 이웃들의 수를 측정하는데, 만약에 측정된 이웃 노드들의 수가 평균 이상이면 그 노드는 자신의 수신용 채널을 이웃 노드에 의해 가장 적게 사용되는 수신용 채널로 변경한다. 그리고 이 정보를 Hello Window 기간 동안 HELLO 패킷에 포함하여 전송한다. 이는 이용 가능한 채널이 네트워크 전체에 분산되도록 해준다.

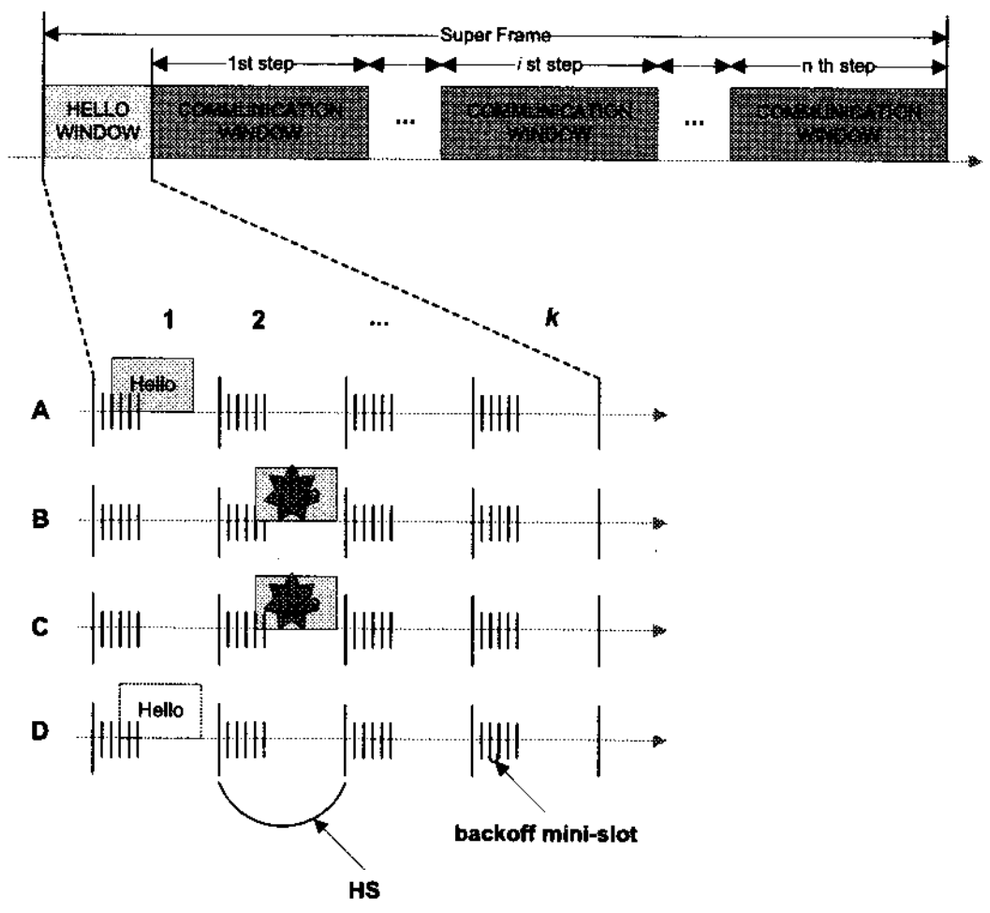


그림 7. HELLO 패킷의 교환 절차 예

HELLO 패킷은 기본적으로 이웃노드의 노드 ID, 수신용 채널, 홉 카운트 (hop count) 정보를 포함하여 브로드캐스트 되고, HELLO 패킷을 이웃노드로부터 수신한 노드는 자신의 NeighborTable에 이웃노드의 채널 정보를 기록한다. 그리고 일정 시간 동안 사용되지 않은 채널 정보는 NeighborTable에서 제거된다.

3.3 DATA 전달 절차

HELLO WINDOW 기간이 끝나면 데이터 전송을 요청하기 위해 전송할 데이터를 가진 노드는 수신 노드에게 RTS (Ready To Send) 패킷을 전송한다. 본 논문에서는 이를 위해 RTS WINDOW기간을 가지며, RTS WINDOW는 HELLO WINDOW와 비슷한 구조를 가지며 t개의 백오프 미니슬롯을 포함하는 *x*개의 RS (RTS slot)로 구성되어 있다.

먼저 전송할 데이터가 있는 노드는 자신의 송신용 인터페이스의 채널을 데이터를 수신하게 될 노드의 수신용 채널로 변경한다. 그리고 각 노드는 RTS 패킷을 송신하기 위해 1에서 *x*사이의 RTS 슬롯을 선택한다. 선택한 슬롯을 이용하여 RTS 패킷을 보내기 위해 각 노드는 0에서 *t*사이의 미니슬롯을 선택하여 미니슬롯 백오프를 실행하고 채널을 센싱한다. 만약 미니슬롯 백오프 동안에 자신이 선택한 RTS 슬롯에 RTS 패킷을 전송하는 노드가 없으면 그 슬롯에 자신의 RTS 패킷을 전송한다. 이때, 노드의 송신용 채널과 수신용 채널이 다르기 때문에, 노드는 자신의 RTS 패킷을 송신하면서 동시에 또 다른 이웃 노드로 부터 RTS 패킷을 수신할 수 있다. 예를 들어, 그림 8에서 노드 *C*에게 전송할 데이터가 있는 노드 B는 자신의 송신용 인터페이스를 노드 *C*의 수신용 채널인 채널 1로 고정하고, 두 번째 RTS 슬롯 타임에 RTS 패킷을 보내기 위해 자신의 미니슬롯 백오프를 실행한다. 노드 *B*는 미니슬롯 백오프 동안 노드 *C*로 RTS 패킷을 전송하는 다른 노드가 없기 때문에 자신의 RTS 패킷을 노드 *C*에게 전송한다. 그리고 동시에 노드 *B*는 RTS WINDOW 기간 동안 노드 *A*와 노드 *C*로부터 RTS 패킷을 받는다.

RTS 패킷을 받은 노드는 수신한 RTS 패킷 중에서 첫 번째 RTS 패킷을 송신한 노드에게 DATA

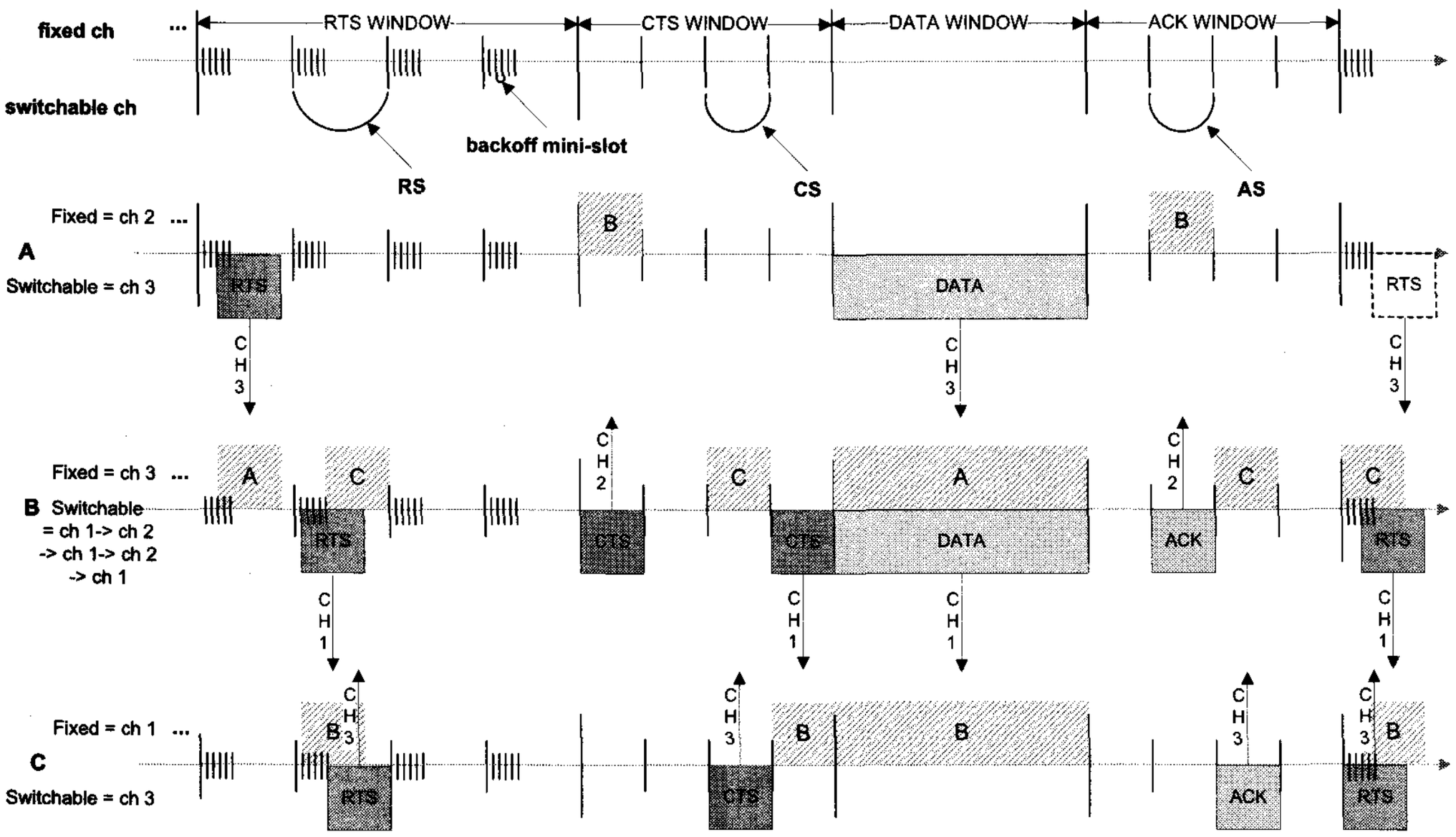


그림 8. RTS, CTS 패킷 전송 예

패킷을 요청하기 위해 CTS (Clear to send) 패킷을 보낸다. CTS 패킷은 y개의 CS (CTS slot) 중 랜덤하게 선택한 한 CTS 슬롯을 이용하여 전송된다. CTS 패킷 전송은 다음절에서 자세하게 설명한다.

노드가 여러 개의 RTS 패킷을 수신한 경우 노드는 자신에게 두 번째 RTS 패킷을 송신한 노드에게 다음 RTS WINDOW에서 RTS 패킷을 우선 전송할 수 있는 권한을 주는 CTS 패킷을 보낸다. 이 CTS 패킷을 받은 노드는 다음 RTS WINDOW가 시작할 때 첫 번째 RTS 슬롯에서 미니슬롯 백오프를 하지 않고 RTS 패킷을 전송할 수 있다. 예를 들면 그림 8에서 노드 *B*는 노드 *C*에게 RTS 패킷을 전송하는 동안 노드 *A*와 노드 *C*로 부터 RTS 패킷을 수신하는 데, 두 개의 RTS 패킷을 수신한 노드 *B*는 자신에게 첫 번째 RTS 패킷을 송신한 노드 *A*에게는 채널 2로 데이터를 요청하기 위한 CTS 패킷을 보내고, 두 번째 RTS 패킷을 송신한 노드 *C*에게는 채널 1로 다음 RTS WINDOW에서 RTS 패킷을 우선 전송할 수 있는 권한을 주는 CTS 패킷을 보낸다. 그리고 이 CTS 패킷을 받은 노드 *C*는 그림 8에서와 같이 다음 RTS WINDOW에서 첫번째 RTS 슬롯에서 미니슬롯 백오프 없이 RTS 패킷을 노드 *B*에게 송신할 수 있게 된다.

두 개 이상의 노드가 한 노드에게 같은 RTS 슬롯과 미니슬롯을 이용하여 데이터를 전송한 경우 RTS 패킷은 충돌할 수 있다. 그림 9에서 노드 *A*와

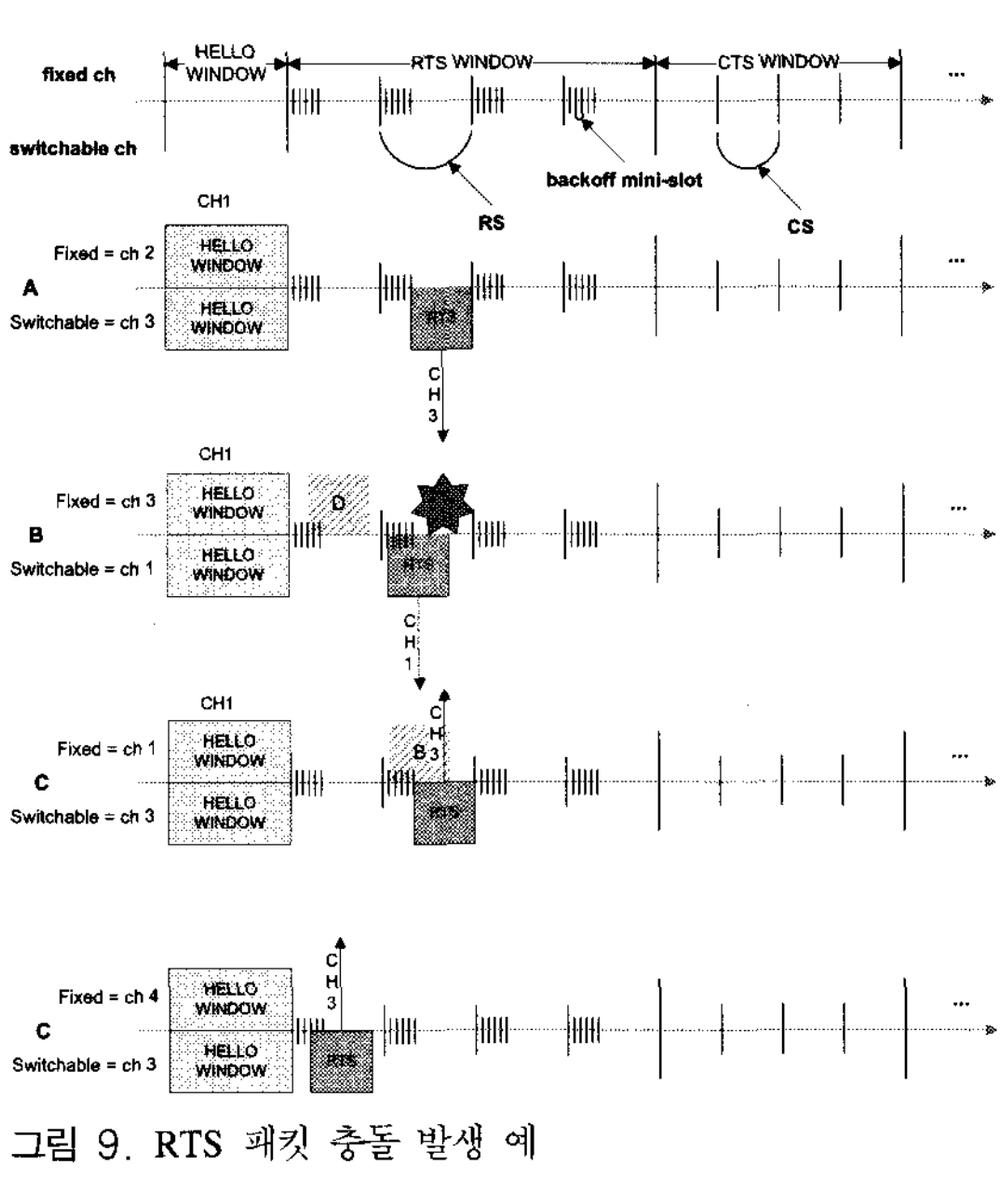


그림 9. RTS 패킷 충돌 발생 예

노드 *C*는 노드 *B*에게 RTS 패킷을 전송하기 위해 송신용 인터페이스를 채널 3으로 고정하고 두 번째 RTS 슬롯을 선택에서 동일하게 5개의 미니슬롯 백오프를 실행한다. 미니슬롯 백오프 동안 다른 노드의 RTS 패킷을 센싱하지 못한 노드 *A*와 노드 *C*는 미니슬롯 백오프 이후 동시에 RTS 패킷을 노드 *B*에게 전송하고 두 RTS 패킷은 노드 *B*에서 충돌한다. 그래서 RTS 패킷을 송신하였지만 CTS WINDOW에

서 노드 *B*로 부터 아무것도 수신하지 못한 노드 *A*와 *C*는 노드 *B*와의 전송을 포기하고 다음 RTS WINDOW에서 다시 전송을 시도한다.

3.4 멀티 채널을 사용하고 송수신 인터페이스가 다른 경우 발생하는 문제

멀티채널 환경에서 한 노드의 이웃노드 수가 이용 가능한 채널 수보다 많은 경우 그 노드는 자신과 같은 채널을 사용하는 이웃 노드를 가질 수 있다. 이로 인해 멀티채널 환경에서 송수신 인터페이스가 다를 경우 CTS 패킷이나 ACK 패킷이 충돌하는 새로운 문제가 발생할 수 있다.

예를 들어 그림 10에서와 같이 두 홉 거리에 있는 노드 *A*와 노드 *C*가 수신용 채널로 같은 채널을 사용할 경우 충돌이 발생할 수 있다. 그림에서 노드 *A*와 노드 *C*는 수신용 채널로 채널 1을 사용하고 있고, 노드 *B*와 노드 *D*는 수신용 채널로 각각 채널 2와 채널 4를 사용하고 있다고 가정한다. 여기서 노드 *B*에게 전송할 데이터가 있는 노드 *A*는 노드 *B*에게 채널 2로 RTS 패킷을 송신하여 전송을 요청하고, 노드 *D*에게 전송할 데이터가 있는 노드 *C*는 노드 *D*의 수신용 채널인 채널 4로 RTS 패킷을 보내 전송을 요청한다. 이때, 노드 *B*는 수신용 인터페이스가 채널 2로 고정되어 있으므로 노드 *C*가 채널 4로 보내는 RTS 패킷을 수신하지 못하게 된다. 그래서 그림 10에서와 같이 노드 *B*는 송신용 인터페이스를 노드 *A*의 수신용 채널인 채널 1로 스위칭하고 CTS 패킷을 노드 A에게 전송하게 되고, 이로 인해 노드 *D*로부터 채널 1로 CTS 패킷을 받고 있던 노드 *C*에서 노드 *B*의 CTS 패킷과 노드 *D*의 CTS 패킷이 충돌하게 된다.

CTS 패킷 전송 시 발생할 수 있는 문제가 ACK 패킷 전송에서도 발생할 수 있다. 그림 11와 같이 노드 *A*와 *C*는 수신용 채널로 채널 3을 사용하고 있다고 가정하고, 노드 *B*와 노드 *D*는 수신용 채널로 각각 채널 2와 채널 4를 사용하고 있다고 가정하자. 이때, 노드 *C*가 노드 *D*에게 DATA패킷을 전송하고 채널 3으로 ACK 패킷을 기다리고 있는 동안, 노드 *A*로부터 데이터를 수신한 노드 *B*가 채널3으로 노드 *A*에게 ACK 패킷을 전송할 경우 채널 3을 수신용 채널로 사용하고 있던 노드 *C*에서 노드 *D*로부터의 ACK 패킷과 노드 *B*로부터의 ACK 패킷이 충돌하게 된다.

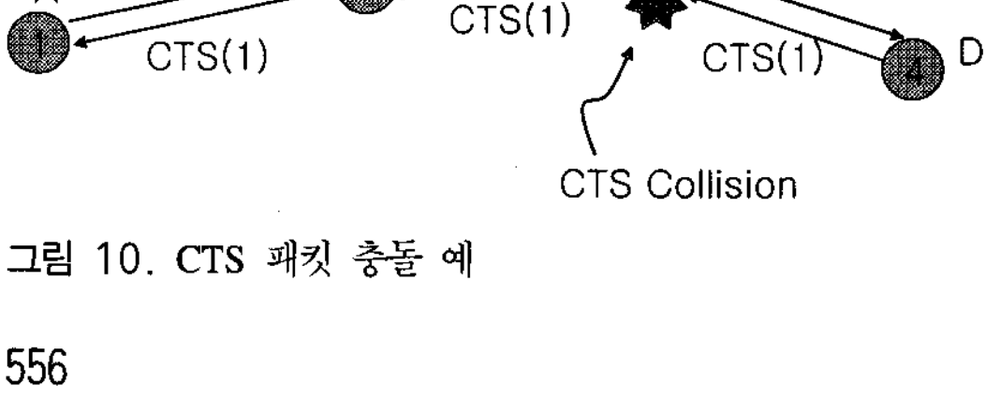


그림 10. CTS 패킷 충돌 예

본 논문에서는 이 문제로 인하여 발생하는 충돌을 줄이기 위하여 CTS WINDOW와 ACK WINDOW를 y개의 CS (CTS Slot) 또는 AS (ACK Slot)로 구성하고 (하나의 CS 또는 AS의 크기는 CTS 패킷 또는 ACK 패킷의 크기와 같다.) CTS 패킷 또는 ACK패킷을 전송하기 위해 *y*개의 타임슬롯 중 한 슬롯을 랜덤하게 선택하도록 하였다. 그리고 선택한 CS 또는 AS를 이용하여 패킷을 전송하도록 하였다. 예를 들어, *t*는 4라고 가정하고 그림 10과 같이 네트워크가 구성되어 있다고 가정할 때, 그림 12에서 노드 *B*에게 전송할 데이터가 있는 노드 *A*는 노드 *B*에게 채널 2로 RTS 패킷을 송신하고, 노드 *D*에게 전송할 데이터가 있는 노드 *C*는 노드 *D*에게 노드 *D*의 수신용 채널인 채널 4로 RTS 패킷을 보내 전송을 요청한다. 이때, 노드 B는 수신용 채널이

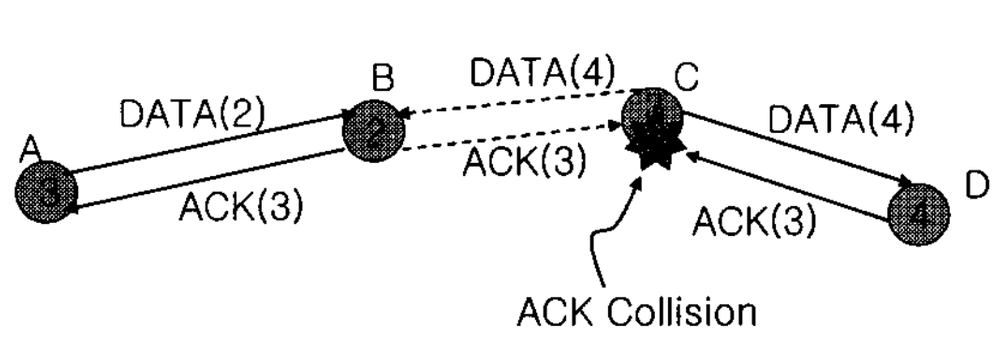


그림 11. ACK 패킷 충돌 예

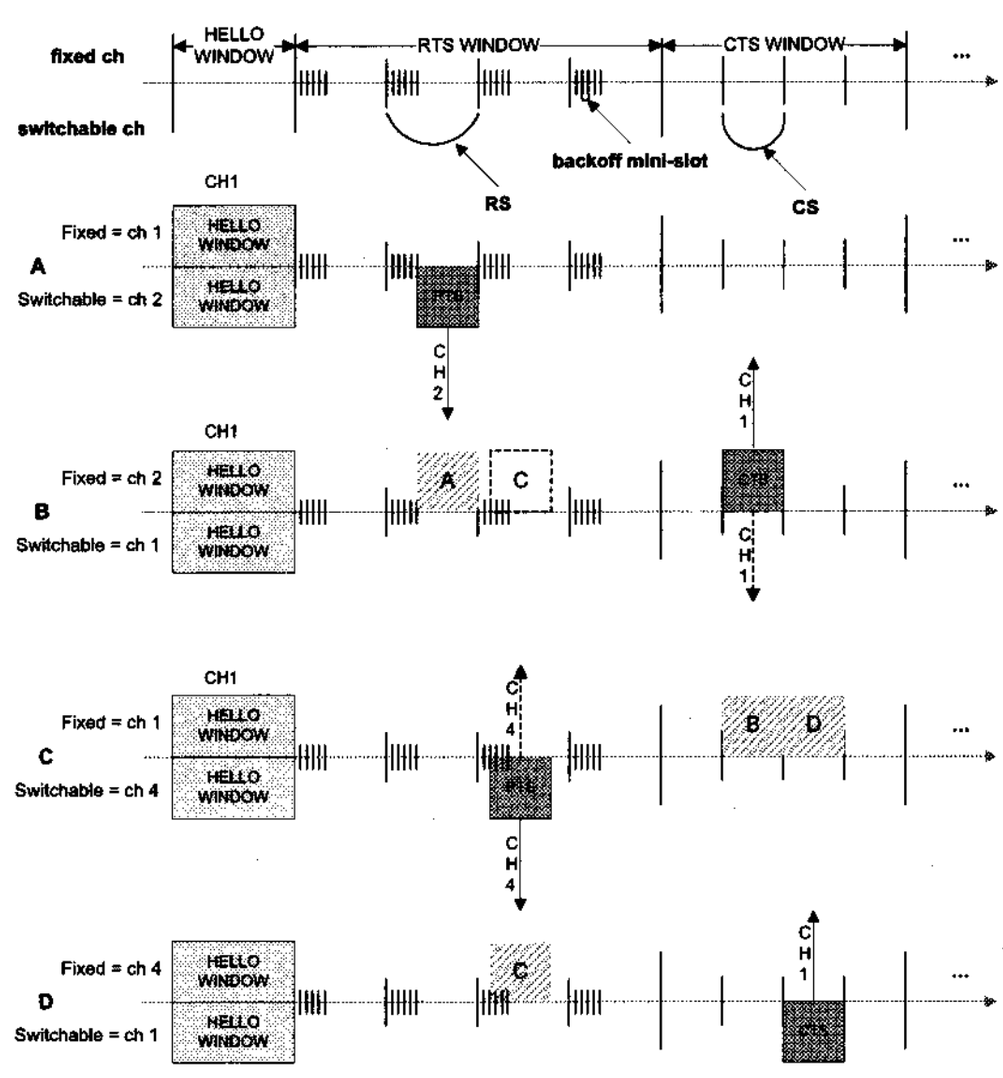


그림 12. CTS 패킷의 충돌을 완화시키는 예

채널 2이기 때문에 인접한 노드 *C*의 RTS 패킷을 수신하지 못한다. 그래서 노드 *A*로부터 RTS 패킷을 수신한 노드 *B*의 채널 1에서의 CTS 패킷과 노드 *C*로부터 RTS 패킷을 수신한 노드 *D*의 같은 채널 1에서의 CTS 패킷이 노드 *C*에서 충돌하게 된다. 하지만 본 논문에서는 CTS 패킷을 송신할 때 1에서 *t*사이의 CS 중 하나를 랜덤하게 선택하여 그 CS를 이용하여 CTS 패킷을 전송하기 때문에, 그림 12과 같이 두 번째 CS를 이용한 노드 *B*로부터의 CTS 패킷과 세 번째 CS를 이용한 노드 *D*로부터의 CTS 패킷이 충돌하지 않게 된다.

마찬가지로 그림 13에서, 노드들의 네트워크가 그림 11와 같이 구성되어 있다고 할 때, 노드 *A*로부터 DATA를 수신한 노드 *B*가 채널 3으로 ACK 패킷을 송신하고 노드 *C*로부터 데이터를 수신한 노드 *D*가 같은 채널 3으로 ACK 패킷을 송신하지만, 노드 *B*는 두 번째 AS를 선택하여 ACK 패킷을 송신하고 노드 *D*는 네 번째 AS를 이용하여 송신하기 때문에 노드 *C*에서 ACK 패킷이 충돌하지 않게 된다. 이는 CTS 패킷과 ACK 패킷의 충돌을 완전히 제거해 주지는 못하지만, CTS 패킷 또는 ACK 패킷을 전송할 때 *t*개의 슬롯 중 하나를 선택하여 전송하도록 하였기 때문에 CTS 패킷 또는 ACK 패킷의 충돌 확률을 $\frac{1}{t^2}$로 줄여줄 수 있다. 하지만 *t*가 커질수록 오버헤드 (overhead)가 증가하기 때문에,

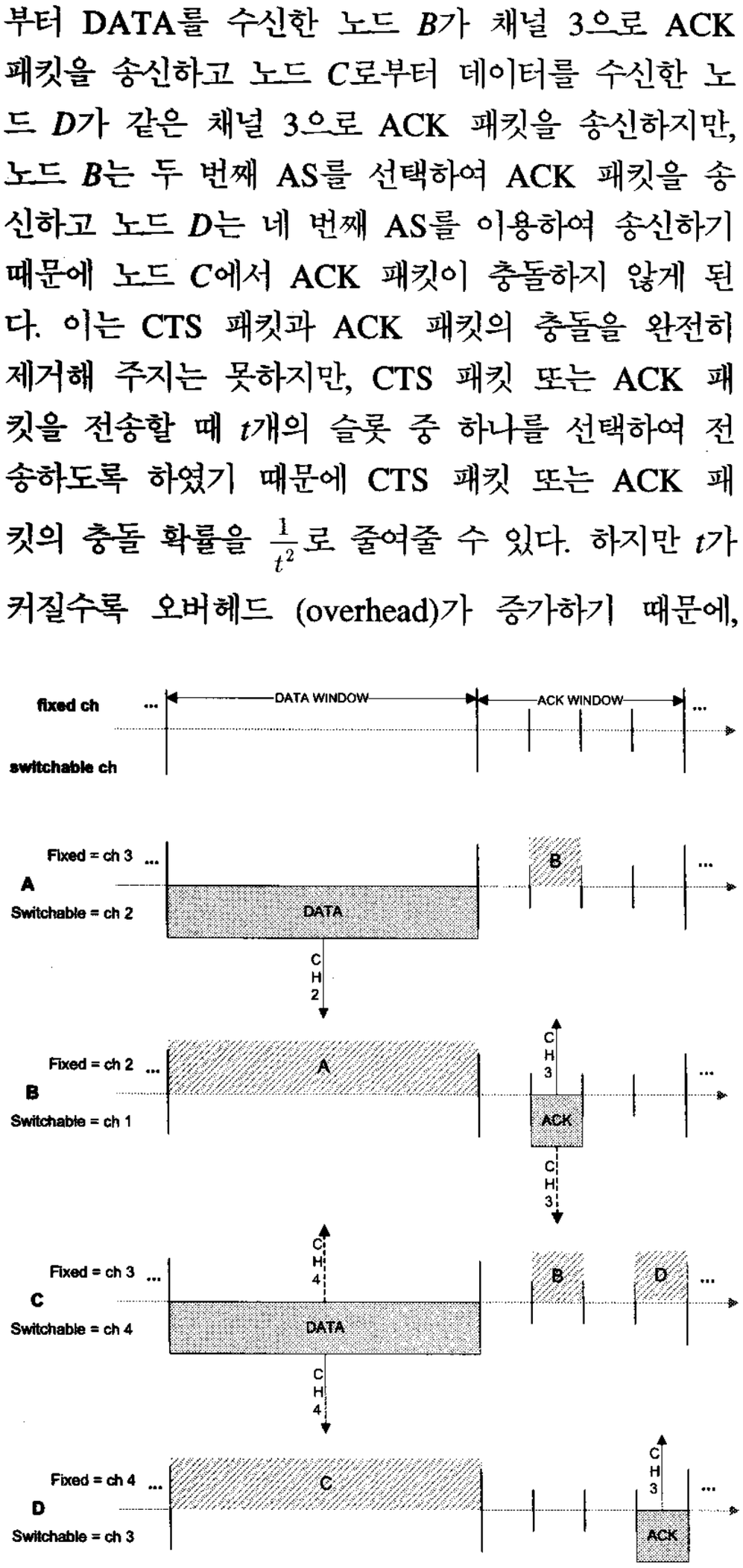


그림 13. ACK 패킷의 충돌을 완화시키는 예

네트워크의 상황에 따라 적절한 선택이 필요하다.

4.5 데이터 전송

CTS WINDOW동안 CTS패킷을 수신한 노드는 DATA WINDOW 동안 CTS 패킷을 송신한 노드에게 DATA 패킷을 전송한다. DATA WINDOW의 길이는 가장 긴 DATA 패킷의 길이와 동일하다고 가정한다. 그리고 DATA WINDOW 동안 DATA 패킷을 수신한 노드는 앞 절에서 설명한 것과 같이 ACK WINDOW 동안 t개의 슬롯 중 랜덤하게 선택한 한 슬롯을 이용하여 ACK 패킷을 전송한다. ACK WINDOW가 끝나면 다시 RTS WINDOW 기간 또는 HELLO WINDOW 기간으로 돌아가 다음 전송을 준비한다. RTS WINDOW 기간으로 돌아가는 경우는, 앞에서 설명한 것처럼 다음 RTS WINDOW에 RTS 패킷을 우선 전송할 수 있는 권한을 주는 CTS 패킷을 받은 노드가 있다면 RTS WINDOW에서 첫 번째 슬롯에 미니슬롯 백오프 없이 RTS 패킷을 전송하고, 그 이외에는 새롭게 전송을 시도하는 노드 혹은 이전 COMMUNICATION WINDOW에서 전송하지 못한 데이터가 남아 있는 노드가 있다면 다시 RTS 전송을 시도한다. HELLO WINDOW 기간으로 돌아가는 경우는, HELLO WINDOW의 길이에 따라 네트워크 전체가 HELLO 패킷을 교환하기 위해 시도하거나, 자신 또는 NeighborTable에 기록된 노드의 수신용 채널이 변경된 경우 HELLO 패킷 교환을 시도한다.

본 논문에서는 TDMA 방식을 사용하기 때문에 HMCP에서 발생하였던 문제인 CTS 패킷과 DATA 패킷의 충돌은 일어나지 않는다. 그리고 그림 8과 같이 송신 인터페이스로 데이터를 송신하는 동안 다른 노드로부터 데이터를 수신할 수 있기 때문에 채널의 수가 많은 경우 더욱 많은 수의 데이터 교환이 가능하여 네트워크 전체의 처리량을 높일 수 있다.

Ⅳ. 성능 분석

본 논문에서 제안한 멀티채널 MAC 프로토콜의 성능을 평가하기 위해 컴퓨터 모의실험을 수행하였으며, 본 논문에서 제안한 멀티채널 MAC 프로토콜과 비교대상인 DCA (on-Demand Channel Assignment)[3] 프로토콜과 HMCP[5] 를 구현하여 성능평가를 수행하였다. 시뮬레이션 시 적용된 파라미터 값은 표 1과 같고, 실험 결과 그래프 상의 모든 데이터 포인트는 20회 이상의 실험 후 평균을 내어 나타내었다.

표 1. 모의 실험 환경 요소

총 노드 수	100개
시뮬레이션 영역	100m×100m
전송 범위	10 meters
Data Rate	2 Mbps
Hello packet	14 Bytes
RTS packet	20 Bytes
CTS, ACK packet	14 Bytes
Data packet	2048 Bytes

본 논문은 첫 번째 실험으로 채널수에 따른 데이터의 처리량을 비교하였다. 처리량은 아래와 같이 계산되었다. 여기서 처리량은 초당 전송된 바이트 수이고, Total Time은 네트워크에서 전송이 진행된 총 시간이다.

$$Throughput = \frac{PacketLength \times NmofSuccessfulPacket}{TotalTime} \quad (1)$$

그림 14에서 DCA 프로토콜은 특정 채널수까지 처리량이 높아지다가 채널수가 5이상으로 갈수록 처리량이 낮아짐을 볼 수 있다. 이는 공통채널에서 처리할 수 있는 패킷량이 한정되어 있기 때문에 데이터 채널수가 증가하여도 데이터 채널 번호를 협상하기 위한 공통채널의 공간이 없기 때문이다. HMCP의 경우는 채널수가 증가할수록 데이터 처리량이 증가하기는 하나 데이터 패킷과 컨트롤 패킷의 충돌로 인하여 전체적으로 데이터 처리량이 완만하게 증가함을 볼 수 있다. 반면에 제안한 프로토콜은 채널수의 증가에 따라 처리량이 증가함을 볼 수 있다.

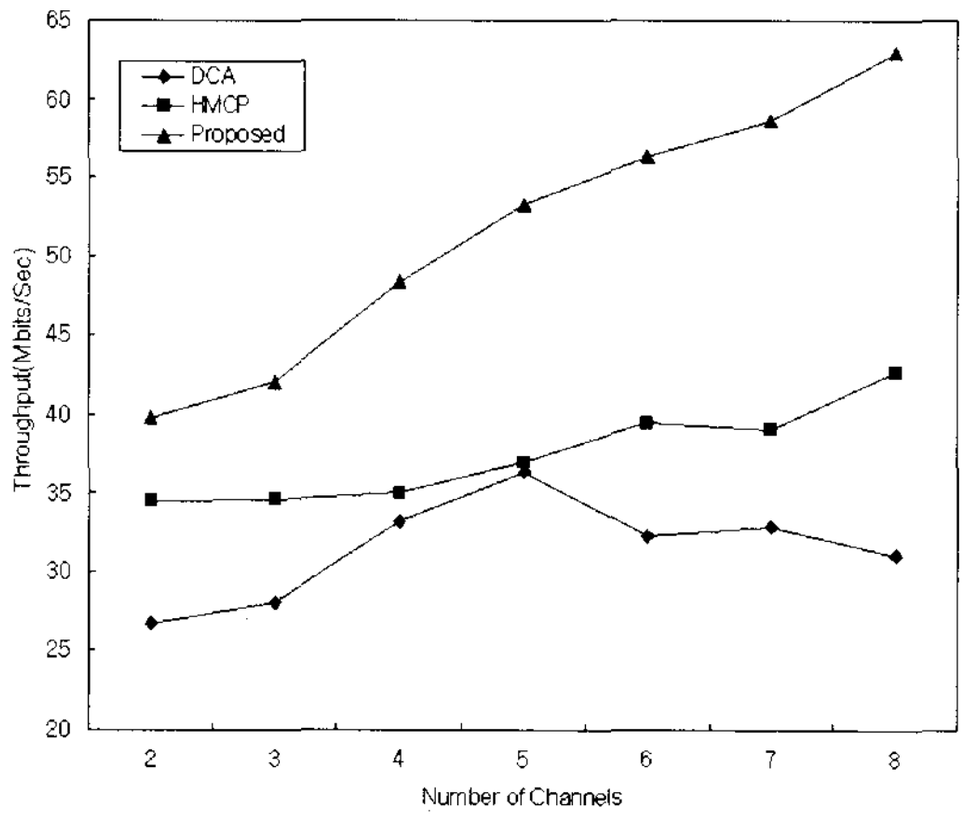


그림 14. 채널수에 따른 데이터 처리량

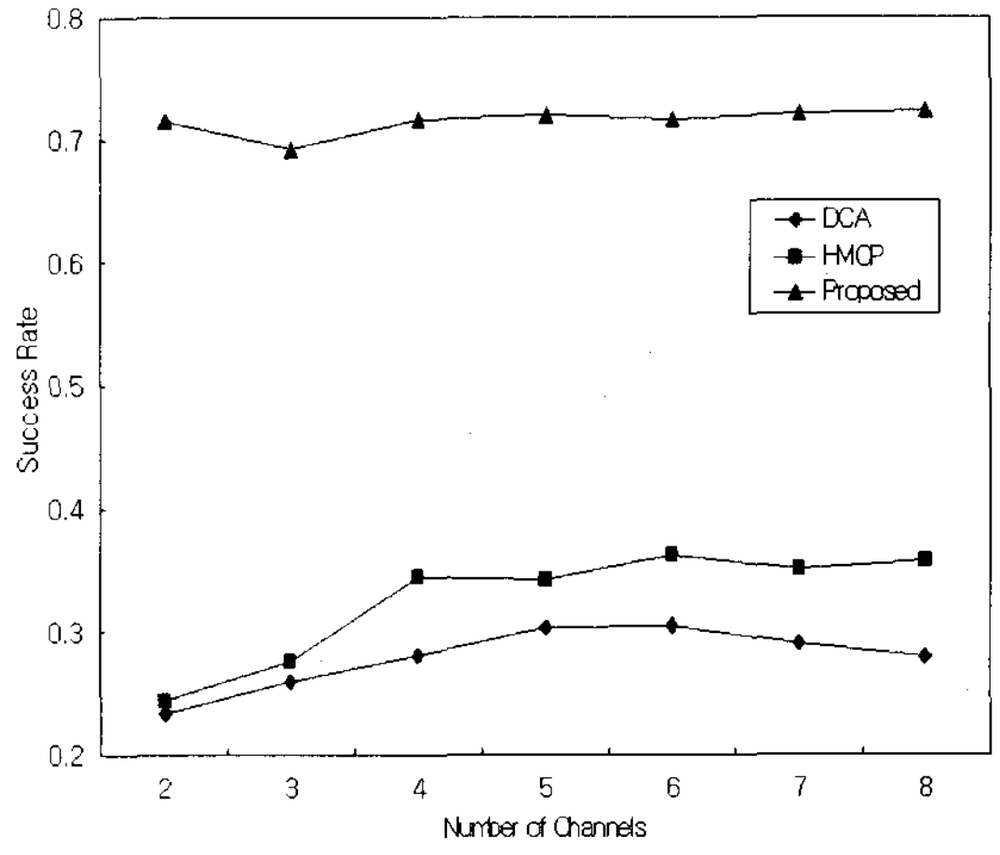


그림 15. 채널수에 따른 성공률

그리고 각각의 채널에서 데이터 교환을 위한 컨트롤 패킷을 교환하기 때문에, 데이터 채널을 사용하기 위해 공통채널에서의 컨트롤 패킷 교환 과정을 거쳐야 하는 DCA 프로토콜에 비해 전체적으로 높은 처리량을 보임을 알 수 있다.

그림 15는 채널수에 따른 성공률을 보여준다. 그림에서 나타낸 그래프는 채널수의 증가함과 관계없이 네트워크에서 생성되는 패킷의 수가 일정하도록 하여 측정하였다. 성공률은 아래와 같이 계산되었다.

$$SuccessRate = \frac{NmofTotalArrivedPacketst}{NmofTotalSendedPackets} \quad (2)$$

그림에서 DCA 프로토콜은 채널수가 5일 때까지 성공률이 증가하다가 성공률이 더 이상 증가하지 않는 것을 볼 수 있다. 이는 공통채널에서의 병목현상이 데이터 채널을 사용하기 위해 협상할 수 있는 노드의 수에 영향을 주어 채널수가 증가하여도 목적지에 도착할 수 있는 패킷의 수를 증가시킬 수 없기 때문이다. 반면에 HMCP의 경우는 채널수에 따라 조금씩 증가함을 볼 수 있다. 이는 채널수의 증가에 따라 NAV의 부재로 인한 컨트롤 패킷과 데이터 패킷의 충돌 확률이 줄어들기 때문이다. 하지만 여전히 낮은 패킷 성공률을 보인다. 본 논문에서 제안하는 프로토콜은 RTS WINDOW, CTS WINDOW, ACK WINDOW에서 각각 발생하는 컨트롤 패킷간의 충돌문제로 인하여 100%의 성공률을 보이지는 않지만, 두 프로토콜에 비해 전체적으로 높은 성공률을 보인다. 이는 본 논문에서 제안하는 프로토콜이 TDMA 방식을 이용하여 노드의 송

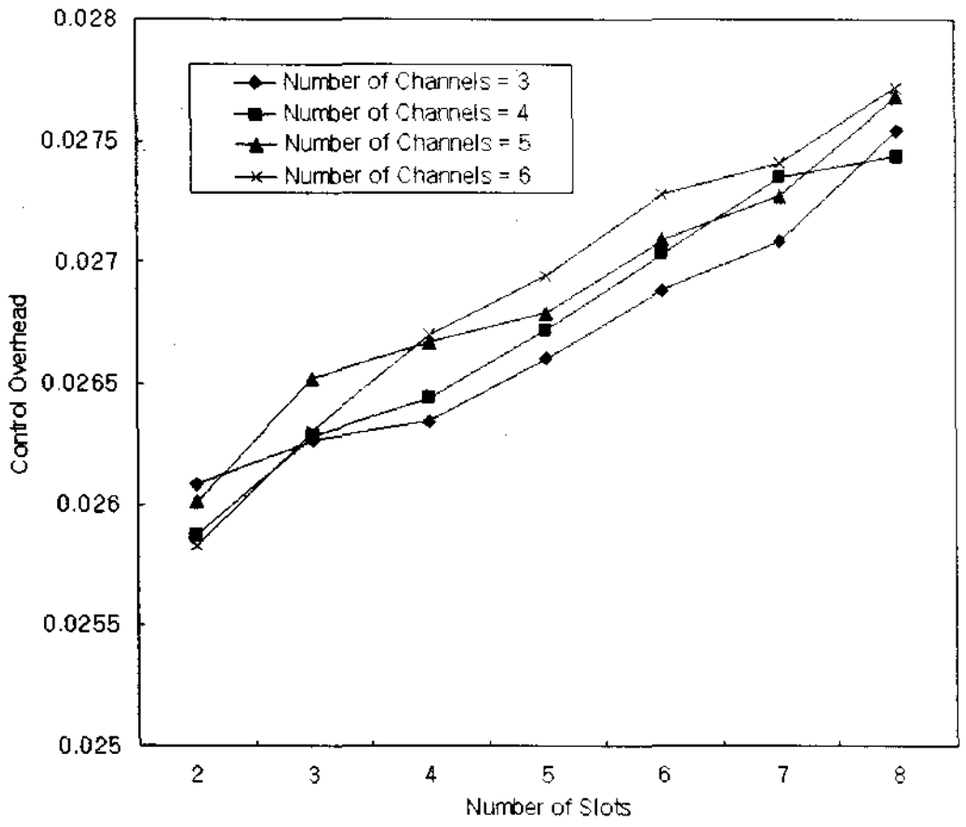


그림 16. 슬롯 수에 따른 패킷 오버헤드

수신이 동시에 가능하도록 하였기 때문에 나타난 결과이다.

그림 16은 본 논문에서 제안된 프로토콜의 슬롯 수에 따른 컨트롤 패킷의 오버헤드를 보여준다. 그림에서슬롯 수는 RTS WINDOW, CTS WINDOW와 ACK WINDOW 각각의 슬롯 수를 나타내고 컨트롤 오버헤드는 전체 패킷량에서, HELLO WINDOW의 슬롯 수를 5로 고정했을 경우, HELLO 패킷을 포함한 컨트롤 패킷량의 비율로 나타내었다. 그림과 같이 슬롯 수가 증가할수록 각 채널에서의 컨트롤 오버헤드가 증가함을 알 수 있고, 채널수가 증가할수록 각 노드에서 이웃노드와 다른 채널을 이용하여 데이터 전송을 시도할 기회가 많아짐으로 컨트롤 오버헤드가 증가함을 볼 수 있다.

V. 결 론

무선 네트워크 환경에서 단일 채널을 사용하는 MAC 프로토콜은 공통적으로 노드의 수가 증가할수록 네트워크의 성능이 크게 떨어지는 문제점이 발생하였고, 이를 해결하기 위해 멀티채널을 이용하는 여러 MAC 프로토콜이 제안되어 왔다. 하지만 멀티채널 환경에서 제안된 기존의 대부분의 연구들은 두 개의 인터페이스를 이용하여 한 인터페이스는 데이터 채널 협상 및 사용 중인 데이터 채널의 모니터링 등을 하기 위하여 고정으로 공통 컨트롤 채널에 할당하고 다른 인터페이스는 협상된 채널에서 데이터를 교환하기 위해 자유롭게 스위치 할 수 있도록 하여, 채널수가 많아질 경우 트래픽이 공통 컨트롤 채널에 집중되는 병목현상을 발생시켰다.

본 논문에서는 이를 위해 노드 당 최소 2개의 인터페이스를 가지고 있다고 가정하고, 하나의 인터페이스에는 수신용 채널을, 다른 인터페이스에는 송신용 채널을 할당하여 DATA 패킷의 송수신이 동시에 가능하도록 하여 공통채널에서 발생하였던 병목현상을 제거하고, 전체 네트워크 처리량을 향상시켰다. 그리고 TDMA 방식을 사용하여, 본 논문과 동일하게 각 노드에게 수신용 채널을 할당하는 HMCP에서 발생하였던 문제인 CTS 패킷과 DATA 패킷의 충돌문제를 해결하였다. 또한 멀티채널 환경에서 한 노드의 이웃노드 수가 이용 가능한 채널보다 많은 경우 자신과 같은 채널을 사용하는 이웃노드를 가질 수 있는데, 이는 송수신 인터페이스가 다른 채널에 할당된 경우 서로의 컨트롤 메시지를 듣지 못하여 CTS 패킷이나 ACK 패킷이 충돌하는 문제를 발생시킨다. 본 논문에서는 이 문제를 새롭게 정의하고, CTS 패킷이나 ACK 패킷의 충돌 가능성을 줄였다. 모의실험의 결과는 본 논문이 기존에 제안된 MAC 프로토콜에 비해 높은 성능 향상과 채널 효율성을 달성하였음을 알려준다.

참 고 문 헌

〔1〕 IEEE Std. P802.11, "Wireless LAN-Medium Access Control and Physical Layer Specification," 1999.

〔2〕 Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu, "A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks," International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), pp.232-237, 2000.

〔3〕 N. Jain and S. Das, "A Mutlichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Mutlihop Wireless Networks," in Proceedings of the 9th Int. Conf. on Computer Communications and Networks (IC3N), Oct. 2001.

〔4〕 P Kyasanur, J So, C Chereddi, NH Vaidya, "Multi channel Mesh Networks: Challenge and Protocols," *Wireless Communications*, Vol.13, No.2, pp.30-36, 2006.

〔5〕 Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *ACM Mobicom*, pp.114

-128, 2004.

〔6〕 Ashish Raniwala, Kartik Gopalan, and Tzicker Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *Mobile Computing and Communications* Review, Vol.8, No.2, pp.50-65, April 2004.

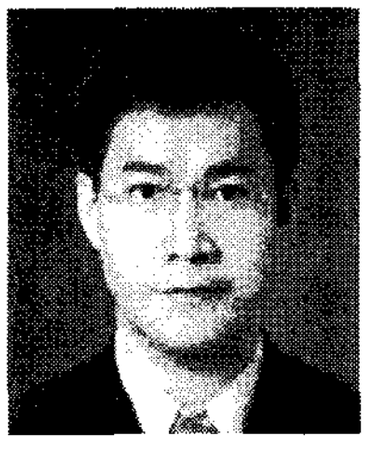
〔7〕 Jungmin So and Nitin H. Vaidya, "Multi-channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals using a Single Transceiver," in Mobihoc, 2004.

〔8〕 Paramvir Bahl, Ranveer Chandra, and John Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *ACM Mobicom*, pp.222-233, 2004.

김 영 경(Young-Kyoung Kim) 정회원

2006년 2월 인하대학교 컴퓨터공학과(공학사)
2006년 3월~2008년 2월 인하대학교 정보통신대학원 석사과정
2008년 2월~현재 PANTECH 연구원
<관심분야> Multi-Channel MAC Protocol, Corss-Layer

유 상 조(Sang-Jo Yoo) 정회원



1988년 2월 한양대학교 전자통신학과(공학사)
1990년 2월 한국과학기술원 전기 및 전자공학과(공학석사)
2000년 8월 한국과학기술원 전자전산학과(공학박사)
1990년 3월~2001년 2월 KT 연구개발본부
2001년 3월~현재 인하대학교 정보통신대학원 부교수
<관심분야> 초고속 통신망, 무선 MAC 프로토콜, 인터넷 QoS, Cross-layer 프로토콜 설계