IEEE 802.11 기반 Ad Hoc Wireless LAN에서의 실시간 통합 서비스를 위한 분산 MAC 프로토콜의 설계 및 분석

김 원 수*, 강 충 구

요 약

본 논문에서는 IEEE 802.11 규격을 기반으로 다중 홉 Ad Hoc 모드에서 비실시간 및 실시간 트래픽에 대한 통합 서비스를 지원하는 Wireless LAN을 구성할 때, 분산 방식으로 실시간 서비스의 우선권을 고려한 채널 액세스의 차별화를 제공할 수 있는 매체접근제어(Medium Access Control: MAC) 프로토콜의 구현 방안을 제안한다. 이를 위해 실시간과 비실시간 트래픽의 Contention Window (CW)를 서로 분리하며, 실시간 서비스의 지연 바운드에 따라 정해진 잔여 수명(residual time)을 우선권 설정에 반영함으로써 실시간 트래픽의 지연 요구 사항을 보장할 수 있도록 한다. 또한 실시간 트래픽의 부하에 따라서 적응적으로 offset 값을 조정함으로써 offset에 의해 발생될 수 있는 수율의 저하를 최소화한다. 제안된 방식은 기존 IEEE 802.11 MAC 규격의DCF (Distributed Coordination Function) 방식을 그대로 사용하면서 CSMA/CA 기반의 경쟁 방식을 통해서도 비실시간 트래픽과의 차별화가 가능하고, 또한 잔여 수명 시간을 고려한 동적인 우선권 할당이 가능하도록 하였다. 시뮬레이션에 의한 분석 결과에 따르면 기존 DCF와 비교할 때 실시간 단말의 용량을 2배까지 향상시킬 수 있으며, 트래픽 부하에 따라 적응적 제어를 통해 추가적인 성능 개선 효과를 확인하였다.

Key Words : Ad-Hoc network, Distributed MAC protocol, integrated service

I. 서 론

Wireless LAN (WLAN)의 전송 속도가 증가함에 따라 QoS 보장과 연계하여 실시간 서비스에 대한 질적 요구도 증가하고 있다. IEEE 802.11 규격 기반 표준 WLAN에서의 매체접근제어(Medium Access Control: MAC) 계층은 Distributed Coordination Function (DCF)와 Point Coordination Function (PCF)의 두 가지 메커니즘을 지원한다. DCF는 경쟁 구간에서 비동기 데이터(asynchronous data)를 전송하기 위한 기본 채널 액세스 메커니즘으로서, 매체접근제어 방식으로 CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) 프로토콜을 채택하고 선택적으로 RTS/CTS 교환 과정이 추가된 기능이다. CSMA/CA 방식은 부가적인 제어 메시지가 없고, 동작 과정이 단순하기 때문에 구현이 용이하다는 장점이 있는 반면, 경쟁기반이기 때문에 실시간 트래픽의 품질을 보장하는데 한계가 있다. 한편, PCF는 AP (Access Point)를 통해 폴링(polling) 방식으로 매체접근을 제어하는 중앙 집중적인 형태의 비경쟁적인 메커니즘이다. 즉, PCF는 지연 시간에 민감한 실시간 트래픽을 비경쟁적인 형태로 지원함으로써 QoS를 보장할 수 있는 것이 특징이다. 그러나, 많은 제어 메시지가 필요하고 구현이 매우 어려운 단점이 있다. 이러한 구현상의 문제로 인해서 현재의 기반 구조망(infrastructure network)에서 상용화되어 있는 대부분의 IEEE 802.11 표준 WLAN 시스템은 DCF만을 구현하고 있다. 한편, IEEE 802.11 표준 규격에서는 DCF를 통해서 AP의 제어 없이 단말간에 직접적인 통신이 가능한 Ad Hoc 모드를 지원하고 있다. 하지만, DCF 방식은 단말 간의 서비스 차별화를 위해 우선권을 제공할 수 있는 어떠한 메커니즘도 포함하고

* 삼성종합기술원(wonsoo.kim@partner.samsung.com)

※ 본 논문은 2003년 4월 JCCI 학술대회에서 우수논문으로 선정되어 게재 추천된 논문입니다.

있지 않으므로, 실시간과 비실시간 서비스가 통합된 환경에서 각 서비스의 QoS를 보장하는 것이 불가능하다.

현재 IEEE 802.11 TGe에서 진행되고 있는 표준화에서는 QoS 지원을 위한 분산방식의 MAC 메커니즘으로 DCF를 확장한 Enhanced DCF (EDCF)를 규정한 바 있다. 이는 각 Traffic Category (TC)별로 큐를 운용하고, 각기 다른 매개변수(CWmin, CWmax, Retry Count, AIFS 등)을 운용하는 방식이다. 이와 같은 개별 매개변수의 설정을 통해서 실시간과 비실시간 트래픽간의 차별화를 가능하게 할 수 있다. 하지만, 여전히 경쟁기반의 틀에서 동작하기 때문에 실시간 트래픽과 비실시간 트래픽간의 접속 우선권에 따른 차별화는 확률적으로만 가능할 뿐, 절대적인 우선권의 제공은 불가능하다. 따라서, 지연 시간에 민감한 실시간 트래픽에 대해서 QoS를 보장하는데 한계 요인으로 작용할 수 있다. 또한, 지연에 대한 제약을 가진 실시간 트래픽 중에서도 지연 바운드(delay bound)가 더 큰 트래픽은 실제로 겪은 지연으로 인해 잔여 수명이 작아질 때에는 지연 바운드가 작은 트래픽보다 더 높은 우선권을 가져야 하는 경우가 발생할 수 있다. 이와 같이 동적으로 우선권이 변동하는 경우에는 EDCF와 같이 각 클래스에 따라 매개 변수를 고정적으로 사용하는 방식은 지연 바운드를 갖는 실시간 트래픽의 품질을 보장하는데 한계가 있다.

본 논문에서는 실시간 트래픽의 지연 시간에 대한 민감성을 고려해 지연 바운드에 따른 잔여 수명을 이용해서 동적으로 우선권을 설정하는 방안을 제시한다. 특히, 본 설계의 핵심은 추가적인 메시지를 도입하지 않고 IEEE 802.11 규격의DCF (Distributed Coordination Function)을 그대로 사용하면서 Ad-hoc 모드에서 실시간 서비스의 제공이 가능하도록 한 것이다. 따라서, AP가 없이도 단말간 직접적인 연결이 가능하고, 이때 지연시간에 민감한 트래픽에 대해 매개변수만을 조정하여 지연 바운드를 보장할 수 있게 된다. 또한, 기존의 DCF를 그대로 사용하므로 AP가 존재하는 기반 구조망에서도 여전히 적용이 가능하다.

본 논문의 구성은 다음과 같다. II 장에서는 IEEE 802.11 표준 규격의 분산 MAC 방식에 대해 개략적으로 살펴 보고, III 장에서는 제안된 방식에 대한 구체적인 동작 과정 및 우선권을 부여하기 위한 방안에 대해 살펴 본다. IV 장에서는 제안된 방식에서 사용되는 시스템 매개변수(system parameter)와 트래픽 모델(traffic model)을 정의하고, V 장에서 시뮬레이션을 통한 성능 분석의 결과를 제시한다.

II. IEEE 802.11 분산 MAC 프로토콜의 개요

본 절에서는 IEEE 802.11 기반의 Ad-hoc WLAN에서 사용되는 매체 접근 제어 방식에 대해서 개략적으로 살펴본다. IEEE 802.11 기반의 Ad-hoc WLAN에서 사용되는 분산 방식은 크게 DCF와 EDCF로 구분할 수 있다 [1,2]. EDCF (Enhanced DCF)는 QoS를 지원하기 위해 기존의 CSMA/CA를 변형한 DCF의 확장된 개념으로서, IEEE 802.11 TGe를 통해 표준화가 진행되고 있다. 다음에서는 DCF 방식과 EDCF를 비교하면서 이들의 개략적인 동작을 기술한다.

EDCF는 기존의DIFS (DCF Inter Frame Space)를 8단계로 나누어서 프레임의 클래스(TC)에 따라 서로 다른 IFS (Inter Frame Space)를 적용함으로써 우선 순위에 차별을 두는 방식이다. 기존의 DCF 방식과 같이 CSMA/CA를 사용하는 것은 동일하지만, 각 트래픽의 특성에 따라서 차별화된 서비스를 제공하기 위한 목적으로 업데이트된 메커니즘이라고 볼 수 있다. 그림 1에서 보는 바와 같이 DCF의 경쟁 구간에서는 DIFS를 사용하며, EDCF에서는 TC TC : Traffic Class 별로 다른 AIFS (Arbitration IFS)를 설정함으로써 각 클래스 별 우선순위를 부여할 수 있다.

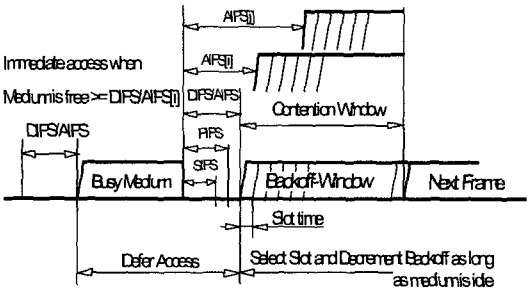


그림 1. IFS의 분류

AIFS AIFS 값은 최소값으로 PIFS (PCF IFS)를 가질 수 있고, 디폴트 값은 기존 DCF와 같은 DIFS 값을 갖는다.와 관련해서, EDCF를 지원하는 단말에서는 각 트래픽 클래스에 따라 독립적인 큐를 운용하게 된다. 즉, TC에 따라 별도의 큐가 할당되고,

각 TC 별로 별도의 경쟁 윈도우(Contention Window: CW)를 관리한다. DCF에서는 다음 (1)식과 같이 트래픽의 특성에 상관 없이 동등한 조건에서 Back-off 시간을 선택한다.

$$Backoff time = Random() \times aSlotTIme \quad (1)$$

(1)식에서 Random( )는 [0, CW] 사이의 균일한 분포에서 임의로 선택되고, 이때 CW는 $CW_{\min} \le CW \le CW_{\max}$의 범위에서 선택된다. 한편, EDCF의 경우는 다음 (2)식과 같이 각 트래픽 클래스에 대해서 지연에 따른 우선 순위에 따라서 서로 다른 Back-off 시간을 운용한다.

$$BackoffTime[i] = Random(i) \times aSlotTIme \quad (2)$$

(2)식에서 $Random[i]$는 [1,CW[i]+1] 사이에서 임의로 선택되고 여기서 CWmin이 1부터 시작되는 이유는 AIFS의 최소값이 PIFS이기 때문에 만약 AP가 채널을 잡으려고 할 때 생기는 충돌을 방지하기 위한 것이다. 이때 CW[i] 값은 $CW_{\min}[i] \le CW[i] \le CW_{\max}$ (혹은 $CW_{\max}[i]$)의 범위에서 선택된다. 여기서, i는 TC 값(우선 순위)을 나타낸다.

한편, DCF와 EDCF의 또 다른 차이점은 재시도 과정에 있다. DCF와 같이 EDCF에서도 Short Retry Counts (SRC)와 Long Retry Counts (LRC)를 유지하는데, 이 경우에 EDCF 단말은 각각의 큐에 대해서 다른 Retry Counts로 Short Retry Counts for Queue i (QSRC[i])와 Long Retry Counts for Queue i (QLRC[i])를 사용한다.

III. 제안된 분산 MAC 프로토콜

DCF 방식은 일반적인 CSMA/CA 방식에 은닉 단말 문제(Hidden Terminal Problem)를 해결하기 위해 RTS-CTS 교환 과정을 선택적으로 적용한다. 이러한 메커니즘에 추가적으로 실시간 트래픽의 QoS를 보장하기 위한 첫번째 제안 사항으로는 실시간 트래픽의 경쟁 구간과 비실시간 트래픽의 경쟁 윈도우를 물리적으로 분리 시킴으로써 실시간 트래픽이 절대적인 우선권을 가지도록 하는 것이다. 또한, 지연 바운드를 갖는 실시간 트래픽 간에 동적인 우선 순위를 부여를 위해서 각 단말 별로 잔여 수명(패킷이 큐에 도착한 시간 이후 지연 바운드에 도달할 때까지 남은 시간)에 따라 우선권을 제공하는 방안이 필요하다. 본 제안 방식에서는 이를 위해 잔여 수명에 따라 차별화된 back-off 시간을 결정하는 새로운 확률 분포 함수를 도입한다. III.1 소절에서는 실시간 및 비실시간 단말간의 분리된 채널 액세스 구간을 부여하는 방안을 제시하며, III.2절에서는 실시간 단말간의 우선 순위를 고려하는 방안을 제시한다. 본절에서 사용되는 각 매개변수 및 해당 기호의 정의는 표 1과 같다.

표 1. 매개변수 및 기호의 정의

Notation	Definition
$CW_I^{(rt)}$	Contention window size of non-real-time station after i-th collision
$CW_i^{(nrt)}$	Contention window size of non-real-time station after i-th collision
CW_{RT_max}	Maximum contention window size of real-time station
CW_{NRT_max}	Maximum contention window size of non-real-time station
CW_{RT_min}	Minimum contention window size of real-time station
CW_{NRT_min}	Minimum contention window size of non-real-time station
L	Maximum number of collisions until which the offset value can be applied
$B_i^{(rt)}$	Back-off time of real-time station after i-th collision
$B_i^{(nrt)}$	Back-off time of non-real time station after i-th collision
$\dot{k}$	Laxity factor; defined by equation (7)
τ_R	Residual time of real-time station
$D_{\max}$	Delay bound of real-time station
STA_{RT}	Real-time station
STA_{NRT}	Non-real-time stations
STA_{NRT_hidden}	Non-real-time hidden terminals

1. 실시간 단말과 비실시간 단말간의 채널 액세스 차별화 방식

제안된 분산 방식에서는 실시간과 비실시간 단말에 대해 각기 다른 크기의 경쟁 윈도우를 도입한다. 이들 경쟁 윈도우의 크기를 각각 $CW_i^{(rt)}$과 $CW_i^{(nrt)}$로 나타내며, 이 값들은 랜덤 접속에 따른 STA간의 충돌 횟수 i에 따라 결정된다. 즉, STA간의 충돌이 발생할 때마다 $CW_i^{(rt)}$ 및 $CW_i^{(nrt)}$는 각각 최대 CW_{RT_max} 및 CW_{NRT_max}까지 지수적으로 증가한다. 이때, 최대 L번 충돌이 발생할 때까지 STA_{NRT}는 offset 값만큼의 대기 시간을 더해서 back-off 시간을 선택한다. 이 offset 값은 STA_{RT}의 채널 액세스 구간을 보장하기 위해 양보하는 시간 값이 된다. 충돌이 발생할 경우 비실시간 단말의 back-off 시간($B_i^{(nrt)}$)은 $[0, CW_i^{(nrt)}]$에서 선택된다. 이때 STA_{NRT}는 패킷이 처음 발생한 후 재시도 횟수가 L번이 될 때까지 액세스 차별화를 위해 offset 값을 두고, 그 후에는 offset 값을 0으로 둔다. 이는 STA_{NRT}의 평균 지연 시간이 급격하게 증가하는 것을 막기 위한 것이다. 물론, $CW_i^{(nrt)}$는 충돌이 발생할 때마다 지수적으로 증가하며, 그 최대 가능한 값은 CW_{NRT_max}으로 제한된다. 따라서, STA_{NRT}를 위한 back-off 시간($B_i^{(rt)}$)은 다음과 같이 나타낼 수 있다.

$$B_i^{(nrt)} = \begin{cases} offset + Rand[0, CW_i^{(nrt)}] \times SlotTime, & 0 \le i \le L \\ Rand[0, CW_i^{(nrt)}] \times SlotTime, & i > L \end{cases} \quad (4)$$

(4)식에서 $CW_i^{(nrt)}$는 [CW_{NRT_min}, CW_{NRT_max}]의 경쟁 윈도우 범위 내에서 $2^i \cdot CW_{NRT_min}$의 값을 가지며, $Rand[0, CW_i^{(nrt)}]$는 $[0, CW_i^{(nrt)}]$의 범위에서 균일(Uniform) 분포를 통해 추출된 유사 임의 정수를 나타낸다. 한편, START를 위한 back-off 시간($B_i^{(rt)}$)은 다음과 같이 결정된다.

$$B_i^{(rt)} = RandomG[0, CW_i^{(rt)}] \times SlotTime, i \ge 0 \quad (5)$$

(5)식에서 $CW_i^{(rt)}$는 (CW_{RT_min}, CW_{RT_max})의 경쟁 윈도우 범위 내에서 $2^i \cdot CW_{RT\_min}$ 값을 가지며, $RandomG[0, CW_i^{(rt)}]$는 $[0, CW_i^{(rt)}]$의 범위에서 제안된 확률 분포 함수 G( )를 통해 추출된 유사 임의 정수이다. 실시간 단말간의 우선권 부여를 위해 이 새로운 확률 분포 함수 G( )를 도입한 것이 기존 DCF 메커니즘에서 back-off 시간을 결정하는 방식과의 차이점으로서, 구체적인 함수의 형태는 III.2 절에서 상세하게 다룬다. (4)식 및 (5)식에 의해서 $B_i^{(rt)}$ 및 $B_i^{(nrt)}$는 구분될 수 있고, 이로 인해서 실시간 단말이 비실시간 단말보다 채널에 우선적으로 접근하는 것이 보장될 수 있다.

그림 2는 (4)식 및 (5)식에 의해서 채널 액세스 구간을 분리하였을 경우, STA_{RT}와 STA_{NRT}의 초기 채널 액세스 과정을 설명한 예이다. 본 예시에서 채널 액세스 구간을 분리하기 위한 offset 값은 8, 그리고 STA_{RT}와 STA_{NRT}의 최소 경쟁 윈도우 값을 각각 4로 두었다 (즉, CW_{RT_min}= CW_{NRT_min}=4). 만약 각 단말이 경쟁 윈도우에서 임의로 4의 값을 각각 선택하였다고 가정해 보자. 이 때, STA_{RT}는 $B_0^{(rt)} = 4$, STA_{NRT}는 offset 값 8에 의해 $B_0^{(nrt)} = 12$로 주어진다. 여기서, $B_0^{(nrt)}$의 offset 값에 의해서 어떤 조건 (예를 들어, $B_0^{(nrt)}$가 임의의 값을 0으로 선택하는 경우)에서도 초기 액세스 시도 시 STA_{RT}의 $B_0^{(rt)}$보다 항상 큰 값을 가지게 된다.

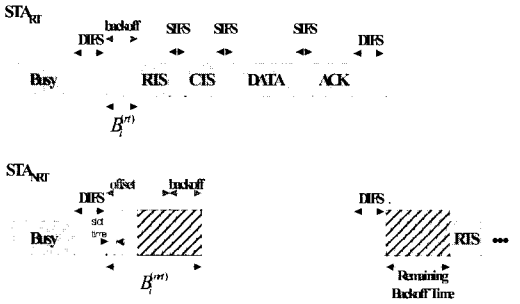


그림 2. 초기 채널 액세스 과정의 예시

STA_{RT}와 STA_{NRT}의 채널 액세스 구간을 분리하는 문제에서 중요한 고려 사항은 offset 값의 설정이다. 여기서는 2가지 형태의 접근 방안을 고려한다. 실시간 단말의 부하에 따라 offset 값을 적응적으로 설정하는 방안과 실시간 트래픽의 부하에 관계없이 고정 offset 값을 적용하는 방안이다.

단일 홉 Ad Hoc 기반의 무선랜에서 하나의 BSS (Basic Service Set) 내에 있는 단말들은 서로 상호 통신이 가능하기 때문에 각각의 충돌에 대해서 인지하고, 자신의 CW 값을 상호 간의 정보 교환을 통해서 업데이트하는 것이 가능하다. 즉, STANRT는 실시간 단말과의 정보 교환으로 offset 값

($CW_i^{(rt)}$)을 갱신하는 방식을 통해 유동적으로 실시간 단말들의 채널 액세스 구간을 보장해 줄 수 있다. 하지만, 이러한 방식은 성능면에서 개선의 여지가 있으나, 부가적인 정보 교환으로 인한 오버헤드가 증가할 가능성이 크다.

한편, 다중 홉 Ad Hoc 망으로의 확장성을 고려할 때 은닉 단말(Hidden Terminal)에 의한 중첩 문제(overlapping problem)을 고려해야 한다. 비실시간 은닉 단말(STA_{NRT_hidden})이 실시간 단말의 채널 접근에 방해를 주지 않기 위해서는 STA_{NRT_hidden}에게 BSS내 실시간 단말의 offset 정보를 알려서 STA_{NRT_hidden}이 offset 값을 갱신할 수 있어야 한다. 즉, 충돌이 발생할 때마다 실시간 단말이 $CW_i^{(rt)}$ 값을 늘려주는데, 하나의 셀 내의 모든 단말은 상호 정보 교환을 통해 자신의 back-off 크기를 늘리는 것은 가능하지만, STA_{NRT_hidden}의 경우는 이를 인지하지 못한다.

그림 3은 다중 홉 환경을 고려했을 때 발생하는 이와 같은 중첩 문제에 대해 예시하고 있다. Offset 값을 유동적으로 적용하는 방식의 경우 STA_{RT}의 초기값 $CW_0^{(rt)}$에 의해서 offset 값이 결정된다. 이후에 충돌이 발생할 때 BSS 내부의 STA_{NRT}는 STA_{RT}와의 정보 교환을 통해서 자신의 offset 값을 $CW_1^{(rt)}$로 갱신하고 실시간 단말의 채널 액세스 구간을 보장할 수 있는 반면에, STA_{NRT_hidden}의 경우는 그러한 정보 교환이 이루어지지 못하기 때문에 offset 값을 그대로 $CW_0^{(rt)}$ 값을 유지하고 있다. 따라서, 그림 3에서 보는 바와 같이 STA_{RT}와 STA_{NRT_hidden}의 경쟁 윈도우가 서로 겹치게 되는 영역이 발생하게 되고, 이로 인해 STA_{RT}의 채널 액세스 구간을 보장할 수 없다.

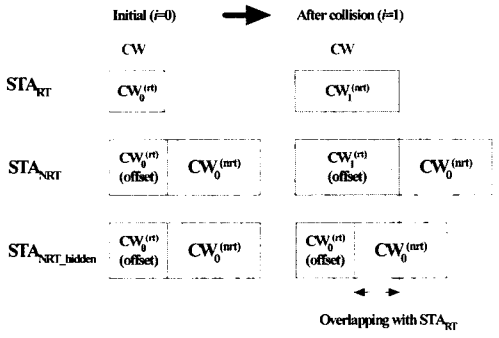
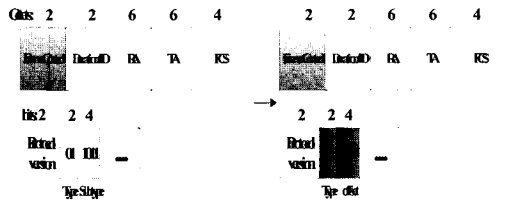


그림 3. 비실시간 은닉 단말과 실시간 단말간의 중첩 문제

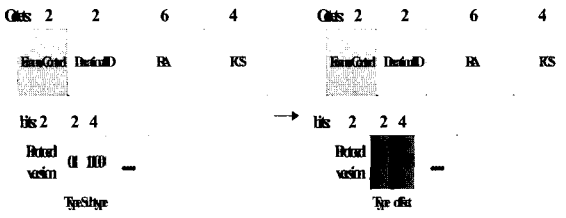
따라서, 이러한 문제에 대한 해결할 수 있는 가장 단순한 방법으로는 (4)식과 같이 초기 back-off 시간 $B_0^{(nrt)}$를 구하는 과정에서 offset 값으로 실시간 단말의 경쟁 윈도우의 최대값인 CW_{RT_max}를 사용하는 것이다. 이와 같이 고정된 offset 값을 이용해서 채널에 접근하면 STA_{NRT_hidden}이 실시간 단말보다 우선적으로 채널에 접근할 수 없도록 제한할 수 있다. 하지만, 이러한 고정 offset 값을 두는 방식은 수율 성능의 심각한 저하가 발생한다. 즉, 최대 크기의 offset 값을 고정적으로 설정함으로써비실시간 단말의 불필요한 지연을 가져오게 된다. 또한, offset을 작게 하기 위해서 CW_{RT_max} 값을 작게 설정하는 경우 실시간 단말의 back-off 크기가 적절하게 증가되지 않아서 충돌 해결이 원활히 이루어질 수 없게 되고, 이로 인해 실시간 단말의 패킷 드롭 확률이 크게 증가할 수 있다.

본 논문에서는 단일 홉 혹은 다중 홉 개념의 망 구조에서 다음과 같이 RTS (Request To Send)와 CTS (Clear To Send) 메시지에 실시간 단말의 경쟁 윈도우 값을 설정해서 보냄으로써 부가적인 오버헤드 없이 비실시간 단말의 offset 값을 트래픽 부하에 따라 적응적으로 설정하는 방식을 제안한다. 기존의 IEEE 802.11 Ad-hoc MAC 프로토콜에서 데이터 전송을 하기 전에 RTS-CTS 정보 교환을 통해서 은닉 단말 문제를 해결한다. 본 논문에서는 RTS 메시지와 CTS 메시지의 Frame control 헤더 부분의 예약된 필드를 이용해서 실시간 단말의 offset 값을 알려주는 방식을 사용한다. 우선 예약된 필드를 사용하였으므로 기존의 메커니즘과의 상호 호환성을 지속적으로유지할 수 있고, 기존의 메시지에 추가적인 비트 삽입이 없기 때문에 별도의 오버헤드가 없이 offset 정보를 알려줄 수 있다. 그림 4 (b)와 그림 5 (b)는 각각 변경된 RTS와 CTS 메시지의 형태를 나타낸다.



(a) IEEE 802.11 RTS message format (b) Modified RTS message format

그림 4. 실시간 단말을 위한 변경된 RTS 메시지 포맷



(a) IEEE 802.11 CTS message format (b) Modified CTS message format

그림 5. 실시간 단말을 위한 변경된 CTS 메시지 포맷

그림 4 (b)와 그림 5(b)에서 Type 필드값 11은 예약된 값이다. 즉, 본 논문에서의 11 값은 실시간 단말의 RTS 혹은 CTS 메시지를 명시한다. 11의 Type 값 이후에 따르는 4 비트는 실시간 단말의 offset 값을 명시한다. 즉, $i(0 \leq i \leq 15)$의 값을 기입해서 전송을 하게 되면, 그 메시지를 받은 비실시간 단말은 offset 값으로 2^i 값을 설정한다.

그림 6은 변경된 RTS-CTS를 이용한 offset 설정 과정을 예시한 것이다. 여기서, 3번 단말은 1번 단말 관점에서 은닉 단말이다. 우선 실시간 단말 1번노드가 자신의 CW 값을 offset으로 설정해서 비실시간 단말 2번 노드에 전송한다. 변경된 RTS를 1번 노드로부터 수신한 2번 노드는 자신의 offset 값을 255로 설정하고 변경된 CTS를 1번 노드에 전송한다. 또한, 변경된 CTS는 1번 노드의 비실시간 은닉 단말 3번 노드에 전송되고, 3번 노드 역시 255의 offset 값을 설정한다. 실시간 단말 4번 노드가 트래픽이 발생할 경우 역시 자신의 CW 값인 32를 offset으로 설정하고 비실시간 단말 5번 노드에 전송한다. 마찬가지로 5번 노드는 32를 자신의 offset으로 설정하고 변경된 CTS를 4번 노드에 전송한다. 이 때 제안된 방식에 따라 비실시간 은닉 단말인 3번 노드는 현재 설정된 offset 값인 255와 4번 노드로부터 전송된 RTS 메시지에 설정된 offset 값을 비교한 후 큰 값을 유지하게 된다. 즉, 이 경우에는 3번 노드의 offset 값이 255로 그대로 유지하게 된다. 또한, 비실시간 단말의 경우 트래픽 전송을 위해서 사용하는 RTS 메시지는 다른 단말에 별도의 offset 값 설정을 요구하지 않기 때문에 기존의 RTS를 그대로 사용한다.

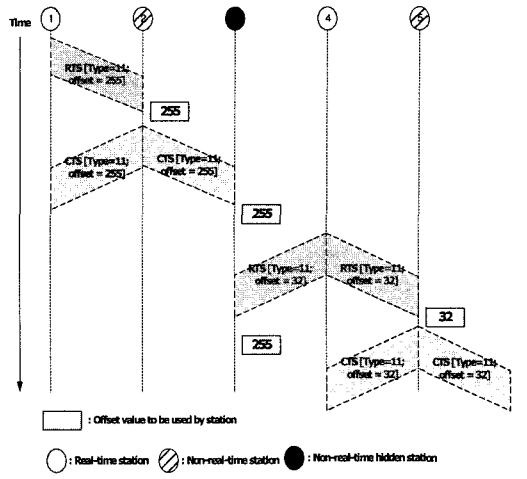


그림 6. 변경된 RTS-CTS 교환 과정의 예

제안된 방식을 통해서 기존의 DCF 방식과 공존성을 유지하면서 별도의 오버헤드 없이 RTS-CTS 교환 과정을 통해서 단일 홉 및 다중 홉 망 구조에서 offset 값을 트래픽의 부하에 따라서 유동적으로 설정할 수 있고, 이를 통해서 망의 수율과 비실시간 단말의 불필요한 지연을 막을 수 있게 된다. 따라서, 고정 offset을 설정하는 방식에서 문제점을 제안된 적응적 offset 방식을 통해서 해결할 수 있다.

2. 실시간 트래픽간의 동적 우선권 제어 (Dynamic Priority Control: DPC) 방식

지연 바운드를 갖는 실시간 트래픽 간의 우선 순위를 결정하는 기준은 패킷 도착 시간 대비 잔여 수명(Residual Lifetime)을 이용한다. 따라서, 본 논문에서는 각 단말의 잔여 수명에 따라 결정되는 확률 분포 함수 G( )에 따라 우선권을 부여한다. 즉, G( )의 역할은 각 실시간 단말의 잔여 수명에 따라 요구되는 우선권을 적절하게 설정하기 위해 필요한 확률 분포를 할당하고자 하는 것이다. 본 절에서는 G( )에 적합한 1차 및 2차 함수 형태를 살펴 보고, 이에 따른 충돌 확률을 비교 분석해서 적절한 함수 및 관련 매개 변수를 도출하고자 한다.

그림 7과 같이 1차 함수 형태로 주어진 확률 분포 함수 G( )가 잔여 수명에 따라 우선권을 부여한 경우 그림 8과 같은 형태로 나타낼 수 있으며, 이를 수식적으로 나타내면 다음과 같다.

$$G_k(x)=\begin{cases}\dfrac{2}{k\cdot CW_{RT}}x, & 0\le x\le k\\ \dfrac{-2}{CW_{RT}(CW_{RT}-k)}\cdot(x-CW_{RT}), & k\le x\le CW_{RT}\end{cases} \quad (6)$$

(6)식에서 k는 잔여 수명에 따라 확률 분포 함수를 제어하는 laxity factor라고 부르는 매개변수이며, $G_x(x)$는 확률 분포 함수이므로 $\int_0^{CW_{RT}} G_k(x)dx=1$을 만족한다. 즉, laxity factor k는 실시간 트래픽의 지연 바운드 D_{max} 대비 잔여 수명(τ_R)에 CW_{RT}를 재조절한 값인데, 다음과 같은 식으로 나타낼 수 있다.

$$k=\frac{\tau_R}{D_{\max}}\cdot CW_{RT} \quad (7)$$

(7)식에 의하면, 잔여 수명이 작을 경우에는 $G_k(x)$가 그림 8(a)와 같이 확률적으로 작은 값의 경쟁 윈도우를 갖게 됨으로써 상대적으로 높은 우선 순위를 실현하게 된다. 그 반대의 경우에는 그림 8(b)에서 보는 바와 같이 확률적으로 큰 값의 경쟁 윈도우를 갖게 됨으로써 상대적으로 낮은 우선 순위를 실현하게 될 것이다.

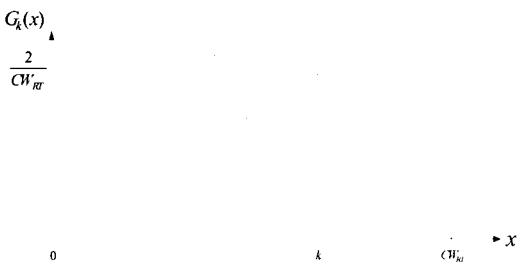


그림 7. 1차 함수 형태의 확률 분포 함수 $G_k(x)$

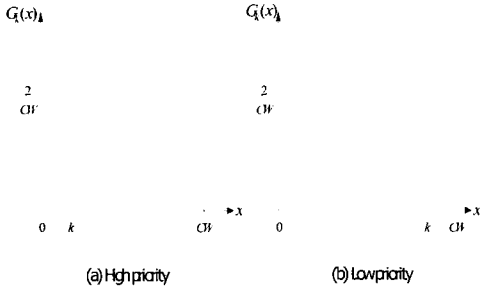


그림 8. Laxity factor k 값에 따른 1차함수 형태 분포 함수 $G_k(x)$의 예시

한편, 동일한 맥락으로 2차 함수 형태의 확률 분포 함수 $G_x(x)$를 다음과 같이 나타낼 수 있다.

$$G_k(x)=\begin{cases}\dfrac{4(1-CW_{RT}\times\epsilon)}{CW_{RT}\cdot k^2}\cdot x^2+\epsilon, & 0\le x\le\dfrac{k}{2}\\ \dfrac{4(CW_{RT}\times\epsilon-1)}{CW_{RT}\cdot k^2}\cdot(x-k)^2+\dfrac{2}{CW_{RT}}-\epsilon, & \dfrac{k}{2}<x\le k\\ \dfrac{4(CW_{RT}\times\epsilon-1)}{CW_{RT}(k-CW_{RT})^2}\cdot(x-k)^2+\dfrac{2}{CW_{RT}}-\epsilon, & k<x\le\dfrac{k+CW_{RT}}{2}\\ \dfrac{4(1-CW_{RT}\times\epsilon)}{CW_{RT}(k-CW_{RT})^2}\cdot(x-CW_{RT})^2+\epsilon, & \dfrac{k+CW_{RT}}{2}<x\le CW_{RT}\end{cases} \quad (8)$$

(8)식의 k 값도 1차 함수 형태와 마찬가지로 (7)식에 의해 주어진다. (8)식으로 주어진 확률 분포 함수를 도시하면 그림 9와 같다. 그림 10은 잔여 수명에 의해 정해진 우선 순위의 정도에 따라 결정된2차 함수 형태의 분포 함수를 예시하고 있다.

한편, 2차 함수 형태의 분포 함수에서 추가로 고려해야 할 것은 ϵ 값의 선택이다(단, ϵ 값은 항상 $1/CW_{RT}$보다 작아야 함). 일반적으로, 이 값은 여러 단말이 임의의 잔여 수명을 가지고 있을 때 충돌을 최소화할 수 있도록 결정되어야 할 것이다. 즉, ϵ 값에 따라서 $G_k(x)$의 볼록한 정도가 그림 11과 같이 달라질 수 있다.

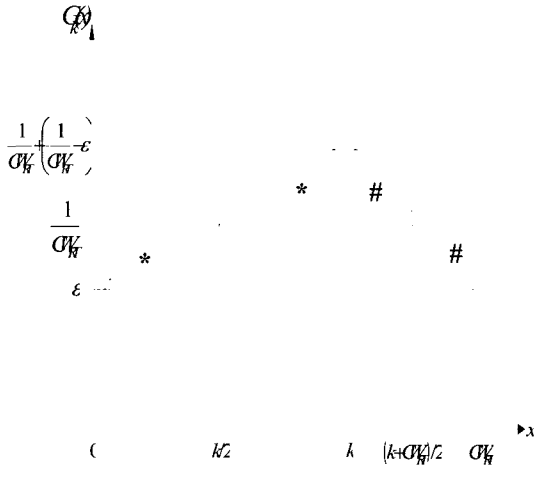


그림 9. 2차 함수 형태의 확률 분포 함수 $G_k(x)$

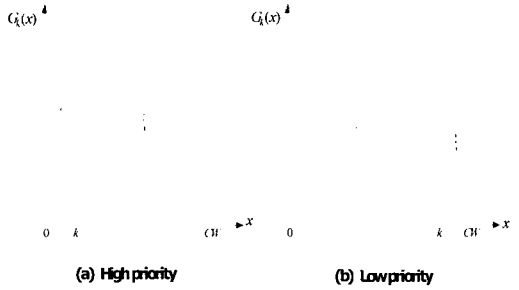


그림 10. Laxity factor k 값에 따른 2차 함수형태 분포 함수 $G_k(x)$의 예시

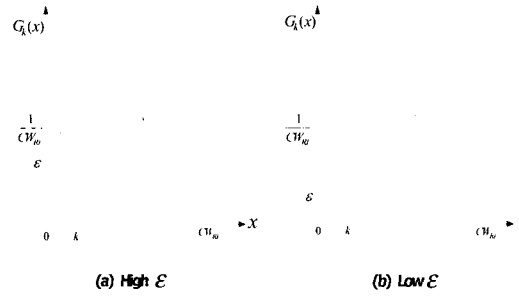


그림 11. ϵ 값에 따른 $G_k(x)$의 예시

우선권을 설정해 주는 확률 분포 $G_k(x)$의 결정에서 우선적으로 고려되어야 하는 정량적인 측면은 충돌 확률이다. 따라서, 동등한 조건에서 가장 낮은 충돌 확률 성능을 갖는 매개 변수를 결정하기 위해 시뮬레이션을 수행하였다. 시뮬레이션에서 각 단말의 잔여 수명은 [0,D_{max}]의 범위에서 임의로 결정하였다. 그리고, 첫번째 경쟁 윈도우 $[0, CW_0^{(rt)}]$에서, 2개 이상의 단말이 동일한 back-off 시간을 선택한 경우에 충돌이 발생한 것으로 처리하였다. 그림 12는 1차 함수 형태의 분포 함수뿐만 아니라 2차 함수 형태의 분포 함수에 대해 각기 다른 ϵ 값을 사용할 때, 총 사용 단말의 숫자에 따른 충돌 확률에 대한 시뮬레이션 결과이다. 여기서, $CW_0^{(rt)}$는 8로 가정하였다. 이 시뮬레이션 결과에서 유추할 수 있는 바와 같이 2차 함수 형태가 일반적으로 1차 함수 형태보다도 충돌 확률 성능이 우수하며, 또한 2차 함수형태의 경우에 트래픽의 부하가 증가함에 따라서 ϵ이 적은 값을 가지는 2차 함수의 충돌 확률이 ϵ이 큰 2차 함수에 비해서 성능이 개선된다는 것을 확인할 수 있다. 즉, ϵ 값이 적을 경우 PDF의 볼록한 정도가 크게 되고, 이에 따라서 트래픽 부하가 클 때 충돌 확률이 낮아지게 된다는 것을 알 수 있다. 본 시뮬레이션 결과에서 $\epsilon = \frac{1}{32}$일 때 주어진 트래픽 부하에 따라서 성능이 우수한 것을 알 수 있다.

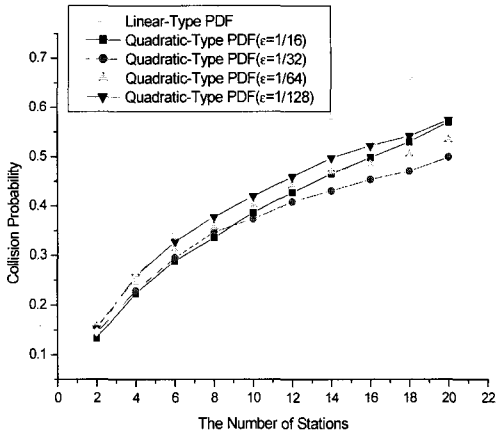


그림 12. 확률 분포 함수의 형태에 따른 충돌 확률 성능 비교 ($CW_0^{(rt)}$=8)

그림 12의 결과에 의해 제안한 분산MAC 알고리즘에 적용되는 확률 분포 함수는 $\epsilon = \frac{1}{32}$을 갖는 2차 함수 PDF를 이용해서 시뮬레이션을 하였고, 그에 따른 성능을 분석하였다.

Ⅳ. 시뮬레이션에 따른 성능 분석

1. 트래픽 모델링과 시스템 매개변수

본 절에서는 제안된 방식에 대한 성능을 분석하기 위한 시뮬레이션에서 고려된 트래픽 모델과 관련 시스템 특성을 기술한다. 시뮬레이션에서 고려되는 트래픽은 크게 실시간과 비실시간 트래픽으로 구분되고 각각 다음과 같이 모델링 된다.

실시간 트래픽은 활성 상태(active state)와 비활성 상태(inactive state)를 갖는 2-State Markov process로 모델링된다. 활성 상태에서 64kbps의 고정 전송률로 패킷이 발생되고, 따라서 패킷의 도착 시간 간격은 일정하다고 가정한다. 또한, 활성 상태와 비활성 상태에 머무는 시간은 평균이 각각 값1 sec와

1.35 sec을 가지고 지수 분포(exponential distribution)를 따른다고 가정한다. 지연 바운드를 갖는 실시간 트래픽을 고려하여 D_{max}= 25ms로 설정하고, 이 보다 더 큰 지연을 겪을 때 단말 내의 버퍼에서 패킷을 폐기하므로 이는 패킷 손실률로 반영된다. 한편, 비실시간 트래픽은 평균 도착률이 0.1인 포아송 프로세스로 모델링하고, 이때 모든 패킷은 1,023 바이트의 고정 길이를 갖는다고 가정한다. 비실시간 트래픽에 대해서는 지연 시간에 대한 특별한 요구 사항은 고려하지 않는다.

본 분석에서 사용된 시뮬레이션은 OPNET Technology Inc.의 OPNET Modeler를 이용하였으며, 이때 사용된 시스템 파라미터를 요약하면 표 2와 같다. 또한, 채널 상에서 오류가 발생하지 않는 이상적인 상황을 가정하였다.

표 2. 시뮬레이션에 사용된 시스템 파라미터

Parameter	**Value**
채널 전송률Value	1 Mbps
지연 바운드(D_{max})	25 ms
Slot time	20 us
SIFS (Short InterFrame Space)	10 us
PIFS (PCF InterFrame Space)	30 us
DIFS (DCF InterFrame Space)	50 us
CWRT_min	31
CWRT_max(Fixed offset)	255
CWNRT_min	31
CWNRT_max	1023
L	0

2. 성능 분석

다음 분석에서 FO 방식은 고정 offset (= CW_{RT_max}) 값을 사용하면서 경쟁 윈도우 값에 따라 실시간과 비실시간 단말에 대해서 채널 액세스 구간을 분리하는 방식만을 적용하는 경우를 지칭한다. 한편, FO-DPC 방식은 고정 offset (= CW_{RT_max}) 값을 이용해서 경쟁 윈도우를 분리하고, 추가적으로 실시간 단말의 잔여 수명에 따른 동적 우선권을 적용하는 경우를 지칭한다. 이때 CW_{RT_max} 값은 단말의 수(=N_{STA})에 따라 임의로 정해진 다음 관계식을 이용해서 결정하기로 한다.

$$CW_{RTmax}=\begin{cases}2^{5+i}, & 3i+1\le N_{STA}\le 3(i+1), 0\le i\le 3\\ 2^{9}, & 3i+1\le N_{STA}, i\ge 4\end{cases} \quad (9)$$

FO-DPC (optimal) 방식은 FO-DPC 방식의 시뮬레이션 결과를 바탕으로 트래픽 부하에 따른 최적의 CWRT_max 값을 도출해서 offset 값을 고정하여 적용한 경우를 지칭한다. 이 경우, 최적의 CWRT_max 값은 실시간 트래픽에 대한 평균 패킷 손실 확률을 정해진 목표 값(=0.01)으로 유지하면서 비실시간 트래픽의 평균 지연 시간을 최소화하는 값으로 설정하였다. 따라서, 각 트래픽 부하에서 최적의 고정 offset을 이용한 결과라는 점에서 고정 offset을 사용하는 방식을 통해 도출할 수 있는 최적의 결과라고 할 수 있다. 그리고, AO 방식은 적응적으로 offset 설정 메커니즘을 사용하는 방식을 말하고, AO-DPC 방식은 적응적인 offset 설정 메커니즘과 실시간 단말의 잔여 수명을 이용한 동적 우선권 부여 방안을 모두 사용하는 방식을 지칭한다.

그림 13과 14에서는 비실시간 단말을 5개로 고정하고(STA_{NRT}=5), 실시간 단말 수 (STA_{RT})를 증가시키면서 수행한 패킷 손실 확률과 평균 접속 지연시간을 분석하였다. 실시간 트래픽의 경우에는 지연 바운드에 따라 패킷 폐기가 발생하며, 이에 따라 허용 가능한 패킷 손실 확률을 0.01로 설정하였다. 이 때 DCF의 경우에는 실시간 단말을 최대 6개 정도까지 수용할 수 있는 것으로 확인할 수 있다. 즉, 6개 이상의 비실시간 단말이 존재할 경우에는 패킷 손실 확률 목표값을 더 이상 만족할 수 없게 된다. 또한, 시뮬레이션 결과로부터 실시간과 비실시간 단말의 채널 액세스 구간만을 분리하는 것만으로는 큰 용량(capacity)의 증가를 가져 올 수 없다는 것을 알 수 있다(DCF와 비교할 때 약 1개의 실시간 단말을 추가를 수용할 수 있음). 한편, FO 방식과 비교해 볼 때 FO-DPC 방식을 통해서 4개의 실시간 단말을 더 수용할 수 있다는 것을 알 수 있다. 또한, FO-DPC (optimal) 방식은 트래픽 부하에 따라서 계속 같은 고정 offset 값을 두는 방안과 비교해 볼 때 목표 패킷 손실율에 근접하면서 용량을 더욱 증가시킬 수 있다는 것을 볼 수 있다. 또한, 제안된 방식인 AO-DPC 방식은 FO-DPC (optimal) 방식과 비슷하게 고정 offset을 설정하는 방식에 비해서 3개 이상의 단말을 더 수용할 수 있다는 것을 알 수 있다. AO-DPC 방식과 비교해 볼 때 FO-DPC 방식은 실시간 단말이 8개 이하일 때 불필요한 패킷 손실 확률 감소가 발생한다는 것을 볼 수 있다. 또한, 실시간 단말 수가 8개 이상일 때에는 부하에 따라 적응적으로 offset 값을 더 이상 늘리지 못하는

관계로 패킷 손실 확률의 증가가 커진다는 것을 확인할 수 있다. AO 방식은 DCF 모듈에 RTS-CTS 교환 과정을 통해서 동적으로 offset을 설정하는 메커니즘만을 부가한 방식이다. 시뮬레이션 결과에서 AO 방식을 적용함으로써 10개의 실시간 단말을 수용할 수 있다. 따라서, 이 방식을 통해서 기존의 DCF를 단순한 모듈 변경을 통해 쉽게 구현할 수 있고 실시간 단말의 용량을 크게 증가시킬 수 있다는 것을 확인할 수 있다. 또한, 비실시간 단말의 지연 시간 성능을 최대한 보장하면서 실시간 단말의 용량을 늘리기 위해서는 주어진 트래픽 부하에서 10개까지는 AO 방식으로 동작하고, 그 이상의 트래픽 부하에서는 추가적인 용량 확보를 위해서 DPC 방식을 추가한 AO-DPC 방식을 적용함으로써 대역 효율성을 극대화할 수 있을 것이다.

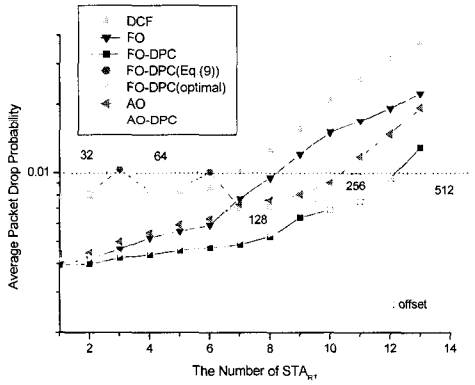


그림 13. 실시간 단말의 평균 패킷 손실 확률
(# of STA_{NRT} = 5)

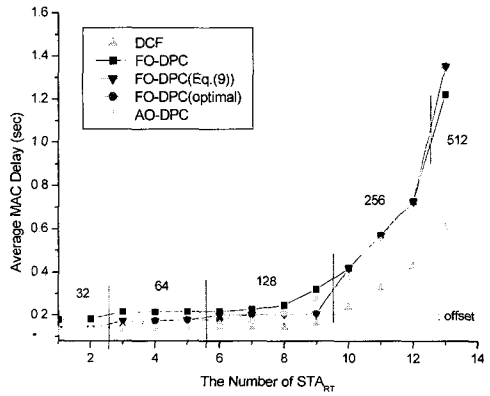


그림 14. 비실시간 단말의 평균 MAC 지연 시간
(# of STA_{NRT} = 5)

그림 14에서는 비실시간 단말이 겪는 평균 지연을 나타낸다. DCF의 경우에는 그림 13에서 본 실시간 단말의 용량 감소에 비례하여 비실시간 트래픽의 평균 지연 성능에서 상대적인 이득을 얻게 된다. 즉, 이는 실시간 트래픽과 비실시간 트래픽간의 서비스 차별화 및 비실시간 트래픽의 동적 우선권을 전혀 고려하지 않기 때문에 발생하는 현상이다. 하지만, 제안하는 접근 방식의 경우에는 비실시간 트래픽의 지연 시간을 최소한으로 증가시키면서 실시간 트래픽의 용량을 극대화할 수 있다는 것을 알 수 있다.
그림 14에서 offset 값을 CW_{RT_max}로 고정시킴으로 인해서 FO-DPC 방식에서 STA_{RT}가 1개일 때에도 비실시간 단말의 지연 시간이 DCF 방안에 비해서 60ms 정도 증가되는 것을 볼 수 있다. 또한, 실시간 단말의 수가 증가함에 따라서 제안된 방안의 평균 지연 시간의 기울기가 DCF 보다 커지는 것을 확인할 수 있다. 그 이유는 비실시간 단말의 back-off 시간 동안 실시간 단말들의 패킷 전송이 증가하므로 back-off 시간이 줄지 않는(freezing) 구간이 증가되고, 이로 인해서 비실시간 단말의 back-off 시간이 0으로 상쇄되는 시간이 기하 급수적으로 증가되기 때문이다. FO-DPC 방식에서는 offset 값을 최대 CW_{RT_max} 값으로 설정함으로써 비실시간 단말의 지연 시간이 불필요하게 증가된다는 것을 알 수 있다. 이와 달리 AO-DPC 방식에서는 트래픽의 부하에 따라 offset 값을 적응적으로 제어함으로써 실시간 단말의 패킷 손실 성능 목표를 만족하고, 불필요한 패킷 손실 확률 성능 증가를 제거함으로써 비실시간 단말의 패킷 지연시간 성능을 향상 시킬 수 있음을 알 수 있다. 예를 들어, STA_{RT}이 1개일 때는 DCF와 비교할 때 15ms의 지연 시간 증가만을 가져온다. 즉, AO-DPC 방식을 통해서 패킷 손실 확률을 일정 목표값인 0.01 이하로 유지하면서 비실시간단말의 지연 시간을 최소화할 수 있다는 것을 볼 수 있다.

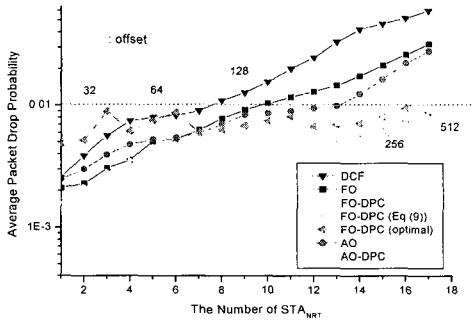


그림 15. 실시간 단말의 평균 패킷 손실 확률
(# of START = 5)

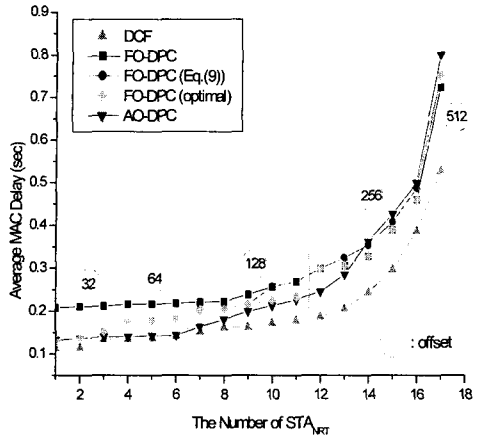


그림 16. 비실시간 단말의 평균 MAC 지연 시간(# of START = 5)

그림 15와 16에서는 실시간 단말을 5개로 고정하고, 비실시간 단말의 수를 증가시키면서 수행한 시뮬레이션 결과를 나타낸다. 이 때, DCF의 경우에는 실시간 단말을 7개 정도까지 수용할 수 있는 것을 알 수 있다. 반면, FO 방식에서는 9개, FO-DPC 방식에서는 16개까지의 비실시간 단말을 수용할 수 있다는 것을 알 수 있다. 이는, DCF 방안과 비교해 볼 때 용량(capacity) 측면에서 2 배의 성능 향상에 해당한다. 또한, AO 방식과 AO-DPC 방식은 고정 offset을 사용하는 방식보다 더 큰 용량 증가를 가져온다는 것을 확인할 수 있다. AO 방식은 단순한 모듈 변경으로 DCF 방식보다 5개의 실시간 단말을 더 수용할 수 있다. 그림 16은 비실시간 단말이 겪는 평균 지연을 나타낸다. 그림 14와 마찬가지로 STA_{NRT_hidden}을 고려해서 CW_{RT_max} 값으로 고정시킴으로 인해서 제안된 방안에서 실시간 단말이 1개일 때에도 지연 시간이 100ms 정도 증가되는 것을 볼 수 있다. 하지만, STA_{RT}를 고정 시켰을 경우 비실시간 단말간의 경쟁에 의한 지연이 크게 영향을 미치기 때문에 DCF와 FO-DPC 방식에서의 평균 지연 시간의 기울기가 그림 14와 달리 거의 유사한 형태를 보임을 알 수 있다. 또한, RTS-CTS 교환을 통해서 offset 값을 적응적으로 유지하는 AO-DPC 방식의 결과를 제시하였다. 동적인 방식을 사용했을 경우에 DCF 방식에서의 비실시간 단말의 지연 시간과 비교했을 때 12ms 정도의 지연 시간 저하만을 가지고 실시간 단말의 용량을 크게 증가시킬 수 있다는 것을 확인할 수 있다. 또한, 패킷 손실 확률의 목표값을 계속 유지하는 댓가로서 비실시간 단말의 지연 시간이 점차 증가되는 것을 볼 수 있다.

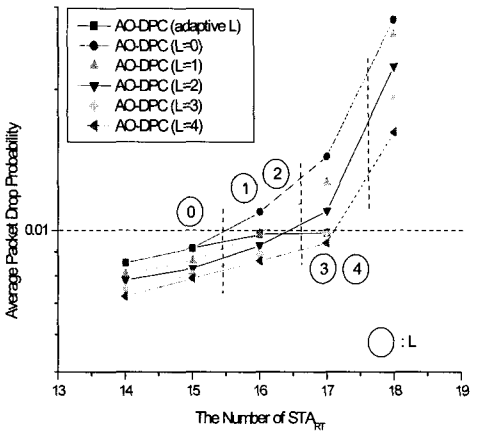


그림 17. L 값에 따른 실시간 단말의 평균 패킷 손실 확률 (# of STA_{NRT} = 5)

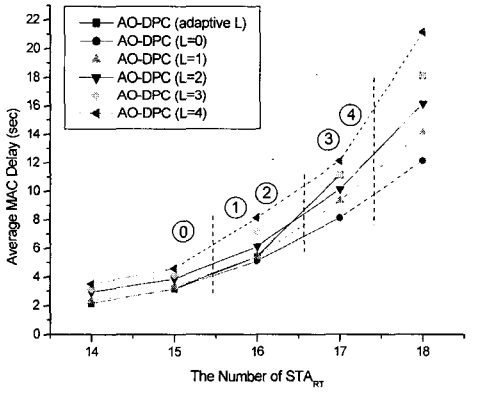


그림 18. L 값에 따른 비실시간 단말의 평균 MAC 지연 시간 (# of STA_{NRT} = 5)

그림 17과 그림 18은 실시간 단말을 증가시키면서 L 값에 따른 평균 손실 확률과 평균 MAC 지연 시간에 대한 시뮬레이션 결과를 제시한다. L 값이 증가함에 따라서 비실시간 단말은 L 번의 재시도 횟수까지 offset 값을 적용하게 되고, 이로 인해서 비실시간 단말의 지연 시간이 증가한다는 것을 확인할 수 있다. 또한, 상대적인 이득으로 실시간 단말의 패킷 손실 확률이 감소한다는 것도 확인할 수 있다. 하지만, 패킷 손실 확률 측면의 성능 결과를 보면 실시간 단말의 트래픽 부하가 매우 크게 증가

함으로 인해서 겪는 패킷 손실 확률의 증가가 크기 때문에 비실시간 단말의 offset 값의 적용 횟수를 증가시켜도 AO-DPC (adaptive L) 방식에서의 실시간 단말에 대한 용량 증가가 크지 않다는 것을 확인할 수 있다. 즉, L=0을 적용시킨 결과와 비교해 볼 때, L=1,2일 때 실시간 단말의 용량 증가가 1개에 불과하고, 또한 L=3,4일 때 증가되는 실시간 단말의 용량도 1개 밖에 되지 않는다는 것을 확인할 수 있다.

그림 19와 그림 20은 비실시간 단말의 증가에 따른 L 값에 따른 평균 손실 확률과 평균 MAC 지연 시간에 대한 시뮬레이션 결과이다. 그림 17과 그림 18의 시뮬레이션 결과와는 달리 비실시간 단말의 증가에 따라offset 값이 직접적으로 성능에 영향을 미친다는 것을 확인할 수 있다. 즉, 비실시간 단말에 offset 값 적용 횟수를 증가시킴에 따라서 실시간 단말의 용량이 계속적으로 증가하게 된다. 따라서, 본 방식을 기반 구조망에서 적용할 때 AP에 의해서 트래픽 부하에 따라 L 값을 적응적으로 제어하는 DPC (adaptive L) 방식을 적용함으로써 실시간 단말의 용량을 크게 증가시킬 수 있을 것으로 본다. 시뮬레이션 결과에서 STA_{RT}가 5개, STA_{NRT}가 19개인 경우 L=0일 때 평균 패킷 손실 확률이 목표값인 0.01을 넘게 되므로, 이 때 AP에 의해서 비실시간 단말의 L 값을 1로 설정하도록 명령함으로써 전체적인 용량을 증가시키는 것이 가능하다. 물론, 실시간 단말의 용량 증가에 따라서 비실시간 단말이 겪는 평균 MAC 지연 시간은 L 값이 증가함에 따라서 기하 급수적으로 증가하게 된다.

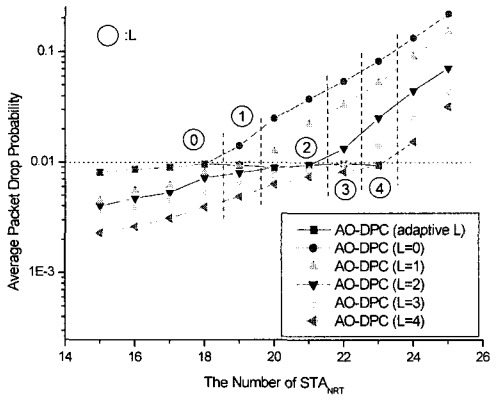


그림 19. L 값에 따른 실시간 단말의 평균 패킷 손실 확률 (# of STA_{RT} = 5)

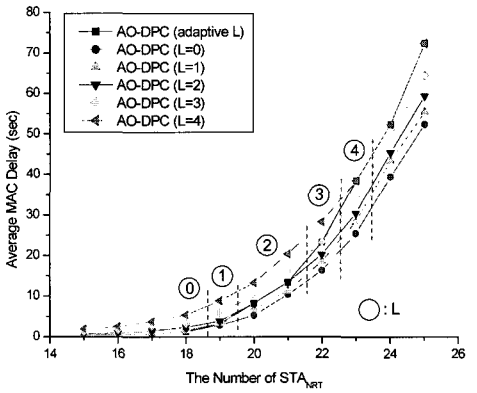


그림 20. L 값에 따른 비실시간 단말의 평균 MAC 지연 시간 (# of STA_{RT} = 5)

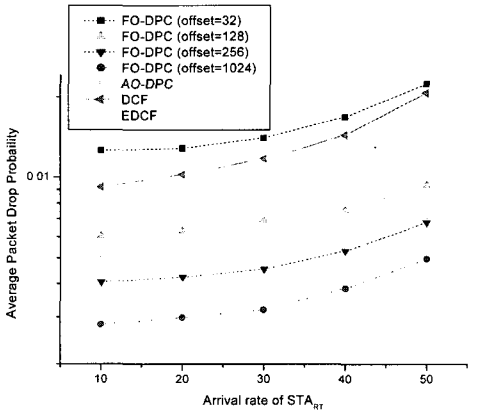


그림 21. 평균 패킷 도착율에 따른 실시간 단말의 평균 패킷 손실 확률 (# of STA_{RT}=5, # of STA_{NRT}=5)

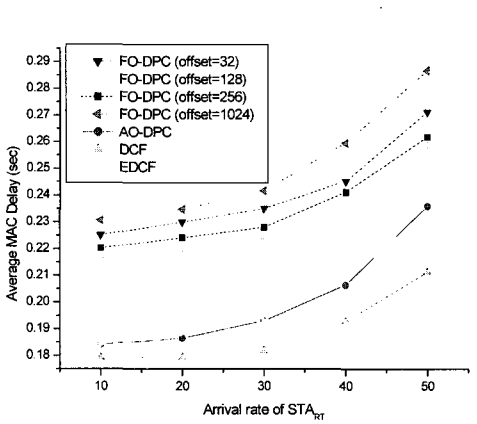


그림 22. 평균 패킷 도착율에 따른 비실시간 단말의 평균 MAC 지연 시간 (# of STA_{RT}=5, # of STA_{NRT}=5)

그림 21와 22은 평균 패킷 도착율에 따른 실시간 단말의 평균 패킷 손실 확률과 비실시간 단말에 대한 MAC 지연 성능에 대한 시뮬레이션 결과를 나타낸다. 여기서, EDCF는 실시간과 비실시간의 2가지 서비스 클래스가 있다고 가정하였고, 실시간 단말의 InterFrame Space인 AIFS[1]을 $SIFS + aSlotTime$으로, 비실시간 단말의 InterFrame Space인 AIFS[2]를 $SIFS + 2 \times aSlotTime$으로 설정하였다. 또한, 실시간 단말은 $CW_{min}[1]=32$, $CW_{max}[1]=1023$으로 두었고, 비실시간 단말은 $CW_{min}[2]=255$, $CW_{max}[2]=1023$으로 가정하였다.

그림 21은 실시간 단말의 평균 패킷 손실 확률을 나타낸다. 시뮬레이션 결과에서 보는 바와 같이 AO-DPC 방식은 FO-DPC 방식과 비교할 때 패킷 도착율이 증가함에 따라서 패킷 손실 확률 성능이 점차 개선되는 형태를 나타낸다. EDCF는 실시간 단말에 우선권을 부여함으로써 DCF에 비해서 실시간 단말의 패킷 손실 확률 성능의 개선을 보이기는 하지만, 제안된 방식과 같이 실시간 단말이 트래픽 전송을 요구할 경우에 절대적인 우선권을 부여하지 못하기 때문에 제안된 방식과 비교해 볼 때 그 성능이 떨어지는 것을 확인할 수 있다. 제안된 방식과는 달리 EDCF 메커니즘은 실시간 단말과 비실시간 단말이 선택할 수 있는 back-off 슬롯 시간의 범위가 서로 중첩되기 때문에 실시간 단말에 우선권을 부여하지만 그에 따른 성능 향상이 크지 않다고 분석된다. 또한, FO-DPC 방식은 해당 트래픽 부하에서 offset 값의 증가함에 따라서 실시간 단말의 평균 패킷 손실 확률은 증가하는 형태를 나타낸다. 트래픽 부하에 따라서 적응적으로 offset 값을 설정하지 못하기 때문에 offset 값이 크게 설정된 FO-DPC 방식의 패킷 손실 확률이 필요 이상으로 매우 낮게 나타나고, 그에 반해 offset 값이 작게 설정된 경우는 패킷 손실 확률이 크게 증가하게 된다.

그림 22는 비실시간 단말의 평균 MAC 지연 시간을 나타낸다. FO-DPC 방식은 고정 offset 값을 둠으로써 offset 값을 크게 설정할 때 지연 시간 성능이 가장 떨어짐을 확인할 수 있다. 또한, offset 값이 32인 경우 실시간 단말의 최대 경쟁 윈도우 크기가 32로 설정되고 이로 인해서 비실시간 단말의 채널 액세스까지 방해하기 때문에 비실시간 단말의 평균 지연 시간 성능이 매우 떨어진다. AO-DPC 방식은 트래픽 부하가 적을 때 offset 값이 적게 설정되는 관계로 EDCF 방식에 비해서 비실시간 단말의 평균 MAC 지연 시간이 작게 되고, 점차 실시간 트래픽의 부하가 증가함에 따라 offset 값을 증가시키므로 지연 시간이 크게 증가하게 된다는 것을 확인할 수 있다. 제안된 AO-DPC 방식은 실시간 단말의 트래픽 부하에 의해서 유동적으로 offset 값을 조정할 수 있으나, 이에 반해 EDCF는 부하에 상관없이 실시간 서비스 단말이 항상 우선권이 높은 InterFrame Space와 경쟁 윈도우 값을 유지하게 된다. 이러한 이유로 인해서 EDCF는 DCF와 비교할 때 항상 일정한 지연 시간의 증가분을 가지게 되나, 제안된 AO-DPC 방식은 주어진 트래픽 부하에 따라 실시간 서비스의 품질을 보장하면서 비실시간 서비스의 지
연 성능에 대해 최적의 trade-off를 실현하게 된다.

V. 결 론

본 논문에서는 IEEE 802.11 규격 Wireless LAN이 Ad Hoc 모드로 비실시간 및 실시간 트래픽을 동시에 지원할 때, 지연 바운드를 갖는 실시간 트래픽의 품질을 보장하면서 용량을 향상 시킬 수 있는 차별화 서비스의 구체적인 구현 방안을 제시하였다. 즉, 기존 DCF 규격을 변형하지 않고 CSMA/CA 기반의 경쟁 방식을 통해서도 비실시간 트래픽과의 차별화가 가능하고, 또한 지연 바운드를 고려한 동적 우선권 할당이 가능하도록 하였다. 본 분석 결과에 따르면, 고정offset 값을 사용한 방식에서 기존 DCF와 비교할 때 실시간 단말의 용량을 2배까지 향상할 수 있었고, RTS와 CTS 메시지 교환 과정의 응용을 이용하여 offset 값을 적응적으로 제어함으로써 비실시간 단말의 평균 MAC 지연 시간을 현격하게 줄이면서 추가적인 용량 증가를 가져올 수 있음을 확인하였다.

제안된 방식은 기존의 IEEE 802.11 규격 Ad-hoc MAC 프로토콜의DCF 방식에서 back-off 슬롯 설정 방식와 재전송 확률에 대한 분포 함수만을 변형함으로써 실시간 트래픽에 대한 QoS를 보장할 수 있는 MAC 프로토콜이다. 즉, 사용자별로 다른 경쟁 윈도우 값을 할당하는 IEEE 802.11e의 Enhanced DCF 방식과는 달리 기존 DCF 방식을 그대로 사용하면서도 실시간 트래픽에 대해 절대적인 우선권을 보장할 수 있도록 하고, 또한 실시간 트래픽의 지연 바운드를 보장할 수 있는 것이 특징이다. 따라서, 제안된 방식은 DCF 기반의 현재 상용 시스템과 호

환성을 유지하면서 Infrastructure 모드뿐만 아니라 다중 홉 라우팅까지 고려한 Ad Hoc 모드에서도 실시간 및 비실시간 서비스를 효율적으로 통합할 수 있는 방안이 될 것이다.

참 고 문 헌

[1] ISO/IEC and IEEE Draft International Standards, Part 11: Wireless LAN Medium Access Control (MAC) and Physical (PHY) Layer Specifications, ISO/IEC 8802-11, IEEE P802.11/D10, Jan. 1999.

[2] ISO/IEC and IEEE Draft International Standards, Part 11: Wireless LAN Medium Access Control (MAC) and Physical (PHY) Layer Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), IEEE P802.11e/D2.0, Nov. 2001.

[3] N. H. Vaidya, P. Bahl and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," Proceedings of ACM MOBICOM'00, 2000.

[4] S. T. Sheu and T. F. Sheu, "A Bandwidth Allocation/Sharing/Extension Protocol for Multimedia Over IEEE 802.11 Ad Hoc Wireless LANs," IEEE JOURNEL ON SELECTED AREAS IN COMMUNICATIONS, Vol. 19, No. 10, Oct. 2001.

[5] H. Woesner, J. Weinmiller, J. P. Ebert and A. Wolisz, "Modified Backoff Algorithms for DFWMAC's Distributed Coordination Function," Proceedings of 2nd ITG Fachtagung Mobile Kommunikation '95, Sep. 1995.

[6] I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11," Proceedings of IEEE INFOCOM 2001, Apr. 2001.

[7] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi and E. Knightly, "Distributed Multi-Hop scheduling and medium access with delay and throughput constraints," Proceedings of MOBICOM 2001, Jul. 2001.

김 원 수(Won Soo Kim)

2001年 2月 : 고려대학교 전파공학과 졸업

2003年 2月 : 고려대학교 전파공학과 석사

2003年 2月~현재 : 삼성종합기술원 연구원

<관심분야> Wireless MAC protocol, QoS

강 충 구(Chung Gu Kang)

1987年 6月: Univ. of California (San Diego), 전자공학과 학사

1989年 6月: Univ. of California (Irvine), 전자 및 컴퓨터 공학과 석사 1993年3月: Univ. of California (Irvine), 전자 및 컴퓨터 공학과 박사

경 력

1992年7月 - 1992年5月: Aerospace Corp. 연구원

1993年4月 - 1994年2月: Rockwell International 연구원

1994年3月 - 현재: 고려대학교 전파통신공학과 조교수/부교수/정교수

<관심 분야>

광대역 무선 전송 기술 및 매체접근제어 프로토콜 설계/구현, 시스템 모델링 및 성능 분석, Wireless PAN/LAN/MAN, 4세대 이동통신 기술