

000

000

000

IEEE 802.11ax WiFi 망에서 Network Slicing 구현 방안에 대한 연구

2021. 1

000

Abstract

네트워크 슬라이싱 기술은 애플리케이션 서비스 혹은 사용자에 따라 네트워크 서비스의 품질을 보장하는 최신 네트워크 기술이다. 최신 5G 네트워크는 무선 상에서도 다중 채널의 공유 및 네트워크 가상화 기술 그리고 Time-Sensitive Network 기술과 결합해서, 서비스 혹은 사용자 별 차별화된 속도, 대역폭 등을 제공하는 네트워크 슬라이싱 기술이 구현되고 있다. 하지만 사용되는 근거리 무선 네트워크인 IEEE 802.11 기반의 기반한 무선 랜 와이파이는 다양한 IoT의 장비 및 모바일 기기를 위해 사용되고 있음에도 네트워크 슬라이싱을 함께 구현하기는 용이하지 않다. 본 논문에서는 무선 랜 와이파이에 네트워크 슬라이싱을 도입하기 위해 특히 시간적 결정성을 확보하기 위해 IEEE 802.11e의 Enhanced Distributed Coordination Acces (EDCA, 향상된 분산 채널 접근)의 최악의 경우 분석하고 그 한계를 살펴보며 그 해결 방법 제안한다.

Network slicing technology is the latest network technology that guarantees the quality of network services according to application services or users. In 5G networks, network slicing that provides differentiated speed and bandwidth for each service or user is being implemented in combination with multi-channel sharing and network virtualization technology and Time-Sensitive Network technology even over wireless. However, although the IEEE 802.11-based wireless LAN Wi-Fi is a mostly popularly used short-range wireless network for various IoT devices and mobile devices, it is not easy to implement network slicing together. In this paper, we presents the analysis results of the worst case of IEEE 802.11e Enhanced Distributed Coordination Acces and provides a solution to guarantee time-determinism in wireless LAN Wi-Fi networks.

차 례

| | |
|--|--------|
| 제 1 장 서론 | v |
| 제 2 장 관련연구 | viii |
| 제 3 장 배경 | ix |
| 제 1 절 EDCA의 QoS | x |
| 제 4 장 실시간 시스템을 위한 와이파이 QoS 한계 분석 | xiv |
| 제 1 절 시스템 모델 및 문제정의 | xiv |
| 제 2 절 최악의 경우 분석 | xvi |
| 2.1 채널 대기 지연 최악의 경우 | xvii |
| 제 3 절 AP 기반의 실시간 시스템 용 와이파이 슬라이싱 | xx |
| 제 4 절 실험 및 결과 | xxi |
| 4.1 실험셋팅 1 | xxii |
| 4.2 실험셋팅 2 | xxiv |
| 4.3 실험셋팅 3 | xxv |
| 제 5 장 새로운 접근방법 | xxvii |
| 제 1 절 우선순위 큐를 이용한 접근법 | xxvii |
| 제 2 절 충돌이 없는 백오프(Non-Conflict Back-Off, NCB) | xxviii |
| 2.1 VO-VI의 경우 문제의 해결 | xxviii |
| 2.2 VO-BE 경우 문제의 해결 | xxx |
| 제 3 절 VO 스트림의 최대 채널대기 지연시간 계산 | xxxi |
| 제 6 장 결론 | xxxiii |
| References | xxxiii |
| 부록 A NS3 구현 결과 | xxxiv |
| 제 1 절 NS3 기본 실행 | xxxiv |
| 1.1 NS3 소개 | xxxiv |
| 부록 B WIFI 슬라이싱 NS3 구현 | xxxvi |

그림 차례

| | | |
|-----|---|-------|
| 3.1 | 와이파이 QoS 기능의 원리 | ix |
| 3.2 | 와이파이 송수신 제어 프로그램 간의 IFS | x |
| 3.3 | 백오프 처리과정 | xi |
| 3.4 | DCF와 EDCA방식의 우선순위 부여 방식(tramarin2019real) | xii |
| 3.5 | 접근부류(Access category) 별 접근타이밍 | xiii |
| 4.1 | 본 연구개발에서의 시스템 모델 | xiv |
| 4.2 | AP가 CW(3)을 선택하고, Station C가 CW(0)을 연속적으로 선택하여, VO 프레임이 지연되는 경우, AP의 백오프시간이 줄지 않음 | xviii |
| 4.3 | AP가 최대 CW(3) 선택하고, BE 프레임이 최소 CW(0)을 연속적으로 선택하여, VO 프레임이 지연되는 경우 | xix |
| 4.4 | 제안된 기법의 AP 적용 | xx |
| 4.6 | 실험셋팅 1의 실험 결과 | xxiii |
| 4.7 | 실험셋팅 2의 실험 결과 | xxiv |
| 4.8 | 실험3-1 | xxv |
| 4.9 | 실험3-2: 우선순위에 따른 최우선 전송 비율 | xxvi |
| 5.1 | CW 조정을 통한 No-BO-Decrement 해결 | xxix |
| 5.2 | CW 조정을 통한 VO-BE 문제 해결 | xxx |

표 차례

| | | |
|-----|--|-------|
| 3.1 | EDCA Contention Window values(802.11g/a/n PHY) | xii |
| 4.1 | 심볼 정의 | xv |
| 4.2 | Queue Assignment Table Fields | xx |
| 4.3 | 실험셋팅 1의 QTA 1 | xxii |
| 4.4 | 실험셋팅 1의 QTA 2 | xxiii |
| 4.5 | 실험셋팅 2의 QTA 1 | xxiv |
| 4.6 | 실험셋팅 2의 QTA 2 | xxv |
| 4.7 | 실험셋팅 2의 QTA 3 | xxv |

제 1 장

서론

와이파이는 근거리 무선 네트워크 기술 중 가장 보편적으로 사용되는 네트워크 기술이다. 2017 1억 2,200만개의 전세계 공개 와이파이가 2022년까지 그 수가 4배 이상 증가할 것으로 예측되며, 글로벌 IP 트래픽 중 와이파이 비중이 2017년 43%에서 2022년에 51%까지 증가할 것으로 예측하고 있다.

네트워크 슬라이싱 기술은 애플리케이션, 서비스 혹은 사용자의 타입에 따라 네트워크 서비스의 품질을 보장하는 최신 네트워크 기술이다. 5G 네트워크는 네트워크 가상화 기술을 구현함으로서 서비스 타입에 따른 차별화된 무선 네트워크 서비스를 보장한다. 네트워크 슬라이싱은 기본적으로 네트워크 가상화에 기반을 둔다. 네트워크 가상화는 네트워크 자원을 분배(partitioning) 혹은 통합(combining)하는 모든 형태를 가리키는데, 네트워크 사용자는 분배 혹은 통합된 자원을 이용하면서, 독점적으로 혹은 분할된 네트워크를 사용한다. 이렇게 분할 혹은 통합되는 자원으로서는 기본적으로 각 노드와 링크 혹은 토폴로지가 될 수 있다. 노드와 링크의 가상화는 자원을 분배/통합/추상화(abstraction)를 모두 포함하고 있다. 이러한 네트워크 슬라이싱은 1) 서비스 관점, 2) 사용자 관점, 3) 애플리케이션 관점에서 이루어질 수 있다. 즉 데이터 서비스, 음성 서비스, 등 서비스 공급자가 제공하는 서비스에 따라, 혹은 사용자의 등급이나 사용 요금제에 따라, 혹은 애플리케이션의 특성, 예를 들면 비디오 스트리밍, 긴급한 위험 경보 방송, V2C (Video to Cloud) 애플리케이션에 따라 하드웨어, 소프트웨어 혹은 대역폭 등을 할당하고 서비스의 품질(QoS)를 차별화한다. 현재 5G에서는 네트워크 슬라이싱을 통해서, 서비스 품질 보증, 서비스 공급자의 네트워크 기반시설의 공유 등을 달성하고 있다.

와이파이 상의 네트워크 슬라이싱인 와이파이 슬라이싱(WIFI Slicing)은 IEEE 802.11xx를 기반으로 한 무선 상에서의 네트워크 자원 분배(partitioning)을 가리킨다. 5G 네트워크 경우, 무선 다중 채널의 공유 및 분할을 통해서 슬라이싱 기술이 구현되는 반면, 와이파이 네트워크은 다중 채널 공유 기술과 같은 채널 공유 기술 사용이 불가능하다. 즉 와이파이 네트워크에서는 하나의 채널을 분할하거나 공유하지 않기 때문에 가상화를 통한 대역폭 분할이 불가능하다.

와이파이 슬라이싱 역시 네트워크 슬라이싱 기술을 기반한다. 다만 액세스 포인트(Access Point, AP)와 단말 기기 간 무선 매체의 네트워크 가상화가 5G와 구분되는 기술이라 할 수 있다. 즉 RAN(Radio Access Network)을 가상화 하는 기술 혹은 AP 서비스를 슬라이스로 만들거나, 그 슬라이스를 단말에 할당하거나, 삭제하는 부차적인 기술이 개발되어야 한다.

실시간 시스템은 모든 컴포넌트의 행위가 시간적 의미에서 결정적이어야 한다. 실시간 시스템은 빠른 시스템이 아니라 시간적으로 결정적 시스템이라 할 수 있다. 따라서 항상 최악의 경우를 고려하여 실시간 시스템을 분석하고 설계한다. 최근 실시간 시스템을 위한 다양한 네트워크가 개발되고 있다. 예를 들어 Time-Sensitive Network(TSN)은 모든 스트림의 프레임에 대해서 목적지로 상의 패스의 모든 위치에 각 프레임의 전송 시간을 정해 놓고 스케줄링함으로 모든 스트림의 시간적 결정성을 보장한다. 하지만 TSN은 현재 연구개발되는 기술이고 많은 부분에서 표준화가 필요하며, 와이파이 만큼 보급되는데 많은 시간이 걸릴 것으로 예상된다. 따라서 본 연구개발에서는 현재 가장 보편적으로 사용되는 와이파이 기술에 시간적 결정을 개선함으로 실시간 시스템 용 와이파이 슬라이싱 기술을 제공하자 한다. 특히 이 연구개발에서는 액세스포인트(Access Point, 이하 AP)를 기반으로, 다운로드 스트림을 위한 실시간 시스템 용 와이파이 슬라이싱 기술을 제안하고, 그 설계와 구현을 제안한다.

이 연구개발에서는 IEEE 802.11의 QoS(Quality of Service) 기능을 활성화하여, 4가지 우선순위-음성전송(Voice transmission, 이하 VO), 비디오전송(Video transmission, 이하 VI), 최선전송(Best-effort transmission, 이하 BE), 배경전송(Background transmission, 이하 BG)-를 활용하고, 프레임의 지연 및 충돌 방지를 위한 개선된 기술을 제안한다. 와이파이의 QoS 우선순위는 각 스테이션이 경쟁원도우(Contention Window 이하 CW)를 기반으로 각 채널을 점유의 가능성을 달리함으로써, 높은 우선순위를 사용하는 스테이션에게 보다 많은 채널 점유도를 제공한다. 본 보고서에서는 최상의 우선순위를 가진 스트림의 시간적 결정성의 한계를 분석하고, 다양한 최악의 경우를 분석한다. 또한 이러한 분석을 기반으로 VO 우선순위를 가진 스테이션의 채널 사용의 시간적 결정성을 향상시키는 두 가지 테크닉을 제공한다.

첫째로, 본 연구개발은 최상의 우선순위 큐(즉 VO큐)의 우선순위 큐의 개선 방법을 제안한다. 이를 통해서 실시간 시스템의 주기적 스트림 간에 우선순위가 존재 할 때, 높은 우선순위를 가진 스트림이 더 적은 채널 대기 지연을 갖도록 설계한다. 둘째로, 본 연구개발에서는 VO우선순위와 나머지 우선순위(VI-BE-BK)의 스트림간의 백오프 충돌로 인한 전송 지연 이분법적으로 해결하고자 한다. 즉 VO 스트림 그룹과 VI-BE-BK 스트림 그룹을 나누어 이 두 그룹 간의 충돌을 이분법적으로 해결하여 두 그룹 간의 충돌을 제거하는 방식을 새롭게 제안한다. 그리고 VI-BE-BK 스트림 간의 백오프 종료시의 충돌은 기존의 전통적인 백오프 종료 충돌 방식을 활용하여 자원 활용도를 극대화한다. 이 방법은 기존의 방법과 마찬가지로 랜덤하게 선택되는 CW 값을 조정하여 채널 충돌 방지하고 이를 통해 최상의 우선순위 큐를 사용하는 스트림의 시간적 결정성을 향상시킨다.

본 연구개발의 기여는 다음과 같다. 첫째, 본 연구개발에서는 기존 확률적인 분석 아닌 실시간 시스템 관점에서 공유 채널을 독점적으로 사용하는 와이파이 채널 점유의 최악의 경우를 분석하여 실시간 시스템의 시간적 결정성 확보할 수 있는 이론적인 토대를 제공한다. 둘째, 와이파이 QoS의 우선순위를 실시간 스트림의 슬라이스로 사용할 때의 한계를 실험적으로 제공하고, 이 분석을 토대로 최상의 우선순위를 사용하는 스트림의 시간적 결정성을 보장하는 새로운 접근을 제공한다.

이어지는 각 장의 구성은 다음과 같다. 본 보고서 2장에서는 본 연구개발의 관련 연구개발을 소개한다. 3에서는 본 연구개발의 기반이 된 IEEE 802.11의 QoS의 핵심 원리인 백오프에 기반한 DCF와 DCF를 확장한 EDCA를 통해서 최악의 경우를 분석할 수 있는 기본적인 지식을 전달한다. 4장에서는 IEEE 802.11e QoS를 EDCA를 활용할 때, VO를 사용하는 주기적 스트림이 가질 수 있는 최악의

경우를 분석하는 프레임워크와 현재의 기술 상의 최악의 경우를 제한할 수 없는 원인을 밝힌다. 또한 다양한 실험을 통해서 실제 발생 가능한 다양한 시나리오 안에서 IEEE 802.11e QoS의 EDCA 가지는 한계를 실험적으로 밝힌다. 5장에서는 앞서 4장에서 고려한 최악의 경우를 피하는 알고리즘과 실시간 주기적 스트림 간의 우선순위가 존재할 때, 우선순위를 가진 주기적 스트림의 반응 시간 즉 전송 시간을 최적화 하는 알고리즘을 제공한다. 6장에서는 이 보고서의 결론을 논한다.

제 2 장

관련연구

5G의 네트워크 슬라이싱과 마찬가지로 와이파이 슬라이싱에 대한 다양한 연구 (**Richart:2017; Richart:2019; deBast:2019**)가 중앙 집중 제어 방식 PCF 방식 중심으로 진행되었다. (**Richart:2017; Richart:2019**)는 각 슬라이스에 대한 최소 대역폭을 보장하기 위한 스케줄링 방법을 제안한다. (**Richart:2017**)는 와이파이 네트워크 자원의 동적 비례 할당을 위한 DRR(Deficit Round Robin)을 개선한 PT-DRR(Proportional Time-Deficit Round Robin)을 제안하여, 슬라이스 별 동등한 에어타임과 최소 지연 시간을 보장하는 방법을 제안하였다. (**Richart:2019**) 해당 연구에서는 와이파이의 슬라이스 별 최소 비트레이트 요구 사항을 만족하기 위해 Lyapunov Drift 이론을 기반하여 다른 슬라이스의 전송 시간 스케줄링을 이용하는 최적화 솔루션을 제안하였다. 이 접근에서는 채널의 상태와 스트림의 도착율을 확률적인 모델로 가정하여 확률적인 최적화 알고리즘으로 보장된 대역폭 슬라이싱 문제를 해결하는 방법을 제안하였다. 본 연구에서는 이러한 기본적으로 실시간 데이터가 주기적으로 도달한다. 따라서 확률적인 접근이 아닌 최악의 경우를 제한하는 실시간 시스템의 접근 방식을 취하고 있다. 각 슬라이스를 큐로 할당한 IEEE 802.11의 QoS 방식에서 각 큐를 하나의 슬라이스로 가정하여 이 중 최상의 우선순위의 최악의 경우를 제한할 수 있는 방법에 집중한다.

IEEE 802.11는 기본적으로 충돌 해결 알고리즘이 CSMA/CA 방식을 채택하고 있다. 그러한 이유 전송 매체의 충돌을 회피하여 전체적인 매체의 효율성을 높이는 많은 연구(**Seuong:2010; Nam:2011; Sukgu:2011; Kang:2015; Hyongju:2016; Nasrallah:2016**)가 진행되었다. 본 연구에서는 QoS의 큐 중 최상위 Voice 큐의 시간적 결정성을 보장할 수 있는 충돌 해결 방법을 고안하며, 기존 연구를 통해서 나머지 큐의 스트림에 대한 충동 해결을 지향한다. 이러한 이유로 위의 앞선 연구들을 보완한다고 할 수 있다.

제 3 장

배경

이 절에서는 와이파이의 QoS 기능인 Distributed coordination function (DCF) 방식의 CSMA/CA(Carrier Sense Multiple Access/Collision Avoidance)와 경쟁윈도우(Contention Window,CW)를 통한 우선순위 기반의 QoS의 기본 원리를 설명한다. 그에 더하여 DCF를 발전시킨 EDCA(Enhanced Distributed Coordination Access, 향상된 분산 채널 접근)를 설명한다.

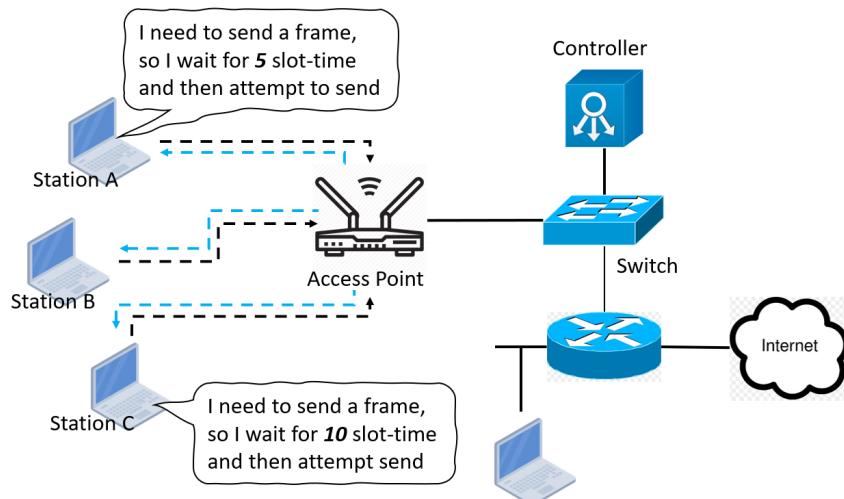


그림 3.1: 와이파이 QoS 기능의 원리

그림 3.1은 CSMA/CA를 위한 DCF 방식을 설명한다. 그림 3.1에서 AP를 비롯한 각 노드는 와이파이의 공유 채널 사용을 선점하여 네트워킹을 수행한다. 이들은 무선 채널을 사용하기 때문에, CSMA/CD처럼 전송 시 에러를 검출하는 방식을 사용하지 않고, 선제적 지연을 통해서 무선 채널 사용을 서로에게 양보한다. 각 노드는 보낼 패킷이 있으면, 즉시 보내는 것이 아니라 “먼저” 특정한 시간을 기다린 후 전송을 시도한다.

그림 3.1에서 보는 것처럼 Station A는 5초 후 그리고 Station C는 10초를 기다린 후, 전송을 시도 한다. 만약 전송 채널이 누군가에 의해 점유되고 있다면, 이전 기다리던 시간에 추가적으로 더 많은 시간을 기다린 후, 재전송을 시도하게 된다. 또한 기다리는 시간에 차등을 두어 채널의 우선순위 점

유를 모방할 수 있다. 각 스테이션은 기다리는 시간을 특정 범위 내의 숫자를 임으로 선택하고, 이를 기반으로 전송 기회를 기다린다. 우선순위가 낮은 스테이션은 우선순위가 더 높은 스테이션보다 더 큰 범위의 숫자를 선택할 수 있게 함으로써 더 오랫동안 기다릴 수 있는 경우를 늘리게 된다. 따라서 우선순위에게 높은 스테이션의 경우 더 작은 범위에서 기다리는 시간을 선택하도록 기회를 부여하고, 전송에 실패했을 때에도 다른 우선순위의 스테이션보다 더 작고 짧은 확대된 범위의 대기 시간을 임의로 선택할 수 있게 함으로써 더 많은 전송 기회가 부여한다. 따라서 우선순위가 높은 프레임의 전송이 우선순위가 낮은 프레임의 전송으로 인해 늦어질 수 있다. 또한 와이파이는 기본적으로 비선점형 데이터 전송을 지원한다. 즉 일단 전송이 시작된 프레임은 그 전송이 끝나기 전까지 다른 프레임의 전송을 허용하지 않는다. 따라서 우선순위가 낮은 프레임의 전송이 일단 시작되면, 우선순위가 높은 프레임은 이전 전송이 종료될 때까지 전송이 지연될 수 있다. 다음 절에서 보다 자세한 분석을 제공한다.

제 1 절 EDCA의 QoS

CSMA/CA는 주로 무선망에서 사용되는 전송 충돌 회피 방식이다. 반면 CSMA/CD (Collision Detection)은 이더넷처럼 전송과 수신이 서로 다른 채널에서 이루어지는 경우 충돌을 감지하는 기술이다. 와이파이에서 CSMA/CA를 채용하는 DCF는 QoS 제공에서 각 무선스테이션에게 동등한 우선순위를 부여한다. 그리고 우선순위 전송 구현을 위해서 각 노드는 데이터 전송 전, 다른 노드와의 전송 충돌을 피하기 위해 특정 범위 내의 난수의 slot-time(ST)만큼 선제적으로 대기한 후, 매체가 사용 가능한지 확인한다 (그림 3.1 참조). 이러한 선제적 대기 시간을 백오프(Back-off, 이하 백오프 혹은 BO)라 한다. 만약 다른 스테이션의 매체 선점을 감지할 경우, 각 스테이션은 특정 범위 내의 난수를 발생시키고, 특정 시간 만큼 대기한 후, 난수 시간만큼 추가적으로 대기한다.

와이파이 표준에서는 기본적인 DCF 가능 외 PCF(Point Coordination Function) 방식을 제안한다 (IEEE 802.11x에서는 반드시 구현을 요구하고 있다.). PCF는 일종의 폴링 방식으로 AP가 전송이 필요한 스테이션을 찾아 전송하는 방식으로 AP는 폴링 신호를 통해서 각 스테이션에서 전송 기회를 부여한다.

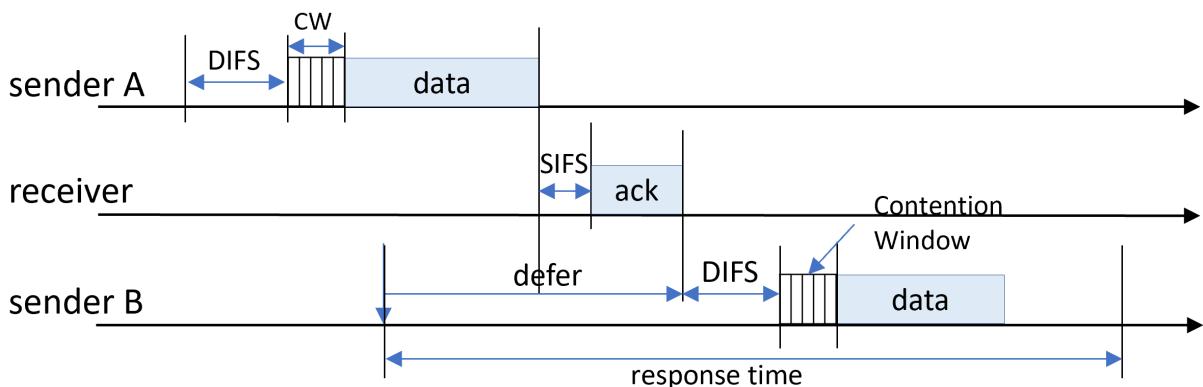


그림 3.2: 와이파이 송수신 제어 프로그램 간의 IFS

DCF는 우선순위를 고려하기 위해 차등을 가진 백오프 시 각 우선순위 간의 차별화된 경쟁원도우 (Contention Window, CW)를 사용한다. 그림 3.2에서는 DCF에서의 데이터 송수신 제어를 위한 대표적인 두 가지 IFS인 DIFS(Distributed IFS)와 SIFS(Short IFS), 그리고 채널 충돌을 회피하기

위한 경쟁윈도우(Contestation Window, CW)의 배치를 보여준다. 모든 와이파이 송신자는 채널이 누구에게도 점유되고 있지 않은 시점부터 DIFS 만큼의 채널 휴지기를 갖는다. 그 후, 주어진 우선순위에 따라 백오프를 위해 0부터 시작하여 임의의 값을 갖는 경쟁윈도우(CW) 값을 랜덤하게 선택하게 되고, 이 값에 slot-time을 곱해서 백오프 시간을 결정한다. 앞서 언급한 것처럼, DCF는 non-QoS로서 모든 노드에 동등한 기회를 부여한다. 즉 모든 전송 스테이션은 최소CW값(CWmin) 15(혹은 31)에서 시작하여 최대 CW값(CWmax)값이 1023까지의 CW를 임의의 숫자를 선택한다. 각 무선스테이션은 최초 0에서 15(혹은 31) 사이의 임의의 숫자를 선택하여 백오프를 수행하게 된다. 만약 전송 후, 수신자로부터 ACK가 오지 않을 경우, 선택된 CW를 지수로 사용하여 그 범위의 한계가 2의 지수승으로 증가하고, 최대 범위인 1023까지 늘어난다. 다시 언급하지만 DCF는 위의 범위는 송신자에게 특별한 우선순위를 두지 않는다. 특정 스트림의 프레임 i 의 백오프를 계산하는 수식은 다음과 같다.

$$BO_k = \text{random}[0..CW_{\min AC(k)}] * \text{st}$$

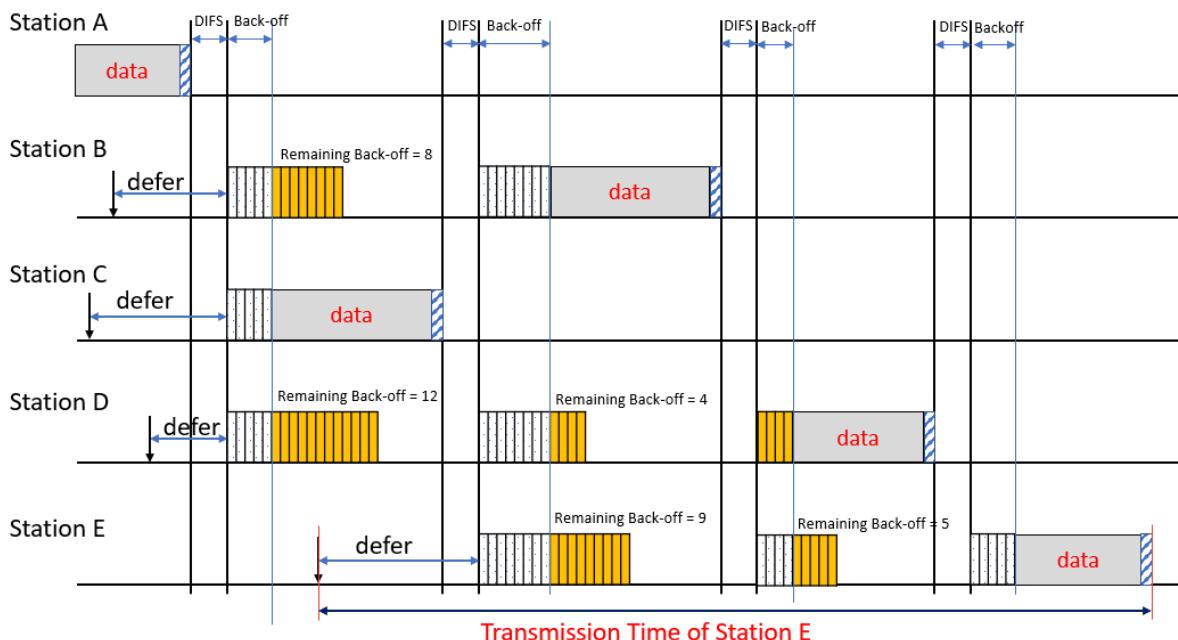


그림 3.3: 백오프 처리과정

그림 3.3은 DCF 방식을 따르는 다수의 스테이션이 어떻게 백오프를 수행하는지 보여준다. 다수의 스테이션이 DIFS 휴지기 이후, 동시에 백오프를 동시에 시작하고, 그 중 미리 계산된 백오프가 종료된 송신자가 채널을 점유한다. 이렇게 특정 스테이션이 전송을 시작하면 백오프를 진행하던 나머지 스테이션 역시 백오프를 중단한다. 이 때 백오프를 중지한 송신자는 남아있는 백오프 시간을 저장한다. 그리고 채널을 점유하여 데이터를 보내던 송신자가 데이터를 전송을 마치면 모든 송신을 원하는 스테이션, 즉 이전에 백오프를 중지하던 송신자와 새로운 송신자들이 모두 동일한 크기의 DIFS를 수행한다. DIFS 구간이 종료되면 이전에 백오프를 중지한 송신자는 나머지 백오프 시간을 진행하고, 새로운 송신자는 새로이 계산된 백오프 시간을 수행한다.

그림 3.4에서는 DCF이 제공하는 동등우선 순위 방식을 개선한 EDCA 방식을 보여준다. EDCA는 네트워크 부하와 사용자의 증가에 의해 발생하는 충돌을 감소시키기 위하여 패킷 전송을 완료하면 CW

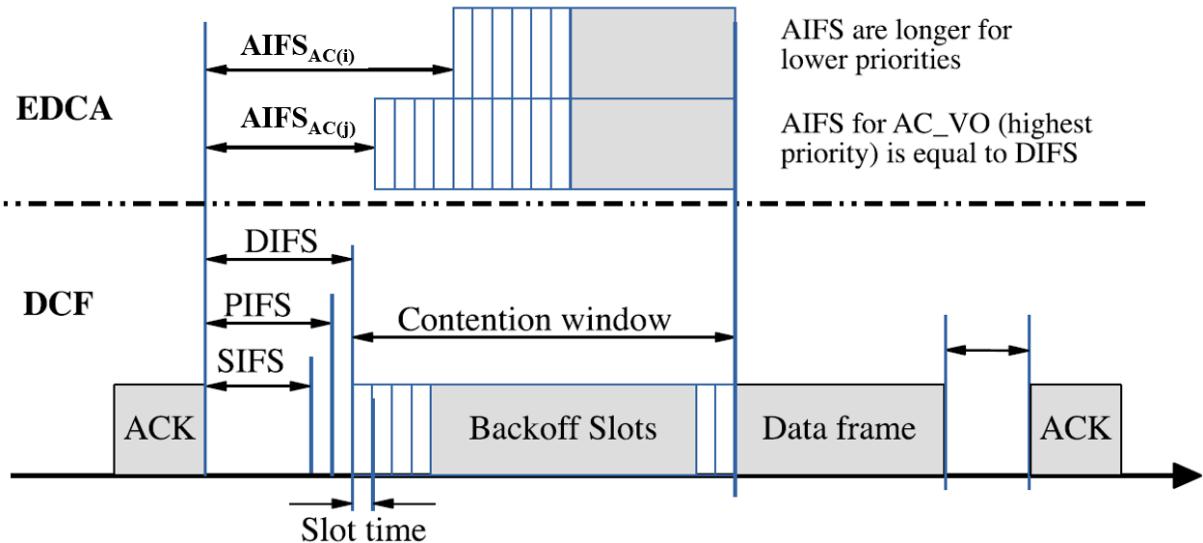


그림 3.4: DCF와 EDCA방식의 우선순위 부여 방식(tramarin2019real)

값을 초기화하는 것이 아니라 네트워크 사용률을 고려하여 천천히 값을 감소시키는 방안을 제시한다. QoS의 BSS(Basic Service Set) 내의 802.11g/a/n PHY를 위한 기본 EDCA CW 값은 표 3.1과 같다.

표 3.1: EDCA Contention Window values(802.11g/a/n PHY)

| | CWMin | CWMax |
|-------------------|-------|-------|
| Voice Queue | 3 | 7 |
| Voice Queue | 7 | 15 |
| Best Effort Queue | 15 | 1023 |
| Background Queue | 15 | 1023 |

또한 EDCA는 DCF의 모든 스테이션에 대한 동일한 크기의 DIFS 대신 스트림의 타입에 따라 차등을 두는 AIFS 방식을 사용한다. 즉 VO, VI, BE, BG의 큐에 우선순위를 부여하여, 각 큐에 다른 AIFS를 부여한다. 그에 더하여 다른 스테이션의 전송 시작으로 인해 중지된 백오프는 CW의 값을 새로 뽑는게 아니라 이전 CW값부터 재개하여 백오프를 수행한다.

그림 3.5는 802.11b PHY 기반한, IEEE 802.11e의 AIFS 값과 CW 값의 예를 보여준다. 이 그림에서 보여주는 것처럼 각 AC(Access Category, 접근부류)이 큐에 따른 AIFS가 다르기 때문에, 모든 카테고리의 CW값이 랜덤으로 선택된 결과가 모두 같은 경우라 할지라도, AIFS 값에 의해서 실제 백오프를 시작하는 시간이 다르기 때문에 VO나 VI 카테고리가 더 높은 우선순위를 갖는다. Voice나 Video도 CW값의 랜덤 선택 범위가 초기인 경우, 0~3과 0~7로 다르다. 이 때문에 VO가 VI가 확률적으로 더 낮은 값을 가질 수 있기 때문에 확률적으로 VO가 VI보다 더 짧은 백오프 시간을 가지게 된다. 그리고 확률적으로 VO가 더 짧은 AIFS + BO시간을 가지기 때문에, 높은 확률을 가지고 매체를 선점할 수 있다.

다음 절에서는 실시간 시스템 용 주기적인 스트림을 위해서, 최상 우선순위인 VO를 사용하여 프레임을 전송하고자 할 때, 다른 우선순위가 낮은 스테이션의 간섭으로 인해 어떠한 최악의 경우가 발생할

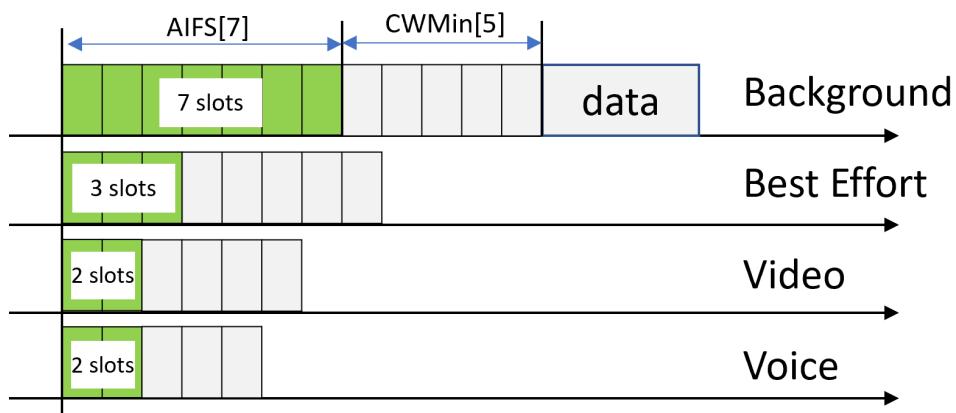


그림 3.5: 접근부류(Access category) 별 접근타이밍

수 있는지 분석한다. 아울러 최악의 경우를 없애고 매체 접유 대기 시간을 최소화하고 결정적이게 만들 수 있는 기법을 제안한다.

제 4 장

실시간 시스템을 위한 와이파이 QoS 한계 분석

이 연구개발의 목적은 IEEE 802.11 EDCA의 QoS의 최상위 우선순위 VO를 사용하여 스트림을 전송할 경우, 그 전송의 최악의 경우를 분석하고, 그 최악의 경우를 개선하는 방법을 제공하는 것이다. 이 장에서는 본 연구개발에서 제시하는 핵심 분석을 다룬다. 특히 실시간 시스템의 주기적 스트림 시간적 요구사항을 만족시키기 위해서, IEEE 802.11의 EDCA 내의 QoS 최상위 우선순위인 VO를 이용한 기술에 집중한다. 그럼 3.3에서 Station E에 프레임이 들어와서, 해당 프레임의 전송에 대한 ACK을 받을 끝날 때까지의 시간을 전송시간이라 할 때, 이 전송시간이 지연되는 최악의 경우를 분석하여 최악의 경우를 없애거나 전송시간을 줄이는 것이 목적이다. 이 장에서는 QoS의 4개의 서로 다른 접근부류(Access category, AC)를 사용하는 무선스테이션 관점에서 최악의 경우를 분석한다.

제 1 절 시스템 모델 및 문제정의

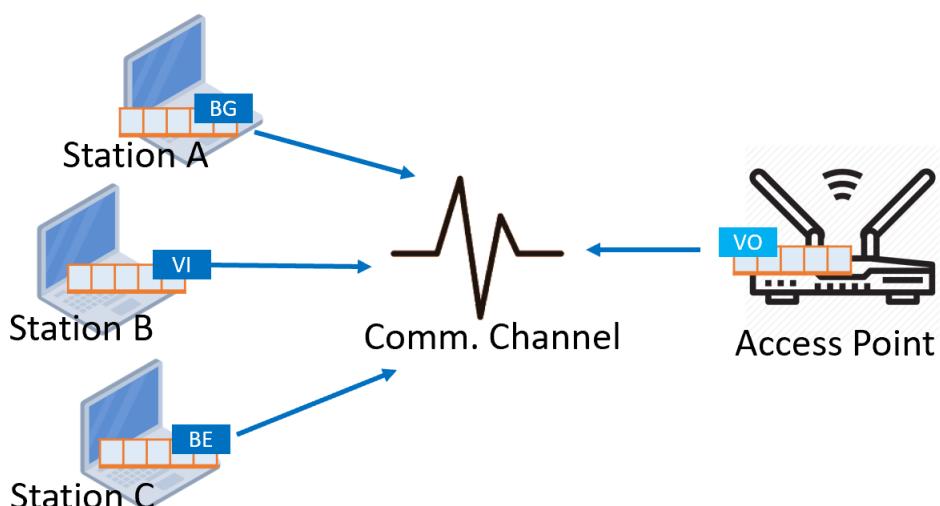


그림 4.1: 본 연구개발에서의 시스템 모델

표 4.1: 심볼 정의

| Symbol | Definition | Symbol | Definition |
|-----------------|------------------------|------------------------------|--|
| F | 모든 스트림 구분자의 집합 | $\text{AIFS}(n)$ | n 크기의 AIFS |
| s_k | 스트림 k | AIFS_{AC} | AC의 AIFS 크기 |
| p_k | 스트림 k 의 주기 | $\text{AIFS}_{\text{AC}(k)}$ | k 스트림의 AC AIFS 크기 |
| e_k | 스트림 k 의 매체 전송시간 | $\text{AIFS}[n]$ | n 번째 AIFS 슬롯 |
| $f_{k,j}$ | k 스트림의 인스턴스, 프레임 j | $\text{CW}(n)$ | n 크기의 CW |
| BO_k | 프레임 k 의 백오프 시간 | $\text{CW}[n]$ | n 번째 CW 슬롯 |
| Θ_k | 스트림 k 의 최대 스위칭 지연시간 | CW_j | 프레임 j 의 CW 크기 |
| Π_k | 스트림 k 의 최대 채널대기 지연시간 | τ | $\text{trt}(\text{SIFS} + \text{ACK})$ |
| \mathcal{R}_k | 스트림 k 의 전송 큐 지연시간 | $\text{trt}(x)$ | 프레임 x 의 매체 전송시간 |
| WCTT_k | 스트림 k 의 최악의 경우 전송시간 | κ | 와이파이 최대 크기 프레임(2358)의 전송시간 |
| st | Slot-time | $\text{AC}(k)$ | 스트림 k 의 접근부류(access category) |

본 연구개발에서는 그림 4.1과 같은 시스템 모델을 사용한다. 이 후 이 연구개발에서 사용하는 모든 심볼은 표 4.1에 정의되어 있다.

시스템 모델이 사용하는 가정은 다음과 같다. AP가 전송할 스트림을 전송 큐에 할당한다. 그리고 실시간 시스템의 주기적 스트림이 최상의 우선순위 VO를 사용하고, 나머지 스트림들이 낮은 VI, BE, 혹은 BG 우선순위를 사용한다. 각 무선스테이션은 하나의 우선순위와 단일 큐를 사용하여 같은 접근부류(AC)의 스트림만을 처리한다고 가정한다. 이 시스템 모델에서의 핵심은 각 스테이션과 AP가 데이터 전송을 공유자원인 통신 채널을 점유하는 하는 것이다. 통신 채널의 특징 상, 현재 통신 채널이 점유되고 있는지 알 수 있지만, 스테이션은 일단 전송을 시작하면 전송 성공 여부를 송신한 데이터에 대한 ACK 통해서만 알 수 있다. 그리고 만약 실패했다면 재전송을 수행한다. 그리고 프레임에 대한 전송이 성공하면 전송된 데이터를 전송 큐에서 삭제한다.

실시간 시스템의 데이터 스트림은 다음과 같이 정의된다. 우선 F 는 전송 스트림의 구분자의 집합이다. 즉 k_{max} 개의 스트림의 정의한다. F 는 다음과 같이 정의된다.

$$F = \{k \in \mathbb{N} \mid k \leq k_{max}\}.$$

스트림 s_k 은 다음과 같이 정의된다.

$$s_k = (p_k, e_k, d_k)$$

p_k 는 전송 주기이며, e_k 전송시간을 가리킨다. 그리고 d_k 는 전송마감시간(데드라인)을 가리킨다. 스트림의 인스턴스 프레임은 $f_{k,j}$ 로 정의되는데, j 는 스트림 k 의 j 번째 인스턴스를 가리킨다. 시스템 설명의 편의를 위해서 몇 가지 정의를 다음과 같이 제공한다.

이 연구개발에서 AP의 전송 시의 **최악의 경우 전송시간(Worst-Case Transmission Time, WCTT)**은 프레임이 AP에 들어 올 때부터, AP가 해당 프레임을 위한 ACK을 수신하기까지의 시간 중 최대 길이로 정의한다. 이 연구개발에서 수신자는 항상 ACK를 보낼 수 있다고 가정한다. 따라서

특정 스트림의 최악의 경우 전송시간은 더 이상 재전송이 필요없는 더 이상 송신대기열에 머물 필요가 없는 시점을 가리킨다.

주어진 스트림 k 에 대하여 $WCTT_k$ 는 최대 스위칭 지연시간 Θ_k 과 최대 채널 대기 지연시간(Π_k), 그리고 데이터 전송 시간 e_k 과 종료시간 τ 으로 구성된다.

$$WCTT_k = \Theta_k + \Pi_k + e_k + \tau \quad (4.1)$$

Θ_k 는 다음과 같이 구성된다.

$$\Theta_k = \delta + \max(\kappa + \tau, \mathcal{R}_k) \quad (4.2)$$

위 수식에서 δ 는 프레임이 스테이션이나 AP에 들어가서 처리되는 **최대 스위칭 지연시간 (Maximum switching delay)**이다. 그에 더하여 Θ_k 값에 $\kappa + \tau$ 와 \mathcal{R}_i 값 중 큰 값이 더하여진다. $\kappa + \tau$ 는 현재 채널을 점유하여 전송되는 프레임으로 인한 최악의 지연을 가르키는데, 주어진 프레임이 들어가는 큐에 선행하는 프레임이 없을 때 즉시 주어진 프레임이 큐를 점유할 수 있지만, 채널이 데이터 전송 중이라면 현재 전송이 종료될 때까지 기다려야 한다. 즉 최악의 경우 $\kappa + \tau$ 를 기다려야만 채널 점유를 시도할 수 있다. 만약 큐에 선행하는 프레임이 있다면 그 프레임이 모두 전송 된 후, 주어진 프레임이 큐의 다음 전송 대상이 되는 시간까지 채널 점유 시도가 지연되는데, 그 지연을 **최대 큐 점유시간 \mathcal{R}_k** 로 정의한다. 따라서 최대의 Θ_i 값이 같은 위의 식 4.2처럼 정의된다. \mathcal{R}_k 에 대해서는 다음 절에서 자세히 다룬다.

매채 점유를 위한 최대 채널 대기 지연시간 Π_k 는 전송 큐에 맨 앞에서 있어도 채널을 선점할 수 있기 때문에 생기는 지연 중 최대 대기를 가리킨다. 이는 다음과 같이 정의될 수 있다.

$$\Pi_k = (\text{AIFS}_{\text{AC}(k)} \cdot \text{st}) \cdot \mathcal{P} + (\Gamma + \tau) \cdot \mathcal{P} + (\text{CW}(n) \cdot \text{st}) \quad (4.3)$$

식 4.3에서 \mathcal{P} 는 다른 스테이션에 의해 채널 점유하여 백오프를 수행/반복하는 횟수이다. 스트림 i 와 j 에 대해서 다음의 조건을 만족하는 경우 스트림 j 이 채널을 선점하며, 스트림 j 이 채널을 선점할 때, 스트림 i 는 백오프를 중지해야 한다.

$$\text{AIFS}_{\text{AC}(i)} + \text{CW}_i > \text{AIFS}_{\text{AC}(j)} + \text{CW}_j \quad (4.4)$$

다음 절에서는 \mathcal{R}_k 와 \mathcal{P} 와 관련된 최악의 경우를 살펴보고 5장에서 이 연구개발에서 제안하는 솔루션을 살펴본다.

제 2 절 최악의 경우 분석

IEEE 802.11e EDCA의 QoS를 사용하는 와이파이 스테이션에서 최대 전송시간은 식 (4.1)에서 볼 수 있는 것처럼 $WCTT_k$ 는 최대 스위칭 지연시간과 최대 채널 대기 지연시간이 좌우한다. 그리고 최대

스위칭 지연시간에서는 \mathcal{R}_k 이 변수이며, 최대 채널 대기 지연시간에서는 \mathcal{P} 가 변수이다. 이 절에서는 이들과 관련된 최악의 경우를 분석한다.

2.0.1 최대 큐 점유시간

주어진 스트림 k 에 대하여, 동일한 큐를 사용하는 다른 프레임에 의한 지연 \mathcal{R}_k 는 전송 큐를 공유하는 서로 다른 스트림이 매 주기 p_j 마다 최악의 전송 시간 WCTT _{j} 의 시간을 소모한다고 가정한다. 전송 큐가 선입선출(First-In, First-Out) 방식을 사용한다면 최대 선점 시간 ψ 는 다음과 같다.

$$\mathcal{R}_k = \sum_{j \in ALL(k)} \left\lceil \frac{\mathcal{R}_k}{p_j} \right\rceil \cdot WCTT_j \quad (4.5)$$

$$\mathcal{R}_k^n = \sum_{j \in ALL(k)} \left\lceil \frac{\mathcal{R}_k^{n-1}}{p_j} \right\rceil \cdot WCTT_j \quad (4.6)$$

수식 4.5에서 $ALL(i)$ 은 스트림 i 와 동일한 큐를 사용하는 i 를 제외한 모든 스트림을 가리킨다. 수식 4.5은 \mathcal{R}_k 을 순환적인(recursive)한 방식으로 정의하고 있다; \mathcal{R}_k 시간 동안, 동일한 대기열을 사용하는 모든 프레임의 생성 횟수와 각각의 전송시간의 곱, 즉 다른 프레임들의 계산 대상 프레임 보다 항상 앞서는 최악의 경우($\sum_{j \in ALL(k)} \left\lceil \frac{\mathcal{R}_k}{p_j} \right\rceil \cdot WCTT_j$)의 횟수에 각각의 전송 시간의 곱하고 대상 프레임의 전송시간을 더하여 계산한다. 수식 4.5은 순환적으로 정의되어 \mathcal{R}_k 팀이 양쪽에서 있어서 그 계산이 수월하지 않다. 수식 4.6은 수식 4.5에서 순환이 되는 \mathcal{R}_k 팀을 없앤 구조를 가지고 있다.

다음 절에서는 최대 채널 대기 지연시간에서는 \mathcal{P} 와 관련있는 최악의 시나리오들을 분석한다.

2.1 채널 대기 지연 최악의 경우

VO우선순위를 사용하는 스테이션이 다른 우선순위를 사용하는 스테이션들이 동시에 매체를 접근할 때, VO우선순위를 사용하는 스테이션이 점유하지 못하는 경우는 다음과 같다. 즉 VO 우선순위를 사용하는 스트림을 i 라 하고 다른 낮은 우선순위를 사용한 스트림이 j 라 할 때 식 (4.4)를 만족하는 경우는 다음과 같다.

Case 1: ($AIFS_{AC(i)} = AIFS_{AC(j)}$) \wedge ($CW_i > CW_j$)

Case 2: ($AIFS_{AC(i)} < AIFS_{AC(j)}$) \wedge ($CW_i > CW_j$)

Case 1은 VO우선순위를 가진 스트림과 VI우선순위를 가진 스트림이 경쟁하는 경우 발생할 수 있다. Case 2는 VO우선순위를 가진 스트림과 VE우선순위를 가진 스트림이 경쟁하는 경우 발생할 수 있다.

각 경우에 대해 어떠한 최악의 경우가 발생하는지 설명한다. 이 연구개발에서 최악의 경우를 고려하면서 공통적으로 갖는 가정은 다음과 같다.

- AP는 최상위 우선순위 큐인 VO 큐를 사용한다. 그리고 전송을 위한 CW로 CW(3)을 뽑는다.
- Station C는 BE 큐 혹은 VI 큐를 사용한다. 그리고 프레임 전송을 위해서 매번 CW(3)을 뽑는다.
- AIFS의 1 st의 길이와 백오프의 1 st의 길이는 같다고 가정한다.

- 모든 데이터 전송은 와이파이의 최대 크기의 데이터 프레임을 전송한다고 가정한다.

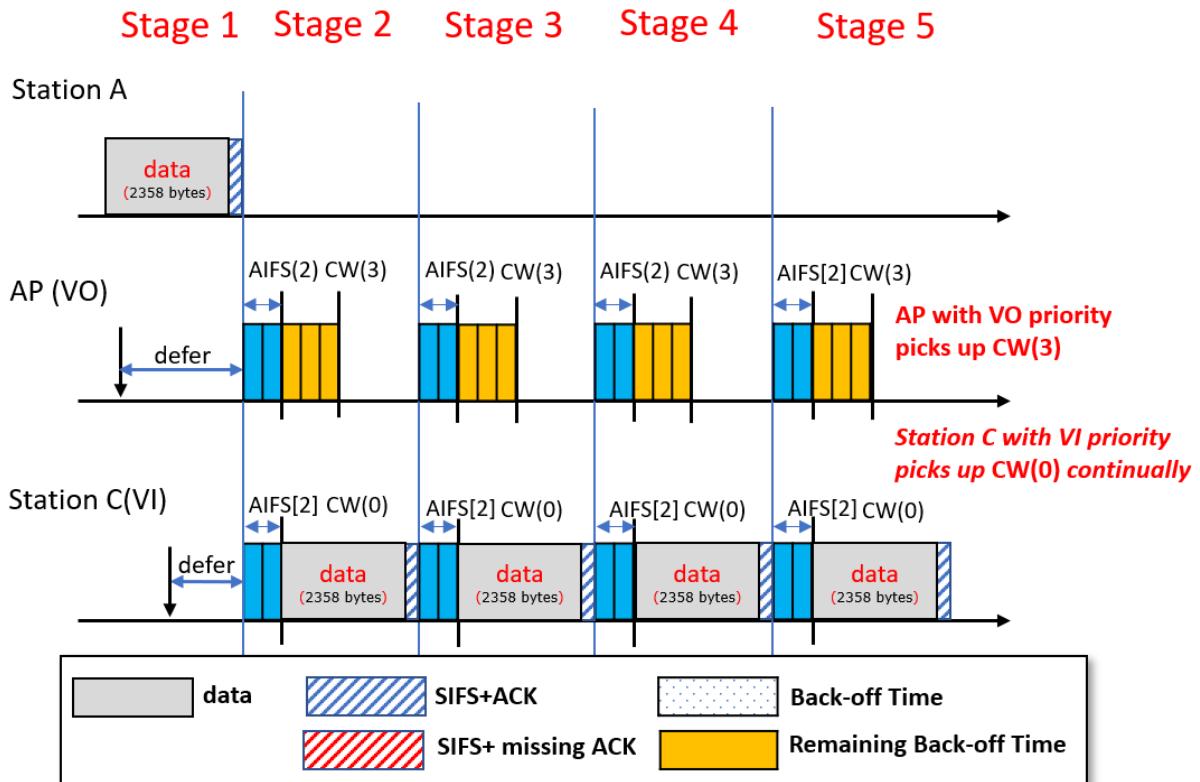


그림 4.2: AP가 CW(3)을 선택하고, Station C가 CW(0)을 연속적으로 선택하여, VO 프레임이 지연되는 경우, AP의 백오프시간이 줄지 않음

Case 1: VO-VI 그림 4.2에서 Case 1의 경우 발생할 수 있는 최악의 경우를 보여준다. AIFS[n]는 n 만큼의 AIFS의 st를 수행함을 의미한다. CW(n)는 스테이션이 경쟁윈도우를 위해 n값을 선택했음을 의미한다.

VO-VI 최악의 시나리오는 다음과 같다. 이 시나리오에서 Station C가 VI 우선순위를 사용한다.

Stage 1: Station A는 ACK을 받은 후 데이터 전송을 종결한다.

Stage 2: AP와 Station C는 동시에 각자의 정해진 각각의 AIFS(2) 씩 수행한다. AP는 백오프 시작하고, 동시에 Station C CW(0)을 뽑았기 때문에 데이터 전송을 시작한다. 따라서 AP는 백오프를 시작하지 못하고 지연하게 된다. Station C가 데이터 전송을 수행하고, 수신자에게 ACK을 받고 종료된다.

Stage 3: Stage 3부터는 Stage 2를 반복한다.

VO-VI 최악의 시나리오 경우 VI가 CW(0)을 뽑는다면 VO의 백오프시간은 줄지 않고, Stage 2를 무한 반복하게 된다. 이렇게 백오프가 줄지 않은 현상을 “No-Backoff-Decrement(NBD)” 현상라하자.

이제 Case 2에서 발생하는 최악의 경우를 살펴본다. 그림 4.3는 우선순위가 높은 VO 사용하는 AP가 세번째 우선순위인 BE 사용하는 Station C에게 지속적으로 우선순위가 무시당하는 경우를

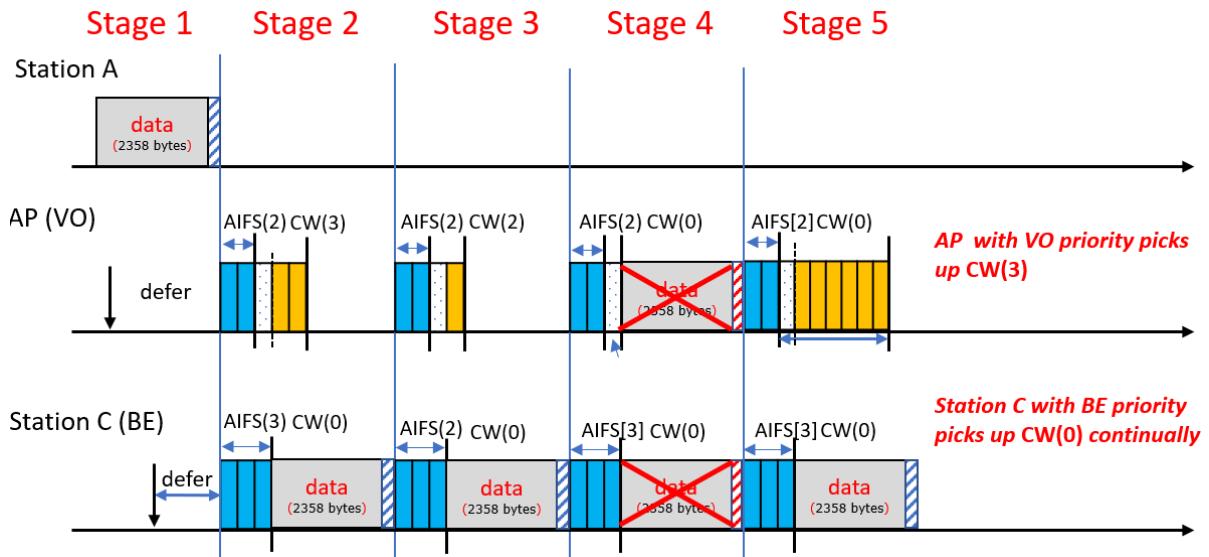


그림 4.3: AP가 최대 CW(3) 선택하고, BE 프레임이 최소 CW(0)을 연속적으로 선택하여, VO 프레임이 지연되는 경우

보여준다. 이 경우의 최악의 시나리오는 다음과 같다.

Stage 1: Station A는 ACK을 받은 후 데이터 전송을 종결한다.

Stage 2: AP와 Station C는 각자의 정해진 각각의 AIFS(2) 와 AIFS(3)을 수행한다. AP가 먼저 AFIS 수행을 마치고 백오프시간 중 첫번째 슬롯 CW[1]을 수행하고, 동시에 Station C는 AIFS[3] 의 나머지 슬롯을 수행한다. Station C는 CW[0]을 선택했기 때문에 AP가 백오프 중 두 번째 슬롯 CW[2]를 시작할 때, Station C가 전송을 시작하고, 이 전송은 수신자에게 ACK을 받고 종료된다.

Stage 3: Stage 2를 반복한다.

Stage 4: AP와 Station C는 Stage 2와 유사하게 각자의 정해진 각각의 AIFS 수행한다. AP가 먼저 AIFS(2) 수행을 마치고 백오프시간 중 마지막 한 슬롯을 수행한다. 동시에 Station C는 마지막 AIFS인 AIFS[3]를 수행한다. Station C는 CW(0)을 선택했고 AP가 백오프시간을 모두 종료했기 때문에 두 스테이션 모두 전송을 시작한다. 하지만 두 스테이션이 동시에 전송을 수행하기 때문에 전송은 실패하게 되고, 두 스테이션 모두 ACK을 받지 못한다.

Stage 5: AP와 Station C는 각자의 정해진 각각의 AIFS[2] 와 AIFS[3]을 수행한다. AP는 이전 전송을 실패했기 때문에 0부터 CW(7) 중 CW(7)을 뽑고, Station C는 0부터 CW(1023)중 0을 뽑는다. 그리고 Stage 2를 반복한다.

불행하게도 그림 4.3의 최악의 시나리오에서 Station C가 CW[0]을 반복해서 선택할 경우, AP는 3 번의 채널 점유 지연 후 전송을 시작하지만, 전송 충돌로 인해 전송이 결구 전송이 실패하고, CW의 값은 이전 CW 값을 지수로 사용한 2의 지수로 증가하여 지속적으로 전송에 실패할 수 있다.

살펴본 것처럼 전송을 위해 채널에 접근하는 우선순위가 다른 스테이션간에는 AIFS가 차이가 있어도, VO우선순위 스테이션보다 다른 스테이션이 더 낮은 CW값을 뽑으면서 채널 충돌이 발생할 수 있다.

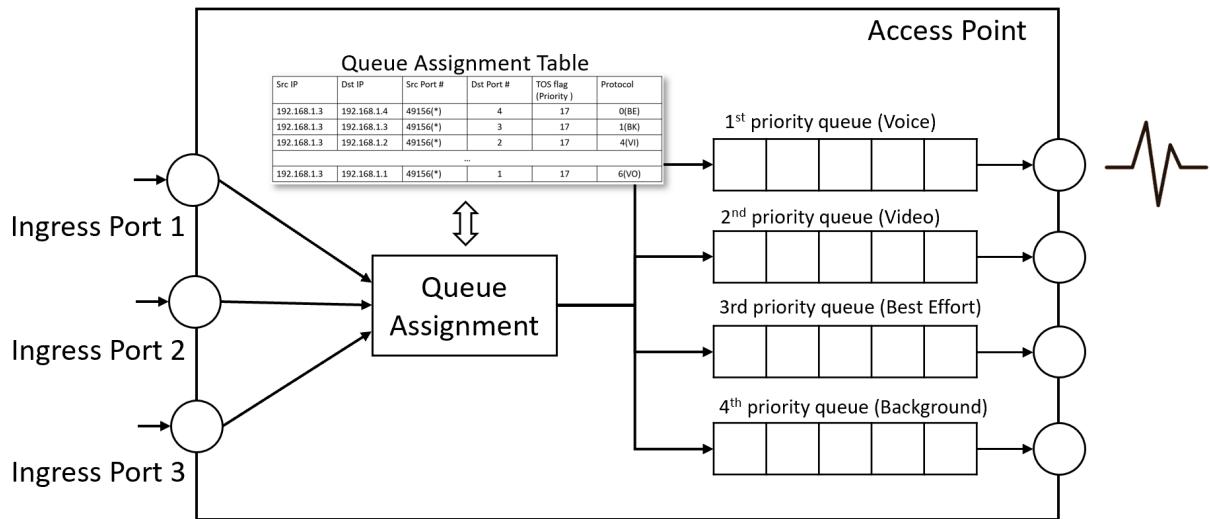


그림 4.4: 제안된 기법의 AP 적용

또한 위의 최악의 시나리오는 VI나 BE의 출현에 따라 더 복잡한 경우로 발생할 수 있다. 결론적으로 우리가 계산하고 하는 식 (4.3)에서 \mathcal{P} 가 결정적으로 전체 식을 비결정적으로 만든다. 5장에서는 \mathcal{P} 가 결정적으로 만드는 알고리즘을 제공한다.

제 3 절 AP 기반의 실시간 시스템 용 와이파이 슬라이싱

본 절에서는 이 연구개발에서 제안한 VO우선순위를 활용한 실시간 용 와이파이 스트림을 AP에 적용 시켜, AP의 다운로드 스트림이 다른 노드들에 비해서 우선순위를 가지고 서비스를 제공하는 플랫폼 구현을 제안한다. 이 연구개발에서는 앞서 고려한 최악의 경우가 가능한 시나리오를 포함하여 실험 한다.

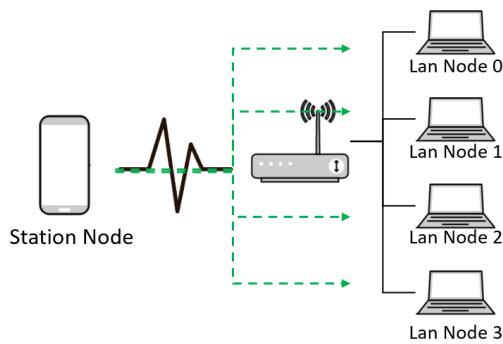
그림 4.4은 실시간 프레임 전송을 위한 AP의 구조를 보여준다. AP는 중앙제어(centralized control)를 통해서 각 스트림에 대한 큐 할당을 정보를 얻어 QAT(Queue Assignment Table)에 저장된다. QAT 테이블은 다음과 같은 정보를 가지고 있다.

표 4.2: Queue Assignment Table Fields

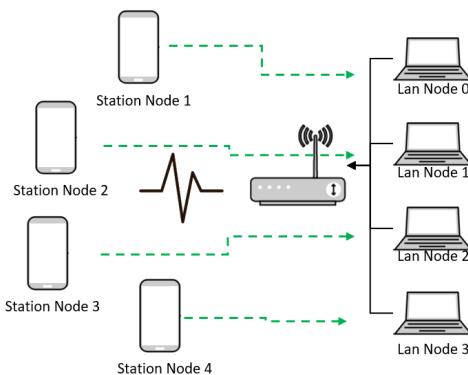
| Src IP | Dst IP | Src Port | Dst Port | ToS Flag (Priority) | Protocol |
|--------|--------|----------|----------|---------------------|----------|
| 출발지 IP | 도착지 IP | 출발지 포트 | 도착지 포트 | 토스 플래그(우선순위) | 프로토콜 |

AP를 통해서 들어온 스트림은 접근부류(Access Category)에 따라서 큐에 할당된다. VO 큐는 최고 우선순위를 갖는 큐로서 실시간 시스템의 주기적 데이터 전송한다고 가정한다. 기본적으로 각 전송 대기열, 즉 전송큐는 FIFO 큐이다. 즉 먼저 들어온 프레임이 반드시 먼저 나가야 한다.

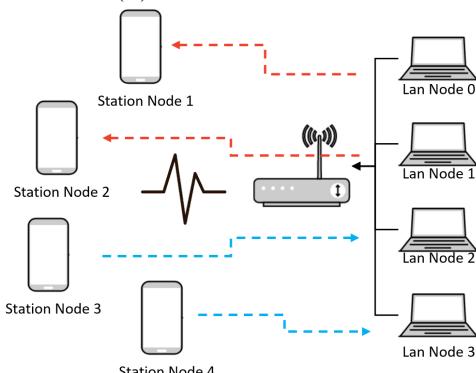
본 연구개발은 AP 관점에서 다운링크 프레임에 집중한다. 따라서 수신자는 AP가 전송한 프레임을 문제없이 받을 수 있다고 가정한다. 즉 수신자는 수신된 프레임에 대해서 ACK를 반드시 보낸다. 앞서 언급한 것처럼 최악의 경우 전송 시간은 프레임이 AP에 들어오면서부터 시작한다. 전송 데이터 종료는 AP의 관점에서 전송 완료 시점을 즉 해당 ACK을 받았을 때의 시간으로 가정한다.



(a) 첫번째 실험의 구성



(b) 두번째 실험의 구성



(c) 세번째 실험의 구성

제 4 절 실험 및 결과

NS3(3.31버전)로 실험하였다. 실험환경은 다음과 같다.

- Ubuntu 18.04
- CPU Amd-Ryzen 5-3500u
- Memory 6GB

이 실험에서는 중앙 제어소에서 AP내의 QAT를 미리 설정했다고 가정하고 실험을 진행하였다. 또, 모든 결과는 프레임 전송 시퀀스와 해당 프레임 수신 시퀀스가 같은 경우만을 고려하였다. 즉 수신자가 수신한 결과가 있는 전송 스트림에 대해서만 결과를 도출하였다.

4.1 실험셋팅 1

실험을 위한 첫번째 셋팅은 그림 4.5a 같은 토플로지 상에서 진행되었다. 이 셋팅에서는 모두 5가지의 실험이 아래와 같이 진행되었다. 각 실험은 시뮬레이션 시간으로 3,650sec 동안 실행하였다.

실험 1-1 (모든 스트림이 동일한 데이터 크기(100bytes)와 동일한 주기(100ms))를 가질 때)
 표 4.3은 실험 1-1부터 실험 1-4번째까지 사용된 QAT이다. 이 QAT에서 출발지 포트의 경우, 와일드 카드(*)를 사용하여 임의로 지정하여도 정상적으로 작동할 수 있도록 큐 할당이 이루어지도록 구현하였다.

표 4.3: 실험셋팅 1의 QTA 1

| Src IP | Dst IP | Src Port | Dst Port | ToS Flag (Priority) | Protocol |
|-------------|-------------|----------|----------|---------------------|----------|
| 192.168.1.3 | 192.168.2.4 | 49156(*) | 4 | 0(BK) | 17 |
| 192.168.1.3 | 192.168.2.3 | 49156(*) | 3 | 1(BE) | 17 |
| 192.168.1.3 | 192.168.2.2 | 49156(*) | 2 | 4(VI) | 17 |
| 192.168.1.3 | 192.168.2.1 | 49156(*) | 1 | 6(VO) | 17 |

모든 스트림의 주기는 100 ms로 발생하고 프레임의 데이터 크기는 100 bytes로 고정하였다. 그림 4.6a는 전송지연시간에 대한 상자그래프 보여준다. 상자의 가운데 빨간색 가로선이 전체의 평균을 가리킨다. 상사의 윗쪽 가로선은 전체의 75% 그리고 상사의 아래 가로선은 전체의 25%를 가리킨다. 박스 바깥쪽의 가로선과 세로선은 통계상 의미있는 최고값과 최저값을 가리킨다. 그 외의 가로선 바깥의 점들은 아웃라이어(Outlier)라고 불린 이상치 값이다. 그림 4.6a에서 가장 높은 VO 스트림의 박스와 평균지연시간이 다른 우선순위의 스트림보다 상대적으로 극히 작은 것을 확인할 수 있다. 또한 VO스트림의 박스가 거의 선에 가까운 것으로 볼 때, 전송시간이 매우 규칙인 것을 알 수 있다.

실험 1-2 (모든 스트림이 동일한 데이터 크기(100 bytes)와 다른 주기를 가질 때) 이 실험 세팅에서는 스트림의 주기를 다르게 부여하여, VO 스트림은 100 ms, VI 스트림은 200 ms, BE 스트림은 300 ms, 그리고 BK 스트림은 400 ms로 셋팅하였다. 여기에서는 실험 1-1에서 보여주었던 양상과 다소 유사하다. 특히 VO스트림이 이상치 없이 전송시간의 매우 결정적인 것을 알 수 있다. 하지만 VO 스트림 외의 다른 스트림들의 이상치들이 많아 진 점으로 볼 때, 다른 것의 전송시간의 비결정성이 더욱 증가한 것으로 볼 수 있다.

실험 1-3 (모든 프레임이 다른 데이터 크기와 동일한 주기를 가질 때) 이 실험에서는 각 프레임의 데이터 크기에 변화를 주어 VO프레임 100 bytes, VI프레임 200 bytes, BE프레임 300 bytes, 그리고 BK프레임 400bytes 으로 설정하였다.

그림 4.6c에서 볼 수 있는 것처럼, 실험 1-1에서 보여준 양상과 거의 같은 결과를 보준다. 다만 상사의 크기가 모드 작아진 것을 볼 때, 전체 프레임들의 시간적 결정성이 높아진 것을 볼 수 있다. 이 경우에도 VO프레임이 다른 우선순위 프레임 보다 전송이 항상 앞섰으며, 우선순위가 전도되는 경우도 발생하지 않았다.

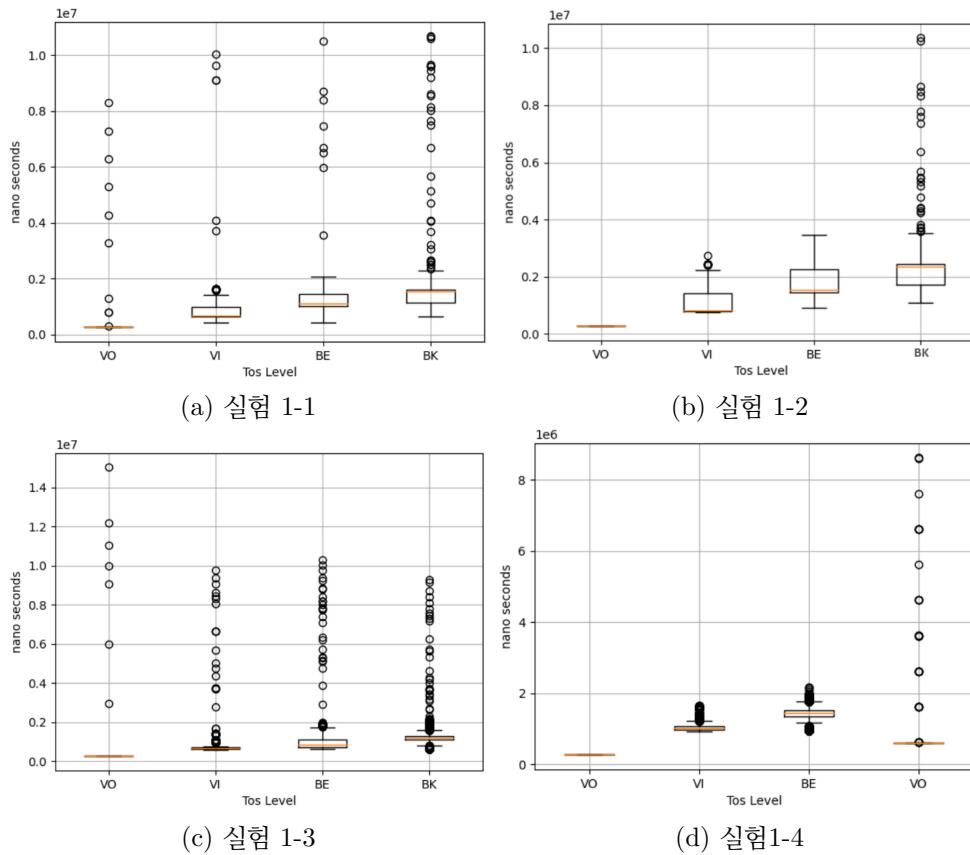


그림 4.6: 실험셋팅 1의 실험 결과

실험 1-4 (모든 프레임이 동일한 데이터 크기와 동일한 주기를 가지면서, 4개의 스트림 중 두 개의 스트림에 동일한 VO우선순위를 부여할 때) 그 다음으로 동일한 세팅하에서 두 개의 프레임이 VO로 우선순위가 부여되었을 때를 실험하여 보았다. 새로운 QAT는 표 4.4 같다.

표 4.4: 실험셋팅 1의 QTA 2

| Src IP | Dst IP | Src Port | Dst Port | ToS Flag (Priority) | Protocol |
|-------------|-------------|----------|----------|---------------------|----------|
| 192.168.1.3 | 192.168.2.4 | 49156(*) | 4 | 6(VO) | 17 |
| 192.168.1.3 | 192.168.2.3 | 49156(*) | 3 | 1(BE) | 17 |
| 192.168.1.3 | 192.168.2.2 | 49156(*) | 2 | 4(VI) | 17 |
| 192.168.1.3 | 192.168.2.1 | 49156(*) | 1 | 6(VO) | 17 |

그림 4.6d에서 앞선 결과와 동일하게 VO스트림들이 가장 낮은 평균전송지연을 보여주고 있으며, 어떠한 경우도 우선순위가 전도되지 않는 경우를 보여준다.

실험 1-5 이 실험에서는 VO 스트림과 다른 우선순위를 가진 스트림 간의 우선순위가 전도되는 경우, 즉 우선순위가 낮은 스트림이 먼저 나가는 경우를 실험하였다. 앞선 실험과 동일한 실험 세팅에서, Station Node에 대해서 모빌리티 옵션(NS3 옵션) 설정하고, 해당 토플로지에 대해서 시뮬레이션 시간으로 1시간 씩 10회 실행하였다. 그 결과, VO우선순위 스트림이 다른 우선순위의 스트림에 의해 우선순위가 전도되는 경우는 0.02%를 넘지 않았으며, 동일한 세팅 하에 시뮬레이션 시간으로 10시간 실행하였어도 전도되는 비율은 단지 0.07%를 넘지 않았다.

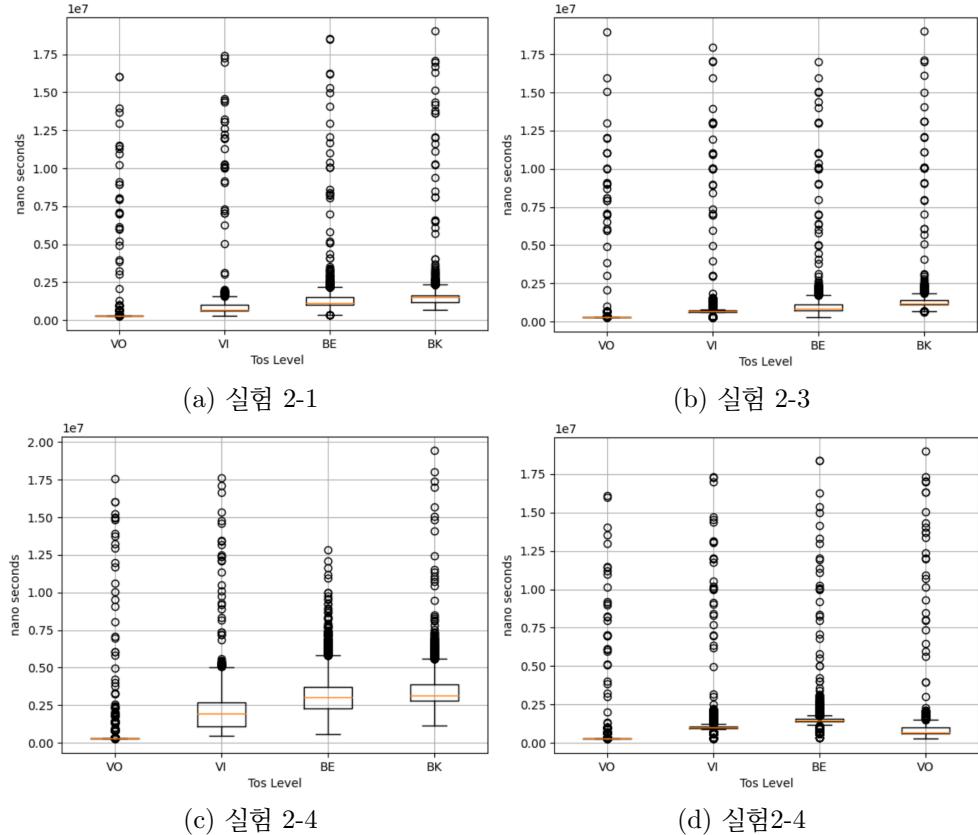


그림 4.7: 실험셋팅 2의 실험 결과

4.2 실험셋팅 2

두번째 실험 셋팅 그림 4.5b는 각각의 무선노드가 서로 다른 우선순위를 가진 패킷을 다른 AP 노드로 전송하는 것이다. 스트리밍 데이터 크기와 시뮬레이션 시간, 시간 간격은 위의 실험셋팅 1과 와 동일하게 설정한 후 실험을 진행하였다. 기본적인 실험의 가정은 실험셋팅 1과 같으며, 실험 순서도 위와 동일하게 진행하였다.

이 셋팅에서 실험 2-1부터 실험 2-4까지 모두 실험셋팅 1과 동일한 조건 하에서 실험을 진행하였다.

다음 표 4.5는 실험2-1부터 2-3까지 사용한 QAT이다. 그리고 표 4.6는 실험2-4가 사용한 QAT이다

표 4.5: 실험셋팅 2의 QTA 1

| Src IP | Dst IP | Src Port | Dst Port | ToS Flag (Priority) | Protocol |
|-------------|-------------|----------|----------|---------------------|----------|
| 192.168.1.4 | 192.168.2.4 | 49156(*) | 4 | 0(BK) | 17 |
| 192.168.1.3 | 192.168.2.3 | 49156(*) | 3 | 1(BE) | 17 |
| 192.168.1.2 | 192.168.2.2 | 49156(*) | 2 | 4(VI) | 17 |
| 192.168.1.1 | 192.168.2.1 | 49156(*) | 1 | 6(VO) | 17 |

그림 4.7a~4.7d의 결과를 보면 실험셋팅 1과 양상이 같으며, 어느 경우도 VO 우선순위의 프레임의 상자가 다른 우선순위의 프레임의 상자와 뒤바뀌지 않음을 볼 수 있다. 다른 점은 이상치들이 많이 늘어난 것을 관찰 할 수 있다. 또한 이상치들로 분포가 상당히 넓게 늘어난 것을 확인할 수 있다. 이러한 이상치들 늘어난 이유에 대해서는 보다 심층적인 분석이 필요하다.

표 4.6: 실험셋팅 2의 QTA 2

| Src IP | Dst IP | Src Port | Dst Port | ToS Flag (Priority) | Protocol |
|-------------|-------------|----------|----------|---------------------|----------|
| 192.168.1.4 | 192.168.2.4 | 49156(*) | 4 | 0(VO) | 17 |
| 192.168.1.3 | 192.168.2.3 | 49156(*) | 3 | 1(BE) | 17 |
| 192.168.1.2 | 192.168.2.2 | 49156(*) | 2 | 4(VI) | 17 |
| 192.168.1.1 | 192.168.2.1 | 49156(*) | 1 | 6(VO) | 17 |

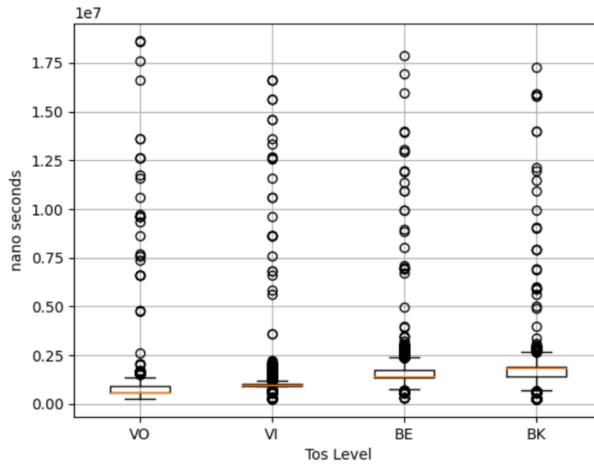


그림 4.8: 실험3-1

4.3 실험셋팅 3

세번째 실험 구성은 AP의 다운로드 스트림에 VO와 VI의 높은 우선순위를 갖는 스트림들로 구성하고, 무선 스테이션은 업로드 스트림으로 구성하였다. 이를 반영한 QAT이 표 4.6이다. 스트림들은 모두 같은 크기의 데이터와 같은 주기를 가지고 데이터를 전송한다고 가정한다.

표 4.7: 실험셋팅 2의 QTA 3

| Src IP | Dst IP | Src Port | Dst Port | ToS Flag (Priority) | Protocol |
|-------------|-------------|----------|----------|---------------------|----------|
| 192.168.2.4 | 192.168.1.4 | 49156(*) | 4 | 0(VO) | 17 |
| 192.168.2.3 | 192.168.1.3 | 49156(*) | 3 | 1(VI) | 17 |
| 192.168.1.2 | 192.168.2.2 | 49156(*) | 2 | 4(BE) | 17 |
| 192.168.1.1 | 192.168.2.1 | 49156(*) | 1 | 6(BK) | 17 |

그림 4.8에서 볼 수 있는 것처럼, VO 스트림의 군집이 다른 우선순위를 가진 스트림의 군집보다 앞서 있음을 볼 수 있다. 다만 이상치들의 분포가 상당히 넓게 퍼져있는 부분은 심층적인 분석이 필요하다.

마지막으로 그림 4.9는 실험세팅 3 하에서, 각 스테이션과 AP에서 같은 주기에 동일한 크기의 프레임을 전송한다고 가정했을 때, 우선순위가 따라 가장 먼저 전송이 완료되는 비율을 보여준다. 그림 4.9에서는 VO 프레임이 단지 65.74%만 가장 먼저 도착했음을 볼 수 있다.

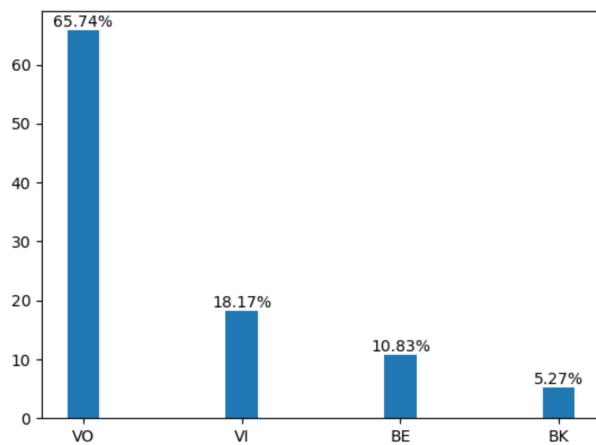


그림 4.9: 실험3-2: 우선순위에 따른 최우선 전송 비율

제 5 장

새로운 접근방법

본 장에서는 VO우선순위를 가진 스트림의 시간적 결정성을 보증하는 기법을 제공한다.

앞선 절에서 살펴 본 것처럼 우선순위가 높은 스테이션이 우선순위가 낮은 스테이션에 선점되어 지속적으로 채널 사용이 지연되는 경우는 다음 두 경우이다.

- Case 1: 두 개 이상의 스테이션이 동시에 백오프를 종료하여 채널 점유 충돌이 발생할 경우 (백오프 종류 충돌)
- Case 2: 우선순위가 낮은 스테이션이 지속적으로 높은 우선순위의 CW값보다 작은 CW값을 뽑을 경우 (NBD현상)

Case 2는 확률적으로 매우 낮다고 할 수 있다. Case 1는 Case 2에 비해 상대적으로 발생 빈도가 높다. 왜냐하면 VO우선순위와 VI우선순위는 동일한 AIFS값을 사용하기 때문에, 만약 그림 3.3와 같이 다른 채널의 사용 종료 후, 두 스테이션이 동시에 AIFS를 시작한다고 가정할 경우, 그리고 AIFS + CW이 서로 같아지도록 CW를 선택한다면 Case 1는 상대적으로 높은 비율로 발생할 수 있다.

본 연구개발에서는 와이파이를 이용한 실시간 시스템 서비스를 위해, 두 최악의 경우를 제거할 수 있는 알고리즘을 제공한다. 이 절에서는 두 가지 기법 제안하고 최대 큐선점 시간 \mathcal{R} 줄이고 최대 채널 선점 시간 Π 을 한계를 제한할 수 있는지 설명한다.

제 1 절 우선순위 큐를 이용한 접근법

AP에서 VO 우선순위를 사용하는 큐에서 큐를 점유하는 시간 \mathcal{R}_i 줄이기 위한 방법으로 기존의 FIFO 큐 알고리즘 대신, 우선순위 큐를 사용해서, 평균 큐 대기시간을 제안하고 줄인다. 또한 Rate Monotonic 알고리즘을 사용해서, 빈도수가 높은 스트림에 높은 우선순위를 부여한다.

$$\mathcal{R}_k = \sum_{j \in HP(k)} \left\lceil \frac{\mathcal{R}_k}{p_j} \right\rceil \cdot WCTT_j \quad (5.1)$$

$$\mathcal{R}_i^n = \sum_{j \in HP(k)} \left\lceil \frac{\mathcal{R}_k^{n-1}}{p_j} \right\rceil \cdot WCTT_j \quad (5.2)$$

식 5.1에서 $HP(k)$ 은 스트림 i 와 큐를 공유하는 모든 스트림 들 중, 우선순위가 높은 스트림의 집합을 가리킨다. 식 5.1는 특정 스트림이 다른 우선순위가 높은 다른 스트림에 의해서만 지연되는 경우의 큐 점유 지연을 계산한다. 따라서 우선순위가 가장 높은 스트림은 지연없이 큐를 점유할 수 있다.

제 2 절 충돌이 없는 백오프(Non-Conflict Back-Off, NCB)

본 연구개발에서는 제안하는 VO우선순위를 사용하는 스트림과 다른 우선순위를 사용하는 스트림 간의 충돌을 없애기 위해 다음과 같이 불변법칙을 정의한다. 그리고 CW값을 위배되지 않도록 CW의 값을 조정한다.

Definition 5.1. VO의 수행해야 할 AIFS + CW의 값은 항상 짹수이며, VI와 BE이 수행해야 할 AIFS + CW의 값은 항상 홀수이다.

모든 스테이션은 백오프를 위한 자신의 CW값을 알 수 있다. 본 연구개발이 제안하는 방법은 각 스테이션 수행할 AIFS값과 백오프 CW값의 합이 위의 정의의 원칙을 위배하지 않도록 CW 값에 1을 더하거나 뺄 수 있도록 허용하는 것이다. 예를 들어, BE의 경우는 AIFS 값이 3이다. 만약 백오프 실행 후, 남아있는 CW가 0이라해도 AIFS + CW의 값은 홀수이다. VI의 경우 AIFS 값이 2이다. 만약 백오프 실행 후, 남아있는 CW가 0이라면 AIFS + CW이 짹수가 되기 때문에 CW에 1을 더해서 AIFS + CW의 값이 홀수가 되도록 강제한다. 이제 이것이 어떠한 효과를 발휘하는지 알아보자.

2.1 VO-VI의 경우 문제의 해결

VO-VI 간의 문제는 기본적으로 두 우선순위를 가진 스테이션의 수행할 AIFS의 크기가 같기 때문에 생기는 문제다. 따라서 VO가 우선적으로 실행하기 위해서 VI가 CW(0)을 선택하지 못하도록, 즉 VI의 AIFS + CW의 크기를 3이상 되게 하면 문제가 해결될 수 있다. 즉 정의 5.1와 같이 VI의 AIFS + CW가 2보다 큰 홀수가 되게 해야 한다.

그림 5.1은 그림 4.2의 VO-VI에서 발생할 수 있는 No-BO-Decrement 대한 솔루션을 제공한다. 솔루션 시나로는 다음과 같다.

Stage 1: Station A는 ACK을 받은 후 데이터 전송을 종결한다.

Stage 2: (a) AP와 Station C는 각각 CW를 뽑는다. 가정에서 AP는 CW(3)을 Station C는 CW(0)을 뽑았음을 가정했다.

(b) VO 우선순위를 사용하는 AP의 AIFS의 값은 2, 그리고 VI 우선순위를 사용하는 Station C의 AIFS의 값 역시 2이기 때문에, AP와 Station C의 AIFS+CW의 값은 각각 5이고 2이다.

(c) AP의 AIFS+CW의 값이 정의 5.1를 위배하기 때문에, CW값에서 1을 빼어서 CW[3]을 삭제한다.

(d) Station C의 AIFS+CW의 값이 역시 정의 5.1를 위배했다. 하지만 삭제할 CW가 없기 때문에 CW값에 1을 더하여 CW[1]을 생성한다.

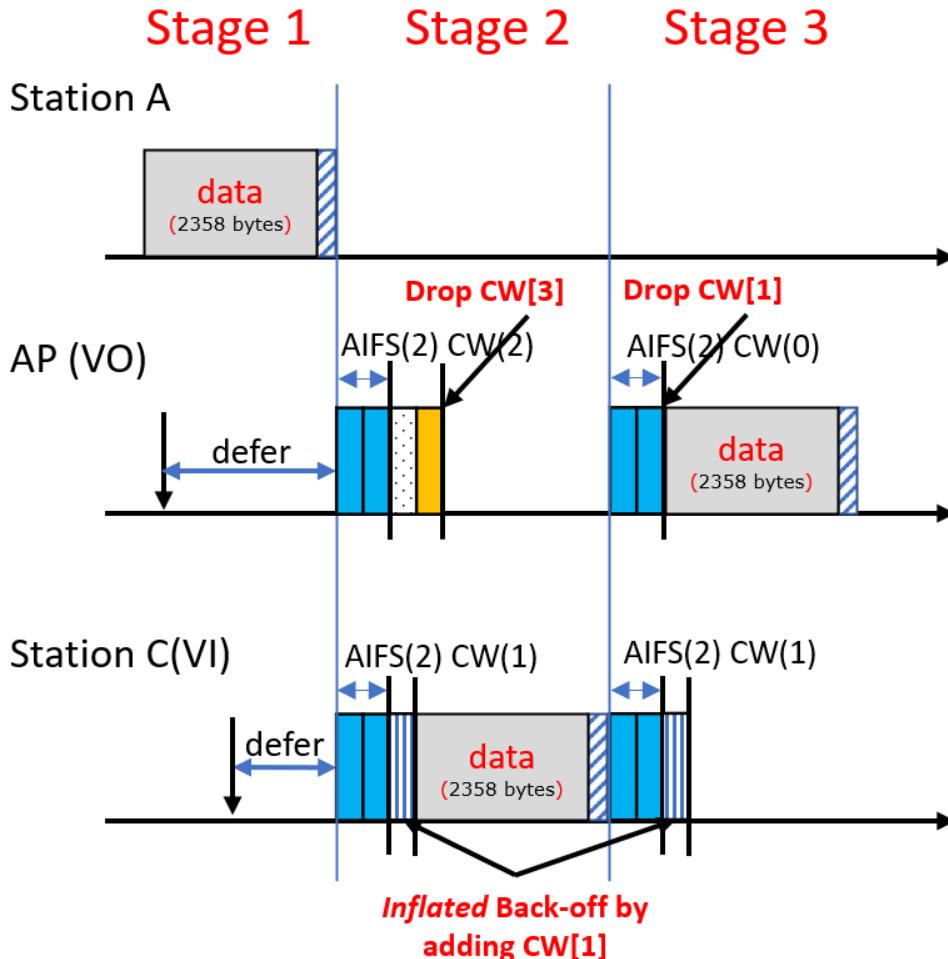


그림 5.1: CW 조정을 통한 No-BO-Decrement 해결

- (e) 이제 AP와 Station C의 AIFS+CW의 값은 각각 4이고 3이다. AP와 Station C는 모두 AIFS(2)을 수행하고 동시에 마친다.
- (f) AP는 백오프 CW[1]하고 Station C역시 추가된 CW[1]를 수행한다.
- (g) 그리고 Station C는 백오프를 마치고 채널을 선점하여 전송을 시작하고, 백오프가 남은 AP는 백오프 수행을 마친다.
- (h) 그런데 다음 백오프 재개 시 사용할 AP의 AIFS+CW 값이 3이다. 따라서 정의 5.1를 위배하고 CW 값이 여전히 남아 있기 때문에 CW값에서 1을 빼서 CW[2]를 삭제하여 CW은 0이 된다.
- (i) AP는 남아 있는 백오프 시간이 없어 재개하고, Station C은 새로운 CW(0)을 뽑지만 AIFS+CW의 값이 홀수이고 현재 삭제할 CW가 없기 때문에 CW에 1을 더하여 CW[1]을 생성한다. 따라서 Station C의 AIFS+CW의 값은 3이 된다.
- (j) AP와 Station C는 모두 AIFS(2)을 수행하고 동시에 마친다.
- (k) Station C는 새로 생성된 CW[1]을 수행해야 하고, AP는 더 이상 수행할 백오프가 없으므로 채널을 점유하여 전송을 개시한다.

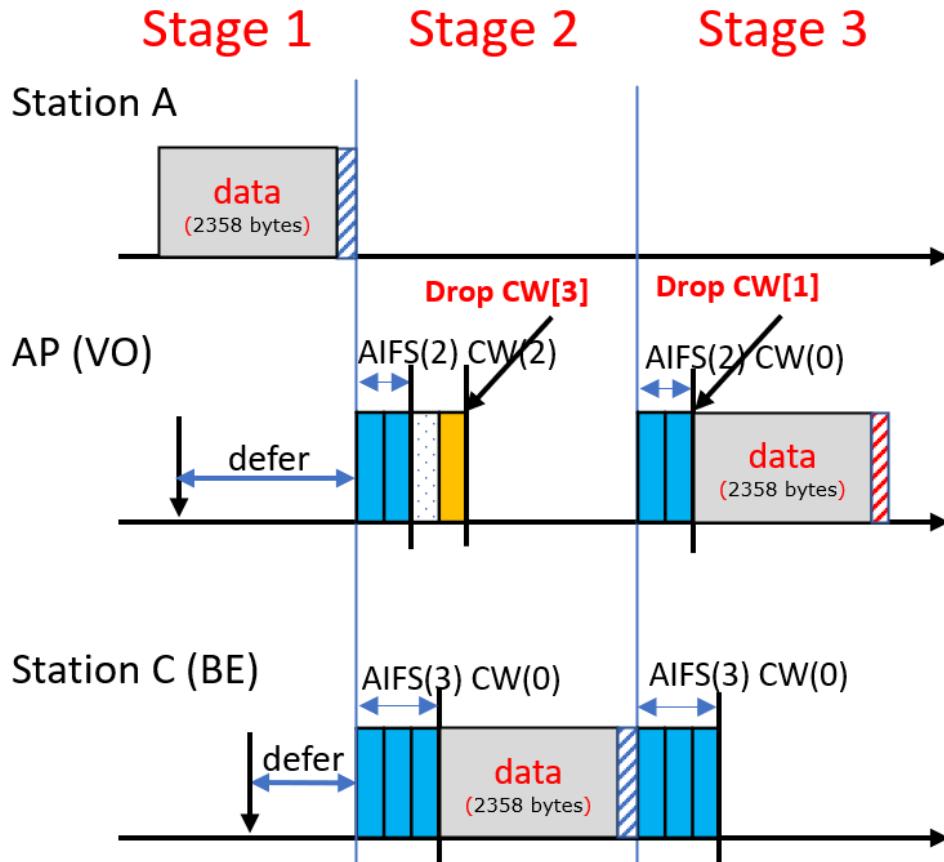


그림 5.2: CW 조정을 통한 VO-BE 문제 해결

2.2 VO-BE 경우 문제의 해결

그림 5.2는 우리의 제안에 따라 그림 4.3가 어떻게 해결되는지 보여준다. 그림 5.2의 시나리오는 다음과 같다. 그림 5.2는 그림 4.3와 같은 가정을 사용한다.

Stage 1: Station A는 ACK을 받은 후 데이터 전송을 종결한다.

Stage 2: (a) AP와 Station C는 각각 CW를 뽑는다. 가정에서 AP는 CW(3)을 Station C는 CW(0)을 뽑았음을 가정했다. VO 우선순위를 사용하는 AP의 AIFS값은 2, 그리고 BE 우선순위를 사용하는 Station C의 AIFS값은 3이기 때문에, AIFS+CW는 각각 5이고 3이다.

(b) AP의 AIFS+CW의 값이 정의 5.1를 위배하기 때문에, CW값에서 1을 빼서 CW[3]을 삭제한다.

(c) AP와 Station C는 각각의 AIFS(2)와 AIFS(3)을 수행한다.

(d) AP가 먼저 AFIS 수행을 마치고 백오프시간 중 CW[3]수행하고, 동시에 Station C는 AIFS[1]을 수행한다.

(e) AP가 백오프 CW[1]를 시작할 때, Station C는 CW(0)을 선택했기 때문에 Station C가 데이터 전송 시작한다. 그리고 AP의 백오프는 중단된다.

- (f) 다음 백오프의 재개에서 사용할 AP의 AIFS+CW의 값이 3으로 홀수이다. 따라서 다시 CW값에서 1을 빼서 CW[2] 삭제한다. 그 결과로 AP의 CW값은 0이 된다.
- (g) Station C이 전송은 수신자에게 ACK을 받고 종료된다.
- (h) AP와 Station C의 AIFS+CW 값은 각각 2이고 3이다. 따라서 AP의 전송이 성공적으로 수행된다.

두 시나리오를 통해서 AIFS가 같을 때에도 혹은 더 작을 때에도 VO의 백오프가 종료되는 시점이 다른 우선순위의 백오프 종료되는 시점과 절대 겹쳐지는 경우가 발생하지 않음을 볼 수 있다.

유의할 점은 AIFS + CW의 계산이 항상 백오프 이전에 수행되어야 한다는 점이다. 만약 각 스테이션은 매 프레임에 대해 초기 CW를 선택을 한 그 시점부터 AIFS + CW의 값이 정의 5.1를 만족하도록 조정해야 한다.

NCB 장점 이 연구개발에서는 이러한 기법을 충돌이 없는 백오프(Non-Conflict Back-off, NCB)라 하자. NCB는 백오프가 동시에 끝나는 현상 뿐만 아니라 백오프가 줄지 않은 NBD 현상도 함께 제거한다. 왜냐하면 홀수와 짝수 관계를 가진 두 AIFS+CW 값들은 작은 AIFS+CW값이 큰 AIFS+CW 값을 반드시 감소시기 때문이다. 또한 AIFS+CW값을 강제적으로 맞추기 위해서 전통적인 방법보다 더 빠르게 백오프 시간을 줄여나간다. 또한 백오프의 충돌이 사라지고 NBD 현상이 사라지면 결론적으로 백오프 시간은 제한될 수 있으며 제한된 시간을 찾는다면 실시간 시스템에서의 핵심인 신간적 결정성을 확보할 수 있다. 또한 NCB는 우선순위가 전도 되는 조건인 식 (4.4) 백오프 정책에 사용될 수 있다.

제 3 절 VO 스트림의 최대 채널대기 지연시간 계산

앞절에서는 NCB가 백오프 충돌 현상을 없애고, NBD 현상을 없애서 반드시 백오프가 끝날 수 있다는 점을 보였다. 이 절에서는 VO 우선순위를 사용하는 스트림이 겪을 수 있는 최대의 관련된 Π_k 을 계산됨을 보인다.

이 절에서는 NCB를 사용하는 와이파이의 QoS에서 VO우선순위를 가진 스트림의 매체 점유를 위한 **최대 채널대기 지연시간** Π_k 에 계산에 대해 논한다. 이 절에서는 우선순위 전도 조건 식 (4.4)을 만족하는 CW값의 조합을 찾고 그 중 최악의 경우를 계산한다. 이 연구개발에서는 CW값을 IEEE 802.11g/a/n을 기준으로 VO우선순위를 가진 스트림의 매체 점유를 위한 최악의 경우를 분석한다.

만약 전송이 실패하게 되는 경우, VO의 CW 값은 0~7에서, VI는 0~15, 그리고 BE는 0~1023 사이에서 랜덤하게 CW값을 선택한다.

VO의 CW 값(0~7)의 범위와 AIFS 크기(2)에 대하여, VI와 BE의 AIFS의 크기(각각 2와 3)를 고려한, 우선순위 전도 조건 식 (4.4)을 만족하는 VO, VI, 그리고 BE 스트림이 가질 수 있는 CW 값은 다음과 같이 제한된다. 즉 다음의 CW를 선택하는 경우 우선순위 전도가 발생할 수 있다는 것이다.

위의 값들 중에서 정의 5.1를 만족시키면서 스트림들이 가질 수 있는 CW 값은 다음과 같다.

| | AIFS | CW |
|--------|------|-----------------|
| VO 스트림 | 2 | 0,1,2,3,4,5,6,7 |
| VI 스트림 | 2 | 0,1,2,3,4,5,6 |
| BE 스트림 | 3 | 0,1,2,3,4,5 |

| | AIFS | CW |
|--------|------|-----------------|
| VO 스트림 | 2 | 2, 4, 6 |
| VI 스트림 | 2 | 1 , 3, 5 |
| BE 스트림 | 3 | 0 , 2, 4 |

NCB 기법을 사용하는 경우, 위의 CW의 조합 중의 최악의 경우는 VO 스트림이 가장 큰 CW를 선택하고, VI 스트림과 BE 스트림이 가장 작은 CW를 반복적으로 선택하는 경우이다. 즉 다음의 경우가 최악의 경우가 될 것이다.

- Worst-Case 1 (VO-VI 경우): VO 스트림이 CW(6)를 선택하고, VI 스트림이 CW(1)를 선택
- Worst-Case 2 (VO-BE 경우): VO 스트림이 CW(6)를 선택하고, BE 스트림이 CW(0)를 선택

다음 식은 우선순위 전도 조건 식 (4.4)을 만족하는 두 프레임, VO 프레임 i 와 VI 혹은 BE 프레임 j 에 대해서, 우선순위가 낮은 VI 혹은 BE 프레임이 동일한 작은 CW를 가지고 반복될 때, 우선순위가 백오프의 수행을 반복하는 횟수를 계산한다.

$$\mathcal{P} = \left\lfloor \frac{\text{CW}_i}{(\text{AIFS}_{\text{AC}(j)} + \text{CW}_j) - \text{AIFS}_{\text{AC}(i)} + 1} \right\rfloor \quad (5.3)$$

따라서 위의 두 최악의 경우는 모두 백오프의 수행을 반복하는 횟수 $\mathcal{P} = 3$ 으로 제한됨을 알 수 있다. 이렇게 NCB를 사용하면 VO 스트림 k 에 대해서 백오프의 수행을 반복하는 횟수가 제한되고, 따라서 최대 채널 대기 지연시간 (Π_k)를 구할 수 있고, 더불어 WCTT $_k$ 를 구할 수 있다.

제 6 장

결론

본 연구은 IEEE 802.11e의 QoS를 활용하여 와이파이 네트워크 슬라이스를 설계 구현하고, 다양한 시나리오를 구성하여 실시간 시스템에 대한 적합성 여부를 분석하였다. 또한 이를 바탕으로 최상위 VO우선순위를 실시간 주기적 스트림에 독점적으로 할당하였을 때에 VO의 주기적 스트림의 시간적 결정성을 보장하는 실시간 시스템 이론에 근거한 두 가지 알고리즘을 제안하였다.

본 연구에서는 실시간 시스템의 분석 방법을 활용하여, 최악의 경우의 전송 지연을 계산하는 프레임워크와 이 프레임워크를 활용하여 현재의 IEEE 802.11e의 QoS 기능을 분석하였다. 또한 현재의 QoS 기능의 시간적 비결정성의 핵심을 제시함으로 향후 IEEE 802.11e의 QoS의 시간적 결정성을 확보하는 토대를 제공하였다. 본 연구에서는 3가지 실험 세트로 총 12종의 실험을 실시하였으며, 현재의 IEEE 802.11e의 QoS구성으로 1) QoS의 우선순위 기반의 우선순위 전송의 성능을 확인하고 분석하고 2) 실시간 시스템의 시간적 결정성 보장의 한계를 확인하였다. QoS의 우선순위 기반 서비스의 경우, 무선상에서 AP스테이션으로 가는 업로드의 경우, 전체적인 평균 전송비율은 우선순위에 따라 전송되는 것을 확인할 수 있었다. 특히 우선순위가 짧은 스트림이 VO 우선순위를 사용할 경우 특히 그 시간적 결정성이 높은 것을 확인하였다. 하지만 최상위 우선순위인 VO 우선순위를 실시간 주기적 스트림이 전용으로 사용할 경우를 가정하여 AP가 다운스트림을 위해 이를 사용할 경우, 전체적인 평균 전송비율은 우선순위에 따라 전송되지만, 그 예외가 상당하다는 것을 알 수 있었다.

본 연구에서는 이러한 문제를 해결하기 위해서 VO의 주기적 스트림 그룹과 VI-BE-BK 스트림 그룹을 나누어 이 두 그룹간의 충돌을 정수론에 기반한 홀수-짝수의 이분법적으로 방법으로 CW를 할당함으로 두 그룹간의 백오프 종료 충돌을 근본적으로 해결하고 있다. 그리고 VI-BE-BK 스트림 간의 백오프 종료시의 충돌은 기존의 백오프 종료 충돌 방식을 활용하여 자원 활용도를 극대화한다.

향후 IEEE 802.11x에서 제공할 PCF가 중앙제어방식을 사용하기는 하지만 시스템에 대한 많은 변화를 동반하여야 하는 단점을 가지고 있어서 본 연구에서 제공하는 QoS가 그 복안으로 사용될 수 있을 것이라 사료된다.

부록 A

NS3 구현 결과

제 1 절 NS3 기본 실행

1.1 NS3 소개

네트워킹 연구 및 교육을 위한 오픈소스 시뮬레이터로, 패킷 데이터가 네트워크상에서 어떻게 작동하는지에 대한 모델을 제공하고 사용자가 실험을 수행할 수 있는 시뮬레이션 엔진을 제공한다. 또한, 최신 무선 네트워크 모델을 비롯한 다양한 유/무선 모델을 지원하며, Tracing 파일을 추출하여, Wireshark를 이용해 전송된 패킷들의 정보를 확인할 수 있다.

다음의 예제는 간단한 NS3의 코드로서 Node A와 B가 Point to Point Channel로 연결되어있고, UdpEcho 어플리케이션(보낸 패킷을 다시 돌아오는 어플리케이션)이 실행되는것을 시뮬레이션 한 것이다.

우선 Nodecontainer로 노드 2개를 생성한다.

```
NodeContainer nodes;
nodes.Create (2);
```

Listing A.1: Nodecontainer로 노드 2개를 생성

PointToPointHelper 이용해서 point to point channel 생성한다. 이때, SetDeviceAttribute 함수를 사용하여, DataRate를 5Mbps로 설정하고, SetChannelAttribute를 사용하여, Delay를 2ms로 설정한다.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

Listing A.2: PointToPointHelper 이용해서 point to point channel 생성

NetDeviceContainer로 NetDevice 생성하고, node에 추가해 준다.

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Listing A.3: NetDeviceContainer로 NetDevice 생성

InternetStackHelper 사용해서 인터넷 스택에 노드들을 넣어준 후, Ipv4AddressHelper로 노드에 설정할 ipv4 주소생성을 위해 게이트웨이와 서브넷 마스크를 설정한다. 이후 Netdevice에 ipv4 인터페이스를 등록한다. 이후 UdpEchoApplicationHelper을 이용하여, Node에서 동작할 Application을 설정한다.

```
InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Listing A.4: 노드를 스택에 삽입

예제에 사용된 Application은 Point to Point 채널을 이용해 보낸 패킷을 다시 돌아오는 Echo Server/Client를 구현한 것이다.

```
UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

부록 B

WIFI 슬라이싱 NS3 구현

이 절에서는 IEEE 802.11의 QoS 기능을 활용한 WIFI 슬라이싱 NS3 구현에 대한 설명을 제공한다.

우선 src/wifi/model/regular-wifi-mac.cc 파일에 Unpacked, SlicingMatch, AddTable 함수를 추가 했다. AddTable함수는 파일에 미리 저장되어 있는 패킷의 정보를 시뮬레이터 상에서 사용할 수 있도록 파일로 부터 구조체 벡터(Struct Vector)를 정의하는 부분이다. Ip주소의 시작점과 도착점은 Ipv4Address 로, 포트번호 들은 16진수 정수로, 프로토콜은 8진수 형태로 구조체를 생성한 후 구조체 벡터에 집어 넣는 방식으로 프로그래밍 하였다.

```
void
RegularWifiMac::AddTable()
{
    string str_buf;
    string str_temp;
    std::fstream fs;

    fs.open("tosList.csv",std::ios::in);
    while(!fs.eof()){
        struct tosLevel tosTemp;
        // 공백줄이 들어갔을경우.
        getline(fs, str_buf);
        if (str_buf.length() == 0){
            break;
        }
        std::stringstream temp(str_buf);

        for (int i=0; i<6; i++){
            getline(temp, str_temp, ',');
            if (i == 0){
```

```

    tosTemp.srcIp = Ipv4Address(str_temp.c_str());
} else if(i == 1{
    tosTemp.destIp = Ipv4Address(str_temp.c_str());
} else if (i == 2){
    uint16_t portTemp = std::stoi(str_temp);
    tosTemp.srcPort = portTemp;
} else if (i == 3){
    uint16_t portTemp = std::stoi(str_temp);
    tosTemp.destPort = portTemp;
} else if (i == 4){
    uint8_t protocolTemp = std::stoi(str_temp);
    tosTemp.Protocol = protocolTemp;
} else {
    int tidTemp = std::stoi(str_temp);
    tosTemp.tid = tidTemp;
}
tosTable.push_back(tosTemp);
}
fs.close();
}

```

Unpacked에서는 Packet을 인자로 받아서 Packet의 내용중 Slicing을 위해 필요한 정보들을 추출한다. 패킷의 meta data의 손상을 방지하기위해 packet을 p에 복사한다. 이후, p에서 논리링크제어 헤더(LLC Header)를 추출하여, GetType 함수를 사용하여, 해당 패킷의 프로토콜을 확인한다.(16진수 0800, 10진수 2048 일 경우 Ipv4, 16진수 0806, 10진수 2054 일 경우 ARP)

```

void RegularWifiMac::UnPacked(Ptr<Packet> packet) {

Ptr<Packet> p = packet->Copy();
LlcSnapHeader llc;
p->RemoveHeader(llc);
Ipv4Header iph;

```

Ipv4일 경우, p로 부터 Ip헤더를 추출하여, 시작 및 도착지점의 ip와 프로토콜을 추출한다. 프로토콜이 0x11일경우 UDP이고, 0x06일 경우 TCP이다, 이외의 경우는 생각하지 않았다. 이후에 각각 udp, tcp 헤더를 추출하고 그 헤더로 부터 시작지점과 도착지점의 포트번호를 추출하였다. 위에서 추출한 정보를 바탕으로 아래의 SlicingMatch함수에서 일치하는 정보가 있는지 없는지를 판단하였다.

```

if (llc.GetType() == 2048)

```

```
{
    // Ipv4
    p->RemoveHeader(iph);
    tosSetting.srcIp = iph.GetSource();
    tosSetting.destIp = iph.GetDestination();
    tosSetting.Protocol = iph.GetProtocol();
    if (tosSetting.Protocol == 0x11)
    {
        UdpHeader udph;
        p->RemoveHeader(udph);
        tosSetting.srcPort = udph.GetSourcePort();
        tosSetting.destPort = udph.GetDestinationPort();
    } else if (tosSetting.Protocol == 0x06){
        TcpHeader tcph;
        p->RemoveHeader(tcph);
        tosSetting.srcPort = tcph.GetSourcePort();
        tosSetting.destPort = tcph.GetDestinationPort();
    }
}
```

SlicingMatch 함수는 Unpacked 함수를 사용해서 추출한 정보와 구조체 벡터에 저장되어 있는 정보를 비교해서 미리 지정되어있는 tid값을 반환하도록 한다. 이때 반환된 tid값을 이용하여 우선순위 큐를 지정한다.

```
int
RegularWifiMac::SlicingMatch(Ptr<Packet> packet) {

    UnPacked(packet);
    int tableSize = tosTable.size();
    int matching[tableSize];
    for (int j = 0; j < tableSize; j++){
        matching[j] = 0;
    }

    for (int i = 0; i < tableSize; i++){

        if (tosSetting.Protocol != tosTable[i].Protocol){
            matching[i] = -1;
        }
    }
}
```

```

    }

    if (tosTable[i].srcIp != WildCardIP && tosSetting.srcIp != tosTable[i].
srcIp){
        matching[i] = -1;
    }

    if (tosSetting.destIp != tosTable[i].destIp){
        matching[i] = -1;
    }

    if (tosTable[i].srcPort != WildCardPort && tosSetting.srcPort != tosTable
[i].srcPort){
        matching[i] = -1;
    }

    if (tosSetting.destPort != tosTable[i].destPort){
        matching[i] = -1;
    }

}

for (int i = 0; i<tableSize;i++){
    NS_LOG_INFO(matching[i]);
    if (matching[i] == 0){
        return tosTable[i].tid;
    }
}

return 1;
}

```

같은 경로의 sta-wifi-mac.cc 의 Enqueue 부분에서 SlicingMatch 함수를 사용하여 tid값을 받은 뒤, SetQosTid함수를 사용해 패킷의 tid값을 패킷헤더에 대입한다.

```

if (GetQosSupported ())
{
    //Sanity check that the TID is valid
    NS_ASSERT (tid < 8);
    tid = SlicingMatch(packet);
    hdr.SetQosTid(tid);
    m_edca[QosUtilsMapTidToAc (tid)]->Queue (packet, hdr);
}

```

```

else
{
    m_txop->Queue (packet, hdr);
}

```

또한, sta-wifi-mac.cc 에서 적용한 부분을 AP에서도 적용시켜 주기 위해 ap-wifi-mac.cc의 Forward 함수에 동일하게 적용시켜 주었다.

```

if (GetQosSupported ())
{
    //Sanity check that the TID is valid
    NS_ASSERT (tid < 8);

    tid = SlicingMatch(packet);
    hdr.SetQosTid (tid);

    m_edca[QosUtilsMapTidToAc (tid)]->Queue (packet, hdr);
}

else
{
    m_txop->Queue (packet, hdr);
}

```