공학석사 학위논문

와이파이 네트워크에서 단말간 빠른 연결을 위한 스캐닝 기법

2021년 8월

서울대학교 대학원

전기정보공학부

최 혁 주

와이파이 네트워크에서 단말간 빠른 연결을 위한 스캐닝 기법

지도 교수 박 세 웅

이 논문을 공학석사 학위논문으로 제출함
2021년 8월

서울대학교 대학원
전기정보공학부
최 혁 주

최혁주의 공학석사 학위논문을 인준함
2021년 8월

위 원 장 김성철

부위원장 박세웅

위 원 이경한

초 록

현재의 와이파이 스캐닝 기법은 2.4 GHz와 5 GHz 대역의 모든 채널을 하나씩 확인하는 방법으로 시간이 오래 소요된다. 이 논문에서는 최근 들어 증가하게 된 듀얼 밴드 와이파이의 특징을 이용하여 스캐닝 시간을 줄이는 기법을 소개한다. 이 기법은 2.4 GHz 대역의 채널을 스캔할 때 AP가 5 GHz 대역의 정보를 같이 전달하여 스캐닝 채널 수를 줄이는 방법으로 스캐닝 시간을 단축시킨다.

또한, 이 기법을 실제 테스트베드에 구현하여 스캐닝 시간이 최대 75%까지 감소함을 확인 하였다.

주요어 : 와이파이 스캐닝, 듀얼 밴드, 멀티 채널 맥
학 번 : 2015-22814

목 차

표 목차

그림 목차

제 1 장 서 론

제 1 절 연구의 배경

와이파이는 무선 LAN 표준의 주류로 자리 잡게 되었으며, 현재 광범위하게 사용되고 있다. 따라서 와이파이 표준을 사용하는 기기 또한 기하급수적으로 증가 중이다.[①] 이러한 추세에 맞춰 와이파이 표준은 더 많은 기기를 대상으로 더 빠른 속도를 제공하기 위해 개정 중이며, 최근의 표준인 802.11n 이나 802.11ac 에서는 기존 표준과 다르게 5 GHz 대역까지 사용할 수 있게 되었다. 이를 통해 기존의 혼잡한 2.4 GHz 채널보다 더 많은 채널 수를 확보하게 됨으로써 기기들은 더 원활히 통신할 수 있게 되었다.

참고로, 오래 전 제정되었던 초기 표준인 802.11a에서도 5 GHz 대역을 사용한 적이 있으나, 지금은 거의 사장되어 사용되지 않는 표준이므로 이 논문에서는 논외로 하겠다.

2.4 GHz 대역에는 14개의 채널이 있고, 5 GHz 대역에는 50개 이상의 채널이 있다.[②] 채널 수가 많아질수록 기기들이 다양한 채널로 분산되기 때문에 각 기기들이 time domain 상에서 더 많은 영역을 차지할 수 있으며 collision 발생 가능성도 줄어들게 된다. 추가적으로 인접한 여러 채널을 묶어서 frequency domain을 더 넓게 사용하는 channel bonding 이 가능하기 때문에 결과적으로 더 빠른 통신이 가능해진다.

그러나, 이전에는 없었던 단점들도 동시에 생겨나는데, 그 중 하나가 채널 수의 증가로 인한 스캐닝 시간의 증가이다. 이는 와이파이 프로토콜의 특징으로부터 기인하는데, 와이파이 기기는 여러 채널을 한번에 스캔할 수 없으며, 여러 채널에 존재하는 개별 AP들의 존재 유무를 확인하기 위해 채널을 하나씩 스캔해야 한다. 그러므로 스캐닝 시간이 채널 수에 비례하여 증가하게 된다.

이를 확인할 수 있는 간단한 방법이 있다. 스마트폰이나 노트북과

---

① Dezfouli, Behnam and Esmaeelzadeh, Vahid and Sheth, Jaykumar and Radi, Marjan, “A review of software-defined WLANs: Architectures and central control mechanisms”, *IEEE Communications Surveys & Tutorials* (Vol. 21, 2018), pp. 431-463.

② https://en.wikipedia.org/wiki/List_of_WLAN_channels

같은 기기의 와이파이 기능을 껐다가 다시 키게 되는 경우 우선 2.4 GHz 대역의 AP들이 먼저 검색되고, 몇 초 후에 5 GHz 대역의 AP들이 검색되는 것을 알 수 있다. 이는 5 GHz 대역을 사용함으로써 채널 수가 증가함에 따른 자연스러운 현상이다.

스캐닝 시간이 길어지면 핸드오프 상황이나 voice over IP (VoIP) 등의 어플리케이션을 사용할 때 큰 문제가 된다. 따라서 스캐닝 시간은 짧아질수록 좋다. 이 논문에서는 이를 위한 기법을 제안하고 실제 구현과 실험을 통해 스캐닝 시간을 크게 단축시킬 수 있음을 확인했다.

제 2 절 기존 연구

스캐닝 시간이 길면 여러가지 불편한 점을 초래하기 때문에 이전부터 많은 연구들이 진행되어 왔다.

표준으로는 802.11k [③] 또는 802.11v [④] 와 같은 표준들이 제정되었는데, 이 표준에서는 AP들 간에 backhaul로 연결된 infrastructure basic service set 시나리오를 가정하였다. 이 경우 AP들간의 정보 교환을 통해 단말이 어느 AP로 handoff 하는 것이 가장 빠른 지 알 수 있으며, handoff 시간을 단축시킬 수 있다. 그러나 이러한 표준은 AP들 간에 유선 등으로 연결되어 있어야 사용할 수 있다는 한계점이 뚜렷하다. 이 표준들은 지하철이나 도서관 같은 기업용 무선 랜 환경에 적합하며, 일반 가정집과 같이 AP들이 분산되어 독립적으로 설치되고 동작하는 환경에는 적합하지 않다.

다른 연구에서는 active scanning 시에 direct probe 패킷을 이용해 스캐닝 시간을 단축시켰다.[⑤] 이는 probe request를 broadcasting 하는 대신 특정 AP를 지목하여 probe request를 보내고 response를 받는 방법으로 채널에 단말이 머무르는 시간을 단축시켜서 결과적으로 스캐닝 시간을 줄이게 된다. 그러나 이 경우 단말이 특정 채널에 어떤 AP들이 있는지를 미리 알고 있어야 한다는 한계점이 존재한다.

또 다른 연구에서는 단말이 움직이는 방향을 고려하여 handoff할

---

③ https://en.wikipedia.org/wiki/IEEE_802.11k-2008
④ https://en.wikipedia.org/wiki/IEEE_802.11v-2011
⑤ Purushothaman, Ilango and Roy, Sumit, “FastScan: a handoff scheme for voice over IEEE 802.11 WLANs,” *Wireless Networks* (Vol. 16, 2010), pp. 2049-2063.

AP를 더 빠르게 찾을 수 있는 방법을 제시하였다.[⑥] 그러나 이 연구는 단말이 가속도 센서 등을 이용해 이동 방향에 대한 정보를 아는 경우에만 적용할 수 있는 한계점이 있다.

마지막으로 이전의 handoff 기록을 통해 AP간에 neighbor graph를 구성하여 채널의 스캐닝 시간을 줄이는 기법에 대한 연구가 있다.[⑦] 그러나 이 연구는 추가적인 AP controller 등이 필요하다는 한계가 있다.

제 2 장 연구의 배경

제 1 절 기존 와이파이 스캐닝 기법

와이파이 단말이 AP를 스캔하기 위해 두 가지 방법이 있다. 이를 각각 passive scanning과 active scanning이라고 한다.

Passive scanning은 근처의 AP로부터 주기적으로 보내진 beacon signal을 단말이 수신하여 어떤 AP들이 존재하는지 확인하는 방법이다. 일반적으로 AP들은 102.4 ms마다 beacon signal을 보낸다. 따라서 passive scanning을 하려면 단말은 매 채널마다 최소한 102.4 ms동안 머물며 beacon signal들을 수신해야 한다. 5 GHz 대역에 특별히 지정된 dynamic frequency channel (DFS)의 경우 레이더 등의 장비들과의 혼선을 막기 위해 passive scanning만 가능하다.

Active scanning은 단말이 probe request 패킷을 보내고, 이를 수신한 주변 AP들로부터 오는 probe response를 수신하여 주변에 어떤 AP들이 있는지를 확인하는 방법이다. Active scanning에는 두가지 파라미터가 사용되는데, 각각 *minChannelTime* 과 *maxChannelTime* 이다. 단말은 채널에서 probe request를 보낸 후 *minChannelTime* 동안 대기하며, 그동안 어떠한 probe response도 받지 못한다면 채널에

[⑥] Han, Sangyup and Kim, Myungchul and Lee, Ben and Kang, Sungwon, "Fast Directional Handoff and lightweight retransmission protocol for enhancing multimedia quality in indoor WLANs," *Computer Networks* (Vol. 79, 2015), pp. 133-147.

[⑦] Shin, Minho and Mishra, Arunesh and Arbaugh, William A, "Improving the latency of 802.11 hand-offs using neighbor graphs," *Proceedings of the 2nd international conference on Mobile systems, applications, and services* (2004), pp. 70-83.

AP가 없는 것으로 판단하고 다음 채널로 이동한다. 만약 하나의 probe response라도 받게 된다면 단말은 *maxChannelTime*까지 대기하며 추가적인 probe response가 있는지 확인하게 된다. *minChannelTime*과 *maxChannelTime*은 각각의 단말마다 다르게 설정할 수 있지만 일반적으로 많이 쓰이는 값은 20 ms와 40 ms이다. 최근의 단말들은 기본적으로 빠른 스캐닝을 위해 active scanning을 사용하도록 설정되어 있다.

Active scanning과 passive scanning 두 경우 모두 동작 방식을 보면 채널 수에 비례하여 스캐닝 시간도 길어지게 된다.

제 2 절 듀얼 밴드 AP

제안 기법이 의도한 대로 동작하게 하기 위해서 5 GHz 대역에서만 동작하는 AP가 없거나 적어야 한다는 가정이 필요하다. 최근 출시된 AP들은 최신 표준을 지원하기 위해 5 GHz 대역을 사용하는데, 이러한 AP들은 또한 하위 호환성을 위해 2.4 GHz 대역도 사용하게 된다. 하지만 이러한 AP들이 실제 환경에서 어떻게 쓰이는지 검증을 하기 위해 사전 실험을 해보았다. 몇 가지 장소에서 근처의 모든 AP를 스캐닝 하는 방법으로 2.4 GHz 대역만 사용하는 AP, 5 GHz 대역만 사용하는 AP, 두 대역을 모두 사용하는 AP가 각각 얼마만큼의 비율로 있는지를 확인해 보았다.

측정은 아파트 가정집, 지하철역, 카페 세 장소에서 진행되었다. 측정에는 Mac OS가 탑재된 노트북을 이용하였는데, 두 대역의 radio를 모두 지원하는 와이파이 칩셋이 탑재되어 두 대역을 모두 스캐닝 할 수 있다. Mac OS의 내장 기능 중 시스템 리포트 기능을 이용해 주변 AP들에 대해 SSID, PHY 모드, BSSID, 채널, WLAN 타입, 보안 프로토콜, SNR 같은 정보들을 알아낼 수 있다.

두 대역을 모두 지원하는 AP인지는 기본적으로 SSID와 BSSID를 통해 알아낼 수 있다. 일반적으로 한 AP에서 두 대역을 모두 사용하는 경우 SSID의 패턴이 비슷하도록 설정된다. 예를 들자면 *myWiFi* 와 *myWiFi_5G* 와 같은 식이다. 이를 통해 두 SSID가 한 AP로부터 나온 것인지 추정이 가능하다. 만약 두 SSID가 서로 완전히 다른 패턴을 갖더라도 BSSID값을 통해 한 AP로부터 나온 것인지 확인이 가능하다.

BSSID는 각 AP의 고유한 값이며, 48비트의 16진수 값으로 되어 있다. 예를 들어 *72:5d:cc:7d:06:4d* 와 같은 형식이다. 앞쪽 24비트는

제조사별로 부여된 고유 값이며, 뒤쪽 24비트는 각각의 기기마다 할당된 고유 값이다. 두 대역을 모두 지원하는 AP의 경우 두 대역의 BSSID값이 1씩 차이가 나거나 거의 유사한 값을 갖게 된다. 아래 표 [2-1]은 실험 결과 이다.

	2.4GHz only	Dual band	5GHz only
Home	10	1	0
Subway	7	8	28
Cafe	4	5	0

표 [2-1]

다른 장소와 다르게 지하철 역에서 측정한 결과 5 GHz 대역에서만 동작하는 AP들이 다수 측정되었다. 그런데 이 경우, SSID를 확인해본 결과 전부 *KTWiFi* 또는 *Twifi zone*과 같은 infrastructure AP들이었다. 이러한 AP들은 이미 backhaul을 통해 유선 연결되어 있을 것으로 생각되며, 802.11k 또는 802.11v와 같은 표준을 적용시킬 수 있다. 그러므로 이러한 경우 이 논문에서 제안한 기법을 사용할 필요가 없어서 논 외로 하기로 한다. 가정집과 카페의 경우 5 GHz 대역만 사용하는 AP는 측정되지 않았으며, 제안 기법을 적용할 수 있다.

제 3 절 스캐닝 소요 시간

Passive scanning의 경우 각 채널마다 102.4 ms만큼의 고정된 시간이 필요하다. 그러므로 총 스캐닝 소요 시간은 *(채널 수) X 102.4 ms*라고 생각할 수 있다. Active scanning의 경우 *(채널 수) X 20 ms* 와 *(채널 수) X 40 ms* 사이의 시간이 소요된다.

두 경우 모두 채널 수가 스캐닝 소요 시간에 영향을 주는 중요한 변수이다. 현재의 와이파이 표준은 2.4 GHz 대역에서 13개의 채널을 사용한다. 그런데 최근 표준인 802.11n 또는 802.11ac에서는 5 GHz 대역의 수십 개의 채널을 추가적으로 사용한다. 채널 수가 몇 배로 증가함에 따라서 스캐닝 소요 시간도 비례해서 늘어나게 된다.

보이스톡과 같은 VoIP 어플리케이션의 경우 패킷 딜레이가 성능에 밀접한 연관이 있다. 예를 들어 end-to-end 딜레이가 200 ms를 초과하게 되면 사용자가 딜레이를 체감할 수 있는 정도가 되며, 이보다 더 커지는 경우에는 call drop 현상이 발생할 수 있다. 스캐닝을 하는 동안에는 단말이 다른 채널들을 확인해야 하므로 업링크와 다운링크

통신이 불가능하다.

스캐닝 동작은 일반적인 패킷 길이와 비교해서 상당히 긴 패킷 딜레이를 유발한다. 일반적인 와이파이 패킷의 길이는 수백 us 단위임을 생각해보면 수천개의 패킷을 보낼 수 있는 긴 시간임을 알 수 있다.

제 3 장 제안 기법

제 1 절 개요

제안 기법에서 스캐닝 소요 시간은 기본적으로 스캐닝 하는 채널의 수를 줄이는 방식으로 감소하게 된다. 이를 위해 각각의 듀얼 밴드 AP들은 2.4 GHz 대역에서 probe response를 보낼 때 현재 자신이 5 GHz 대역에서는 몇 번 채널을 사용하는지를 함께 담아서 보내게 된다. 이러한 추가적인 정보를 기재하기 위해 probe response frame 내의 vendor specific한 필드를 사용할 수 있다. 이 부분을 수정해서 구현하는 경우 표준에 위배되지 않는다. 단말은 이렇게 수정된 probe response 를 처리하여 5 GHz 대역에서는 어떤 채널만 선택적으로 스캔해야 하는지 알 수 있다. 아래 동작 방식을 설명한 그림에서 2.4 GHz 대역의 모든 채널을 스캔하는 것은 파란 박스로 표시해 두었으며, 5 GHz 대역에서 선택적으로 채널 스캔을 하는 것을 화살표로 표시해 두었다.

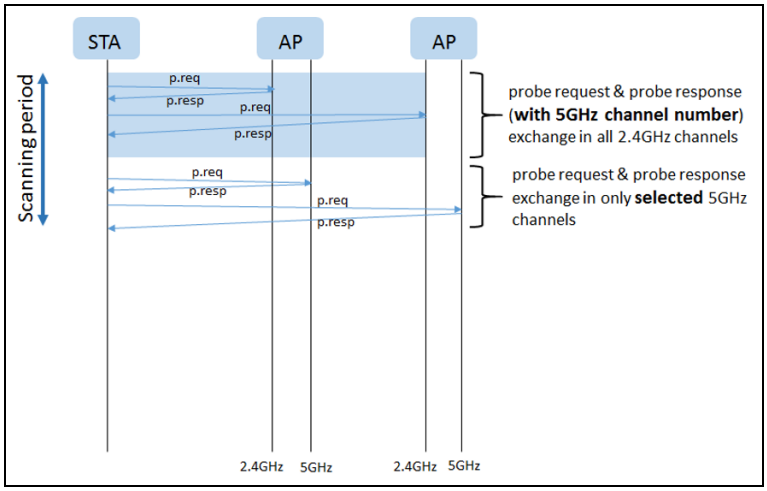

그림 [3-1]

제안 기법이 실제 환경에서 스캐닝 시간을 단축시키는지를 확인하기

위해 AP와 단말을 각각 테스트베드에 구현하여 실험을 하였다. AP는 Ubuntu 14.04 LTS와 리눅스 커널 4.4.6 버전을 기반으로 ath9k-htc 드라이버를 수정해 구현하였다. 단말은 Ubuntu 16.04와 backports 4.2.6 버전을 기반으로 ath9k 드라이버를 수정해 구현하였다. 구현을 위해 커널을 새로 빌드하기 위한 환경을 구축하고, 스캐닝 과정에 관여하는 함수들과 변수들을 확인 하였다. 그 이후 이러한 함수들이 호출되는 path를 구성하였으며, 제안 기법을 구현하기 위한 코드를 적절한 위치에 삽입하였다.

제 2 절 AP 구현

AP모드로 동작하도록 한 노트북에 위의 설정대로 드라이버를 수정하여 구현하였다. 듀얼 밴드 AP를 구성하기 위해 무선 랜 칩셋도 두개가 필요했다. 아래 그림과 같이 5 GHz 대역의 칩셋은 PCI 방식으로 노트북에 내장된 것을 사용했으며, 2.4 GHz 대역은 USB타입의 외장 무선 랜 모듈을 사용하였다.

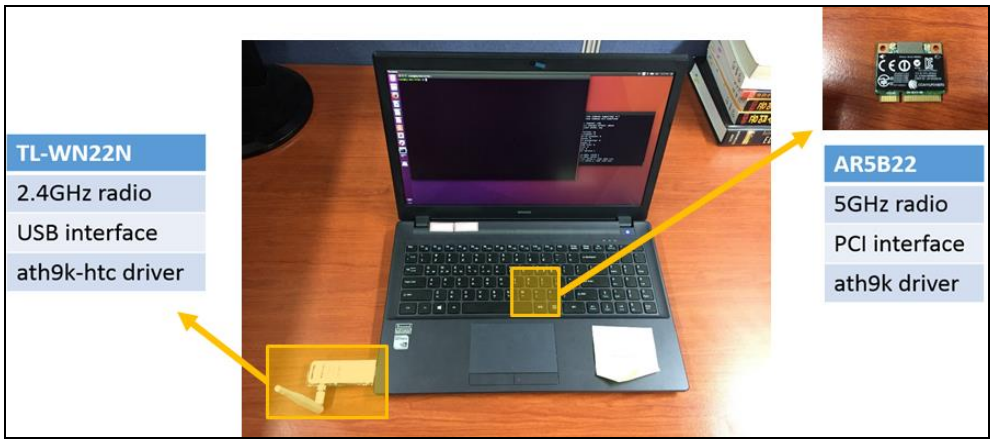

그림 [3-2]

각각의 칩셋은 ath9k와 ath9k-htc 드라이버를 사용한다. AP는 probe response frame을 수정하는데, 이는 드라이버 내에 *ath_9k_tx_mgmt* 함수에서 생성되고 발송된다. Probe response frame는 코드 내에 *ieee80211_mgmt*라는 이름의 구조체로 정의되어 있으며, 간략한 구조는 아래 그림과 같다.

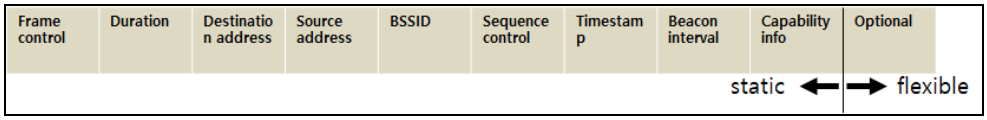

그림 [3-3]

Static으로 표시된 field 들은 frame 내에 반드시 포함되어야 하는 부분이며, 위치와 크기가 각각 할당되어 있다. Optional로 표시된 field는 반드시 포함될 필요는 없으며, 크기도 변동 가능하다. 드라이버 코드를 분석하여 이 field에 어떤 정보들이 포함되는지 알 수 있다. 초기 구현 계획으로는 optional field에 추가 데이터를 기재하려 하였으나, 실제 데이터를 삽입하기에는 크기와 위치를 설정하는 동작에 몇 가지 난점이 있었다. 따라서 우회 방법을 사용하여 구현 하였다.

관찰 결과 근처의 AP들은 beacon interval 값으로 항상 100 또는 102를 사용하는 것으로 확인되었다. Beacon interval field의 크기는 16비트로 할당되어 있으며, 이는 최대 65535까지의 큰 값을 담을 수 있는 값이다. 일반적으로 사용하는 beacon interval 값은 field의 크기에 비해 훨씬 작으므로 이 field에 값을 담아 보내기로 하였다. 따라서 AP가 probe response를 보낼 때 이 값을 *100 + (5 GHz 대역에서 동작하는 채널 번호)*로 수정하여 보내도록 하였다. 예를 들어 AP가 5 GHz 대역에서 40번 채널을 사용한다면 beacon interval 값을 140으로 수정하여 보내는 방식이다. 5 GHz 대역의 경우 채널 번호는 14부터 시작하여 사용하는 주파수 대역이 높아질수록 점점 커지게 된다. 우리가 구현한 단말의 경우 probe response frame을 열어서 beacon interval 값이 100 또는 102가 아니라면 동작을 수정한 AP로 판단하여 5 GHz 대역의 스캔할 채널 set에 반영하게 된다.

제 3 절 단말 구현

단말 구현은 크게 두 부분으로 나눠진다. 첫째로 스캐닝 동작이 시작되면 *scan_state* 라는 이름의 변수를 관리하는 함수들이 필요에 따라 서로를 호출하며 어느 채널을 스캔할 지 결정한다. 또한 스캔이 시작되면 *ieee80211_start_sw_scan* 함수가 실행되는데, 이 함수에 추후 스캐닝 시간을 재기 위한 타이머 코드를 삽입하였다. 둘째로 probe request 패킷을 보내고 probe response 패킷을 수신할 때 마다 필요한 정보를 받아 저장하는 함수들이 있다. 아래의 그림은 이 과정을 간단히 보여주는데, 수정한 부분은 빨간색 글씨로 기재하였다.

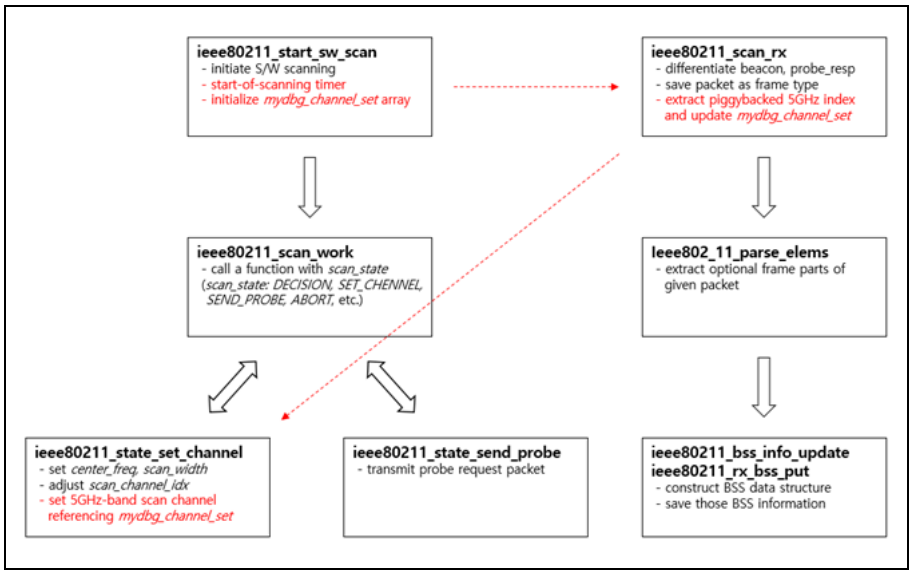

그림 [3-4]

구체적으로, 동작의 구현을 위해 *mydbg_channel_set* 이라는 이름의 boolean array를 전역 변수로 선언하였다. 이 array의 각 index는 스캔할 채널의 번호를 나타내게 된다. 스캐닝 동작이 시작되면 이 array에서 2.4 GHz 대역에 해당하는 채널은 true로, 5 GHz 대역에 해당하는 채널은 false로 초기화 된다. 그 후 2.4 GHz 대역의 스캐닝은 일반적인 방법으로 이루어지며, 단말은 5 GHz 대역에서 사용하는 채널 정보를 담은 probe response frame들을 수신하게 된다. 이러한 probe response frame에서 5 GHz 대역의 어느 채널들을 스캔하면 되는지 추출하여 *mydbg_channel_set* array의 값들을 실시간으로 수정하게 된다. 5 GHz 대역을 스캔할 때는 이 array에서 값이 true로 설정된 채널들만 선택적으로 스캔하게 된다.

제 4 장 성능 측정

제 1 절 환경 세팅

실제 환경에서의 성능 향상을 확인하기 위해 스캐닝 시간을 측정하는 실험을 수행하였다. 실험은 연구실과 중앙도서관 두 장소에서 각각 수행하였다. 연구실은 여러 AP들이 독립적으로 동작하는 환경이다. 실험 당시 32개의 BSS들이 검색되었으며, 어떤 채널들이 사용

중인지는 아래 표에 노란색으로 표시해 두었다.

2.4GHz			5GHz			
channel number	channel idx	center freq	channel number	channel idx	center freq	DFS
1	0	2412	36	14	5180	
2	1	2417	40	15	5200	
3	2	2422	44	16	5220	
4	3	2427	48	17	5240	
5	4	2432	52	18	5260	o
6	5	2437	56	19	5280	o
7	6	2442	60	20	5300	o
8	7	2447	64	21	5320	o
9	8	2452	100	22	5500	o
10	9	2457	104	23	5520	o
11	10	2462	108	24	5540	o
12	11	2467	112	25	5560	o
13	12	2472	116	26	5580	o
			120	27	5600	o
			124	28	5620	o
			128	29	5640	o
			132	30	5660	o
			136	31	5680	o
			140	32	5700	o
			149	33	5745	
			153	34	5765	
			157	35	5785	
			161	36	5805	

표 [4-1]

중앙도서관에는 152개의 BSS가 검색되었으며, 다양한 독립적 AP들과 인터넷 서비스 제공자들이 설치한 infrastructure AP들이 분산되어 설치되어 있었다. 마찬가지로 중앙도서관에서 사용되는 채널들은 아래 표에 노란색으로 표시해 두었다.

2.4GHz			5GHz			
channel number	channel idx	center freq	channel number	channel idx	center freq	DFS
1	0	2412	36	14	5180	
2	1	2417	40	15	5200	
3	2	2422	44	16	5220	
4	3	2427	48	17	5240	
5	4	2432	52	18	5260	o
6	5	2437	56	19	5280	o
7	6	2442	60	20	5300	o
8	7	2447	64	21	5320	o
9	8	2452	100	22	5500	o
10	9	2457	104	23	5520	o
11	10	2462	108	24	5540	o
12	11	2467	112	25	5560	o
13	12	2472	116	26	5580	o
			120	27	5600	o
			124	28	5620	o
			128	29	5640	o
			132	30	5660	o
			136	31	5680	o
			140	32	5700	o
			149	33	5745	
			153	34	5765	
			157	35	5785	
			161	36	5805	

표 [4-2]

성능 측정 실험에는 한가지 물리적인 한계점이 존재하는데, 기존에 설치되어 있는 AP는 소스 코드를 수정할 수 없기 때문에 제안 기법을 구현할 수 없다는 점이다. 이러한 제한적인 상황에서 스캐닝 시간을 측정하기 위해 사용되는 채널을 미리 확인한 후 해당되는 채널들의

array 값을 초기화 시에 수동으로 설정하였다. 이렇게 함으로써 기존에 설치되어 있는 AP들도 제안 기법에 따라 동작하는 것처럼 단말이 스캐닝하도록 할 수 있다.

각각의 실험에서 스캐닝은 5번씩 수행하였으며, 그 평균값을 측정하였다. 스캐닝 시간은 *ieee80211_start_sw_scan* 함수와 *ieee80211_scan_completed* 함수에 각각 타이머 코드를 삽입하여 두 함수가 호출되는 시간 차를 측정하여 확인하였다.

제 2 절 결과

두 환경에서의 스캐닝 시간은 아래 그래프와 같이 측정되었다.

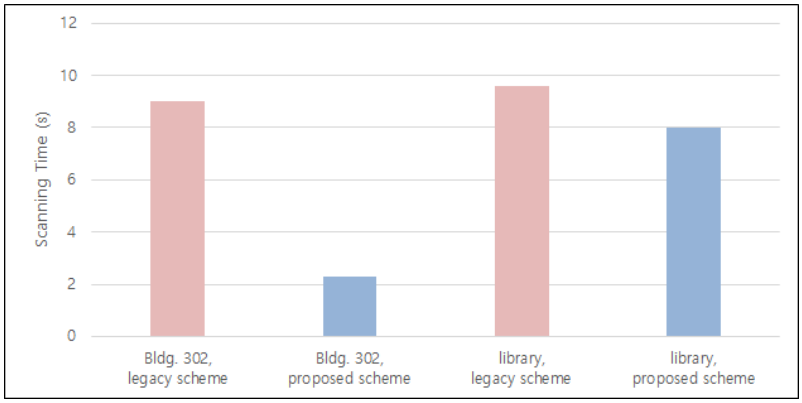

그림 [4-1]

제안 기법을 사용한 결과 기존 스캐닝 기법에 비해 스캐닝 시간이 두 환경에서 각각 74.8%와 16.5% 감소하였다. 연구실 환경에서는 제안 기법이 현재 사용되고 있는 5 GHz 대역의 채널만 스캐닝하여 대부분의 5 GHz 대역의 채널을 생략했기 때문에 큰 성능 향상 효과가 있었다. 5 GHz 대역에서 사용되는 채널의 수가 적을수록 성능 향상의 효과도 더 큰 것으로 확인 된다. 반면, 도서관의 경우 성능 향상은 있었지만 5 GHz 대역의 채널 중 절반 이상이 사용되고 있었기 때문에 연구실 환경에서만큼 큰 성능 향상 효과는 없었다.

제 5 장 결 론

제 1 절 결과 분석

실험 결과 연구실과 도서관 환경 모두에서 제안 기법의 성능 향상이 있음을 확인했다. 그러나, 두 경우에서 성능 향상의 폭은 크게 차이가 났다. 그 원인은 기존 환경에서 5 GHz대역의 채널이 얼마나 사용되는지에 따라 제안 기법에서 스캐닝을 생략하고 건너뛰는 채널 수가 달라지기 때문이다. 연구실 환경에서는 총 4개의 5 GHz 대역 채널이 사용되고 있었으며, 중앙도서관에서는 총 16개의 5 GHz 대역 채널이 사용되고 있었다. 일반적으로 AP들은 현재 regulatory domain의 DFS 채널은 사용하지 않도록 설정되기 때문에, 5 GHz 대역의 DFS 채널들은 주로 사용되지 않을 가능성이 높다. 그러므로 실제 작동 환경을 고려하면 제안 기법의 성능 향상 정도는 항상 어느 정도 존재할 것으로 생각된다.

제 2 절 후속 연구

위 실험과 별도로 단말이 아무 AP와도 association을 맺지 않은 상태에서 스캐닝 시간도 확인해보았다. 해당 스캐닝 시간은 연구실 환경에서 기존 기법과 제안 기법이 각각 3.27초, 0.88초 소모되었으며, 중앙도서관 환경에서는 3.11초, 2.55초 소모되어 association을 맺은 경우와 같은 경향성을 보이며 평균 시간은 짧아졌다. Association을 맺지 않은 단말의 스캐닝 시 관련 파라미터들은 association을 맺은 경우와 다르게 설정되기 때문에 직접적인 비교는 어렵지만, 이 경우에도 역시 제안 기법이 앞선 실험과 비슷한 비율로 성능 향상을 보임을 확인할 수 있다. 이 파라미터가 구체적으로 어떻게 달라지며 스캐닝 flow 또한 기존과 어떻게 변화하는지는 후속 연구를 통해 알아볼 수 있을 것이다.

이번 연구를 위한 AP 구현에서는 5 GHz 대역의 채널을 얼마나 사용하는지에 대한 정보를 probe response에 담아 전달할 때 beacon interval field를 사용하였다. 그러나 이러한 동작 수정은 제안 기법이 적용되지 않은 단말이 AP와 통신할 때 동작 이상을 초래할 가능성이 있다. 그러므로 기존 단말들에게 영향을 주지 않도록 probe response의 optional field를 이용하는 것이 바람직하다. 부록 1을 보면 probe response가 커널 내에서 어떤 형식으로 정의되어 있는지 알 수 있다. 하나의 구조체 정의로 다양한 management frame을 나타낼 수 있도록

공용체(union)을 사용하였는데, 종류와 상관없이 필수적으로 포함되는 값들은 *frame_control*, *duration*, *da*, *sa*, *bssid*, *seq_ctrl*이다. 그 아래에는 frame의 종류에 따라서 추가적인 field들이 정의되어 있고, probe response frame의 경우 *timestamp*, *beacon_int*, *capab_info*가 정의되어 있는 것을 알 수 있다. *capab_info* 뒤에 *variable[0]*라고 선언된 array field가 있는데, 이는 optional하게 활용할 수 있는 field로 보인다. 변수 타입은 u8로, 이는 8-bit unsigned int를 의미한다. 이 field는 array지만 size가 1만 있어도 최대 255까지의 정보를 담을 수 있기 때문에 5 GHz 대역의 채널 정보를 담기에는 충분한 크기이다. 따라서 beacon interval 대신 이 field를 수정하면 기존 단말들에 영향을 주지 않으면서도 제안 기법을 적용할 수 있을 것으로 보인다.

참고 문헌

1. Dezfouli, Behnam and Esmaeelzadeh, Vahid and Sheth, Jaykumar and Radi, Marjan, “A review of software-defined WLANs: Architectures and central control mechanisms”, IEEE Communications Surveys & Tutorials (Vol. 21, 2018), pp. 431-463.

2. https://en.wikipedia.org/wiki/List_of_WLAN_channels

3. https://en.wikipedia.org/wiki/IEEE_802.11k-2008

4. https://en.wikipedia.org/wiki/IEEE_802.11v-2011

5. Purushothaman, Ilango and Roy, Sumit, “FastScan: a handoff scheme for voice over IEEE 802.11 WLANs,” Wireless Networks (Vol. 16, 2010), pp. 2049-2063.

6. Han, Sangyup and Kim, Myungchul and Lee, Ben and Kang, Sungwon, “Fast Directional Handoff and lightweight retransmission protocol for enhancing multimedia quality in indoor WLANs,” Computer Networks (Vol. 79, 2015), pp. 133-147.

7. Shin, Minho and Mishra, Arunesh and Arbaugh, William A, “Improving the latency of 802.11 hand-offs using neighbor graphs,” Proceedings of the 2nd international conference on Mobile systems, applications, and services (2004), pp. 70-83.

부 록

부록 1

ieee80211_mgmt 구조체는 커널 폴더 내에 위치해 있는데, 그 위치는 source/include/linux/ieee80211.h 이다.

```
struct ieee80211_mgmt {
        __le16 frame_control;
        __le16 duration;
        u8 da[ETH_ALEN];
        u8 sa[ETH_ALEN];
        u8 bssid[ETH_ALEN];
        __le16 seq_ctrl;
        union {
                struct {
                        __le16 auth_alg;
                        __le16 auth_transaction;
                        __le16 status_code;
                        /* possibly followed by Challenge text */
                        u8 variable[0];
                } __packed auth;
                struct {
                        __le16 reason_code;
                } __packed deauth;
                struct {
                        __le16 capab_info;
                        __le16 listen_interval;
                        /* followed by SSID and Supported rates */
                        u8 variable[0];
                } __packed assoc_req;
                struct {
                        __le16 capab_info;
                        __le16 status_code;
                        __le16 aid;
                        /* followed by Supported rates */
                        u8 variable[0];
                } __packed assoc_resp, reassoc_resp;
                struct {
                        __le16 capab_info;
                        __le16 listen_interval;
                        u8 current_ap[ETH_ALEN];
                        /* followed by SSID and Supported rates */
                        u8 variable[0];
                } __packed reassoc_req;
                struct {
                        __le16 reason_code;
                } __packed disassoc;
                struct {
                        __le64 timestamp;
                        __le16 beacon_int;
                        __le16 capab_info;
                        /* followed by some of SSID, Supported rates,
                         * FH Params, DS Params, CF Params, IBSS Params, TIM */
                        u8 variable[0];
                } __packed beacon;
                struct {
                        /* only variable items: SSID, Supported rates */
                        u8 variable[0];
                } __packed probe_req;
                struct {
                        __le64 timestamp;
```

```
                __le16 beacon_int;
                __le16 capab_info;
                /* followed by some of SSID, Supported rates,
                 * FH Params, DS Params, CF Params, IBSS Params */
                u8 variable[0];
        } __packed probe_resp;
        struct {
                u8 category;
                union {
                        struct {
                                u8 action_code;
                                u8 dialog_token;
                                u8 status_code;
                                u8 variable[0];
                        } __packed wme_action;
                        struct{
                                u8 action_code;
                                u8 variable[0];
                        } __packed chan_switch;
                        struct{
                                u8 action_code;
                                struct ieee80211_ext_chansw_ie
data;
                                u8 variable[0];
                        } __packed ext_chan_switch;
                        struct{
                                u8 action_code;
                                u8 dialog_token;
                                u8 element_id;
                                u8 length;
                                struct ieee80211_msrment_ie
msr_elem;
                        } __packed measurement;
                        struct{
                                u8 action_code;
                                u8 dialog_token;
                                __le16 capab;
                                __le16 timeout;
                                __le16 start_seq_num;
                        } __packed addba_req;
                        struct{
                                u8 action_code;
                                u8 dialog_token;
                                __le16 status;
                                __le16 capab;
                                __le16 timeout;
                        } __packed addba_resp;
                        struct{
                                u8 action_code;
                                __le16 params;
                                __le16 reason_code;
                        } __packed delba;
                        struct {
                                u8 action_code;
                                u8 variable[0];
                        } __packed self_prot;
                        struct{
                                u8 action_code;
                                u8 variable[0];
                        } __packed mesh_action;
                        struct {
                                u8 action;
                                u8
trans_id[WLAN_SA_QUERY_TR_ID_LEN];
                        } __packed sa_query;
                        struct {
                                u8 action;
```

```
                        u8 smps_control;
                } __packed ht_smps;
                struct {
                        u8 action_code;
                        u8 chanwidth;
                } __packed ht_notify_cw;
                struct {
                        u8 action_code;
                        u8 dialog_token;
                        __le16 capability;
                        u8 variable[0];
                } __packed tdls_discover_resp;
                struct {
                        u8 action_code;
                        u8 operating_mode;
                } __packed vht_opmode_notif;
                struct {
                        u8 action_code;
                        u8 dialog_token;
                        u8 tpc_elem_id;
                        u8 tpc_elem_length;
                        struct ieee80211_tpc_report_ie tpc;
                } __packed tpc_report;
        } u;
    } __packed action;
  } u;
} __packed __aligned(2);
```

Abstract

Fast scanning scheme between devices in Wi-Fi network

최 혁 주(Hyeog Ju Choi)
공과대학 전기정보공학부(Department of Electrical and Computer Engineering)
The Graduate School
Seoul National University

Existing Wi-Fi scanning techniques take a long time because all channels in the 2.4 GHz and 5 GHz bands are scanned one by one. We propose a scheme to reduce the scanning time by utilizing the characteristics of the dual-band AP. When the device scans the 2.4 GHz band, the dual-band AP delivers the channel usage information of the 5 GHz band together. Through this, the scanning channel set can be reduced and the scanning time can be shortened. By implementing this technique on a testbed, we experimented to see if there is any improvement in performance in a real environment. As a result of the experiment, it was confirmed that the scanning time was reduced by up to 75%

Keywords : Wi-Fi Scanning, Dual band, Multi channel MAC
Student Number : 2015-22814