SDL을 이용한 802.11 Legacy MAC 프로토콜 설계 및 구현

이진호[0], 이성훈, 이형근
광운대학교 컴퓨터 공학과
{zinbro[0], sunghune, hklee}@kw.ac.kr

The Design and Implementation of IEEE 802.11 MAC Protocol

Jin-ho Lee[0] , Sung-hune Lee, Hyung-keun Lee
Dept. of CE, Kwang woon Univ.

요 약

현재의 소프트웨어 개발은 구현 중심의 개발에서 설계 중심의 개발로 바뀌고 있다. 설계 중심의 개발은 구현 프로그램의 시스템 개발 시간을 크게 단축할 수 있고, 정형화된 명세를 검증함으로써 설계와 구현의 일관성을 유지할 수 있으며 유지 보수가 효율적이다. 이러한 설계 중심의 개발에 있어서 SDL (Specification and Description Language)은 표준으로 재정된 기술을 구현하는데 있어 어떠한 기준 명세라 할 수 있다. 하지만 이러한 SDL도 대다수의 사람이 쉽게 알아보기에는 어려움이 있기 때문에 좀 더 높은 범용성을 위해 본 논문에서는 엔지니어들이 보편적으로 많이 사용하는 언어인 c언어를 이용해 SDL을 직접적으로 구현 해 보았는데 점점 더 발전해가는 Wireless LAN의 표준인 IEEE 802.11의 spec에 명시되어 있는 SDL을 기반으로 MAC 부분을 구현하였다.

1. 서 론

지금까지 무선 통신망은 수 Mbps에서 수십 Mbps의 데이터 전송 속도의 고속화에 중점을 두고 발전해 왔다. 그 중 Wi-Fi(IEEE802.11x)는 IEEE 802.11a, IEEE 802.11b, IEEE 802.11g으로 발전하면서 전송 속도에 발전이 있었는데, MAC(Medium Access Control)부분에는 큰 변화가 없었다. 이 MAC은 상용화는 거의 되지 않았지만 4개의 AC(acce ss category) Queue를 가지고 QoS지원을 목적으로 개발 된IEEE 802.11e와 security 강화를 목적으로 개발 된 IEEE 802.11i가 나오면서 MAC에 여러 가지 변화가 생겼다. 하지만 이 경우도 wireless channel을 점유하는 방식 등 기본적인 기능은 같고 QoS 부분 혹은 Security 부분이 추가된 것이고 거기에 최근에 표준화가 진행 중인 IEEE 802.11n의 경우는 앞서 말한 이 두 가지 프로토콜을 모두 포함하고 있는 것뿐이기 때문에 WLAN의 MAC을 이해하기 위해서는 가장 먼저 표준으로 된 IEEE 802.11 legacy MAC을 이해하여야 한다.

MAC의 기본적인 기능인 채널 할당 처리와 프로토콜 데이터 유닛(PDU), 주소할당, 프레임 형성, 에러검사, 프레임 조각화(fragment) 및 조립(defragment)은 IEEE 802.11 legacy MAC의 기능을 그대로 사용하고 있기 때문이다.

※ 본 연구 결과물은 2008년도「서울시 산학연 협력사업」의 「나노IP/SoC설계기술혁신사업단」의 지원으로 이루어졌습니다.

본 논문에서는 IEEE 802.11 표준안의 MAC 프로토콜을 표준안에 포함된 SDL을 참고해 c 코드로 구현하는 과정을 기술 하였다. 제 2장에서 IEEE 802.11 MAC의 개요와 기본 개념을 설명하고, 제 3장에서 SDL에 대한 간략한 개요 및 이 SDL 이용한 MAC의 C코드 구현에 대하여 기술 한다. 마지막으로 제 4장에서는 결론으로 끝을 맺는다.

2. IEEE 802.11 MAC Protocol

IEEE 802.11 MAC 프로토콜 은 정합 기능이라는 논리적인 기능을 기초로 하는 프로토콜이다. 기본적인 구성인 BSS(Basic Service Set)는 AP를 포함하는 여러 개의 노드들로 이루어진 하나의 집합을 의미하며 ESS(Extended Service Set)는 BSS들을 분산 시스템(distributed system)에 의해 연결된 하나의 집합을 의미한다. BSS 내의 노드들은 무선 매체를 이용하여 서로 프레임을 주고받게 된다. 이때 앞에서 언급한 정합 기능은 통신을 위한 규약을 정의하고 있는데 기본적으로 DCF(Distributed Coordination Function)와 선택적으로 제공되는 PCF(Po int Coordination Function) 두가지의 기능으로 나눌 수 있다. DCF는 CSMA/CA(Carrier Sense Multiple Access/ Collision Avoidance)프로토콜을 사용한 경쟁적 채널 할당 방식이고 PCF는 매체를 점유할 수 있는 권한을 선택적으로 노드들에 할당해 주는 방식은

polling-and-respond 방식을 사용하고 있다. 이러한 채널 할당 처리 외에도 MAC 부계층은 프로토콜 데이터 유닛(PDU), 주소할당(addressing),프레임 형성, 에러검사, 조각화(fragment) 와 조립(defragment)을 수행한다.[1]

2.1 Disrtibuted Coordination Function(DCF) 방식

IEEE 802.11 MAC의 기본적인 접속 방법은 CSMA/CA를 이용한 비동기 데이터 최선 전송 모드의 DCF이다. DCF는 모든 스테이션에서 수행되며 IBSS(Independent BSS)와 기반 통신망에서도 사용된다. 각 노드는 항상 매체의 신호를 감지하고 있다가 적어도 DIFS 시간만큼 매체가 쉬고 있는 상태(idle)라면 데이터를 전송하려하는 노드는 임의의 지연시간(random backoff interval)후에 전송을 시도 할 수 있는데, 이는 두 개 이상의 노드가 동시에 매체가 DIFS 시간만큼 idle이라는 것을 감지하고 바로 전송을 시작하는 경우 반드시 충돌이 발생하기 때문이며, 이를 충돌 방지(collision avoidance)라고 한다. DIFS동안 채널이 비어있었다는 것을 감지한 여러 노드들은 자신이 가진 임의 지연시간 값(random backoff coun ter)을 줄여나가게 되고 이 값을 가장먼저 0으로 가지게 된 노드가 먼저 프레임을 전송하게 된다. 이때 아직 카운터가 0이 아니어서 전송하지 못한 노드들은 줄여나가던 카운터 값을 그대로 가지고 있다가 현재 프레임 전송중인 노드의 전송이 끝나면 다시 그 카운터의 값을 하나씩 감소시켜 0 이 되면 데이터를 전송할 수 있는 권한이 생긴다. 그림 1은 DCF 전송의 예를 보여주는데 여기서 x축은 시간을 의미한다.

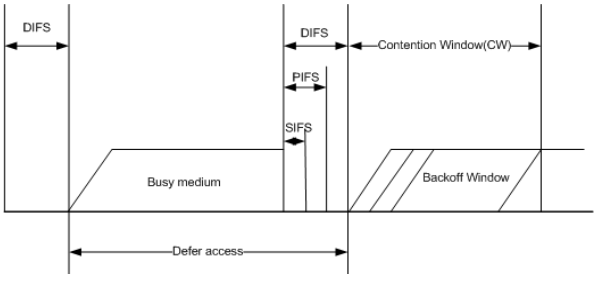

그림 1. DCF 채널 접근

2.2 Point Coordination Function(PCF)

IEEE 802.11 MAC의 또다른 접속방법인 PCF는 infra structure 네트워크에서만 사용되며 음성이나 화상처럼 지연에 민감한 전송에 대한 비경쟁(contention free) 서비스를 제공한다 PCF는 폴링과 유사한 방법으로 PC(Point coordinator)를 사용하여 채널을 사용할 노드를 결정하여 채널 사용권을 부여한다. 비경쟁 구간(contention free period)이 시작되면 AP가 전송한 Beacon 프레임에 의해 노드들의 채널 사용이 금지되고, 폴링 프레임을 수신한 노드만이 패킷을 전송할 수 있다. PCF의 동작과정을 그림 2에 나타내었다. 여기서도 그림1과 마찬가지로 x축은 시간을 의미한다.

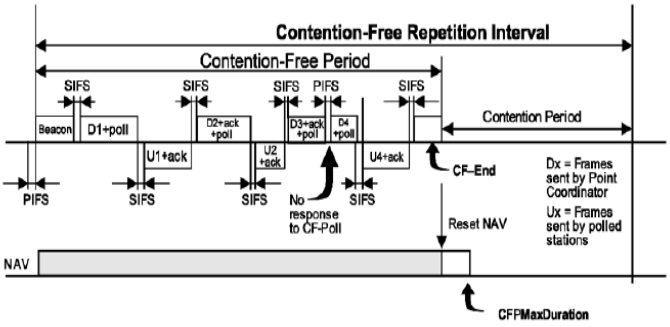

그림 2. PCF 프레임 전송 예

3. SDL 기반의 IEEE 802.11 MAC Prootocol 구현

SDL은 시스템의 동작을 확장된 유한 상태 기계 개념을 기초로 정의 하고있는데, 흐름도와 유사한 형태의 그래픽 다이어그램으로 전체 시스템을 표현하는 SDL/GR(Graphical Representation)과 프로그래밍 언어와 비슷한 형태로 SDL/GR에 상응하는 SDL/PR(Program Representation)의 두 가지 형태로 이루어진다. 이중에서 본 논문에서 구현한 IEEE 802.11 MAC은 SDL/GR로 표현되어 있는데 SDL/GR은 시스템의 구조와 행위를 명확하게 보여주고 제어와 자료의 흐름을 시각화하므로 유한 상태 기계에 효과적으로 표현되는 통신 시스템의 명세와 설계에 적합하다. 그림 3은 SDL로 표현된 시스템의 구성요소들이 계층적으로 모듈화됨을 보여준다.

프로세스는 자신의 동작을 수행하고 자신의 지역 속성을 갖는 동적인 개체로서 다른 프로세스들과 시그널을 통해

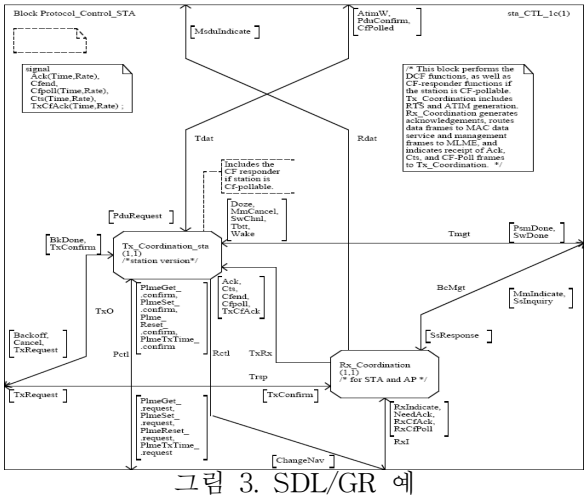

그림 3. SDL/GR 예

정보를 상호 교류한다. 또한 프로세스는 프로시저를 호출 할 수 있다.

전문화(sepcialization)와 상속(inheritance)개념이 블록, 프로세스, 데이터 타입 등 SDL 개념 전반에서 사용 될 수 있게 함으로써 객체 지향 설계를 지원한다. 이러한 객체 지향 설계는 압축된 시스템 설계를 가능하게 하며, 구성 요소를 재 사용할 수 있게 하여 시스템 유지에 필요한 수고를 줄인다.[2]

IEEE 802.11 standard 의 2006년 revision문서의 annex C를 보면 MAC의 SDL 설명이 나와 있다. 가장 먼저 MAC 전체 블록도가 나오고 그 안에서 각 블록별로 프로세스들 흐름이 나온뒤 그 프로세스들에 설명이 나오는 순서로 되어 있는데 MAC의 매커니즘을 생각해보면 데이터의 흐름이 LLC Data로부터 시작이 되거나 혹은 PHY에서부터 시작되는 것을 알 수 있기 때문에 SDL이 시그널 별로 동작하는 것으로 설명이 되어 있다고 해도 꼭 병렬적으로 시그널이 한꺼번에 여러 프로세스에 전달이 되어 처리가 되지 않는다는 것을 알 수 있다. 하지만 그렇다고 해서 순차적인 처리를 하도록 C코드를 작성하면 내부 데이터 흐름이 LLC Data에서 시작될 때와 PHY에서부터 시작되는 것을 함께 처리 하지 못하게 된다. 그래서 이러한 문제를 해결하기 위해 스레드(thread)를 이용한 C 코드를 작성 하였다. 그리고 최종적으로는 하드웨어로 구현하여 함께 연동하는 것을 목표로 했기 때문에 임베디드 측면을 고려해 linux환경에서 C 코드를 작성하였다.

가장먼저 나오는 MAC_Data_Service 블록을 예로 살펴보면 MSD U_from_LLC 와 MSDU_to_LLC 두가지 프로세스로 나뉜다.

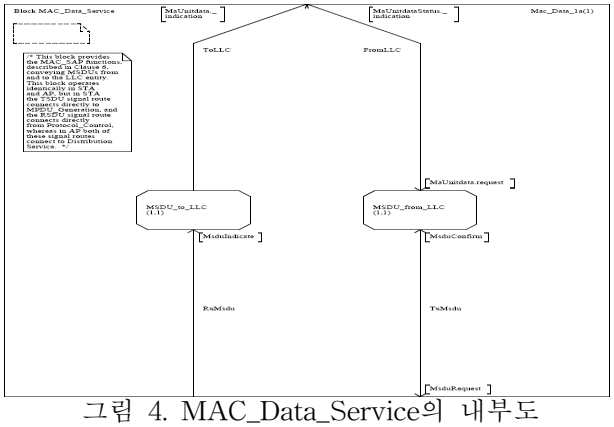

그림 4. MAC_Data_Service의 내부도

그리고 그림 5에 나와있는 것처럼 MSDU_to_LLC프로세스의 내부 SDL을 보면 MSDU indicate 시그널을 받으면 그 시그널의 정보를 바탕으로 몇 가지 데이터 처리를 한

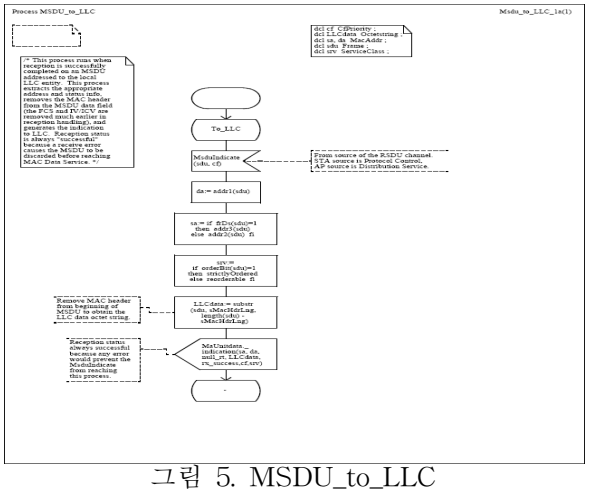

그림 5. MSDU_to_LLC

뒤 MaUnitData indication 시그널을 발생시키는 것을 알 수 있다. 이것은 이 프로세스는 Msdu indicate 시그널이 발생(입력)이 되어야만 실행이 된다는 뜻이다. 이 프로세스뿐만 아니라 다른 프로세스들도 이렇게 어떤 시그널이 입력되어야만 그 프로세스가 동작하는 경우가 대부분 이다. 이런 경우는 구현을 할 때 처음 시작되는 시그널을 이벤트라 생각하고 그 이벤트가 발생 했을 때 마다 해당 프로세스의 동작을 하게 만들면 된다. 그런데 이러한 프로세스가 하나만 있는 것이 아니고 각 블록별로 한 개 이상의 프로세스가 있기 때문에 이런 시그널 이벤트 기반의 프로세스를 하나의 thread로 만들어야 한다. 리눅스 기반으로 프로그램을 구현하였기 때문에 posix thread를 사용하였는데, 하나의 메모리를 공유하기 때문에 mutex를 사용하였고 프로세스 간 시그널 이벤트를 pthread_cond 값(condition value)을 사용해 구현하였다.[3]

pthread_mutex_t	mutex 객체
pthread_cond_t	condition 객체

표 1. 사용하는 thread 객체

즉 처음 thread를 생성할 때 condition 이벤트가 발생하기 전까지 들어올 때 까지는 대기하고 이 이벤트가 발생했을 때 이후의 동작을 하게 하면 SDL에서 설명하는 프로세스를 C로 구현하는 방법이 되는 것이다.

예로 들어 MaUnitData Request 시그널과 관련된 프로세스를 작성한다면. 먼저 MaUnitdataRequest의 구조체에 임의의 멤버(예: MaUnitdataRequest.value)의 초기값으로 0을 넣어놓고 그 thread를 생성하는 함수의 처음부분에 teread의 mutex에 대한 부분을 먼저 해결 한 뒤 임의의 멤버(위의 예에서 MaUnitdataRequest.value) 값을 검사해 그것이 초기 값이라면 while문을 이용 해 cond 이벤트가 발생할 때 까지 대기하도록 만드는 것이다. 아래는 실제코드이다.

```
int s_MaUnitdata_request;
s_MaUnitdata_request=
pthread_mutex_lock(&MaUnitdata_request.mutex);

while (MaUnitdata_request.value == 0)
s_MaUnitdata_request=
 pthread_cond_wait(&MaUnitdata_request.cond,
```

그림 6. 실제 코드의 한부분

```
typedef struct MaUnitdata_request_tag
{
        pthread_mutex_t     mutex;
        pthread_cond_t          cond;
        unsigned char           sa[6];
        unsigned char           da[6];
        Routing                 rt;
        char*                   LLCdata;
        CfPriority          cf;
        ServiceClass            srv;
        int                     value;
} MaUnitdata_request_t;
```

그림 7. 구조체의 선언부분

이렇게 각 프로세스를 시그널 이벤트로 분류하고 각 시그널별로 따로 tread를 만들어 놓고 필요 부분에서 시그널을 발생 시키는 것으로 그에 맞는 thread의 cond 값만 발생시키면 결국 원하는 부분의 프로세스만 동작하게 구현 할 수 있다.

5.결론

본 논문에서는 형식언어인 SDL을 기반으로 IEEE 802.11 MAC을 posix thread를 이용해 C코드로 구현을 해보았다. C를 사용함으로써 직접적으로 MAC이 어떻게 동작하는지 확인을 할 수 있었고 좀 더 범용 적으로 이용되는 C로 MAC을 구현해 유지 보수가 더 용이해진 이점을 얻었다.

SDL은 흐름도와 유사한 그래픽 다이어그램으로 프로토콜, 알고리즘을 알아보기 쉽게 제공한다. 하지만 시그널에 대한 정보만 있고 자세한 정보들, 예를들어 시그널 발생부분에서 마치 C코드의 함수처럼 몇 개의 인자(argument)가 들어가는지에 대해서만 나와 있고, 이것에 대한 자세한 설명은 찾아보기 힘들다. 인자 두 개를 이용해 octet string으로 만들어낸다는 정도의 추상적인 설명만이 되어있기 때문에 이러한 자세한 부분들은 먼저 이론적으로 그리고 표준문서의 앞부분에 기술되어있는 부분을 충분히 이해하고 있어야만 실질적으로 이용할 수 있는 언어(C 언어등)로 구현 할 수 있다. 거기에 어떤 시그널을 출력하는 부분에서 필요 인자로 2개의 인자가 들어가는데 그 시그널은 받는 부분에서는 3개의 인자를 필요로 하는 등의 몇몇 잘못된 부분들 또한 종종 발견 할 수 있다. 그래서 C로 구현에 있어서 이 부분을 어떻게 해결해야 하는지 많은 시간이 걸렸던 부분이기도 했다.

WLAN의 MAC을 이해하기 위해, 실제 FPGA보드를 이용해 하드웨어를 구현하기 위해 먼저 C 코드로 802.11의 MAC을 C로 구현해 보았는데 표준안에 사용된 SDL임에도 불구하고 그대로 구현 하면 동작하지 않는 부분이 몇몇 발견되어 이 부분에 대한 좀 더 자세한 연구가 필요하다. 또한 향후 과제로 현재 표준이 진행중인 IEEE 802.11n의 MAC에서 이용되는 QoS지원 부분과 security 강화 부분을 어떻게 구현할 것인가에 대한 연구도 필요하다.

6.참고문헌

[1] IEEE WG, "Part 11: Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) specifications." Revision of IEEE Std 802.11-1999, 2006

[2] Telelogic SDT 3.2, "SDT getting Started Part 1 : Tutorials on SDT Tools," Sep. 1997

[3] Threads Programming with Posix Threads, David R. Butenhof, Addison-Wesley, May. 1997