# Project Progress Report

**이선호, 이현민, 노순현**

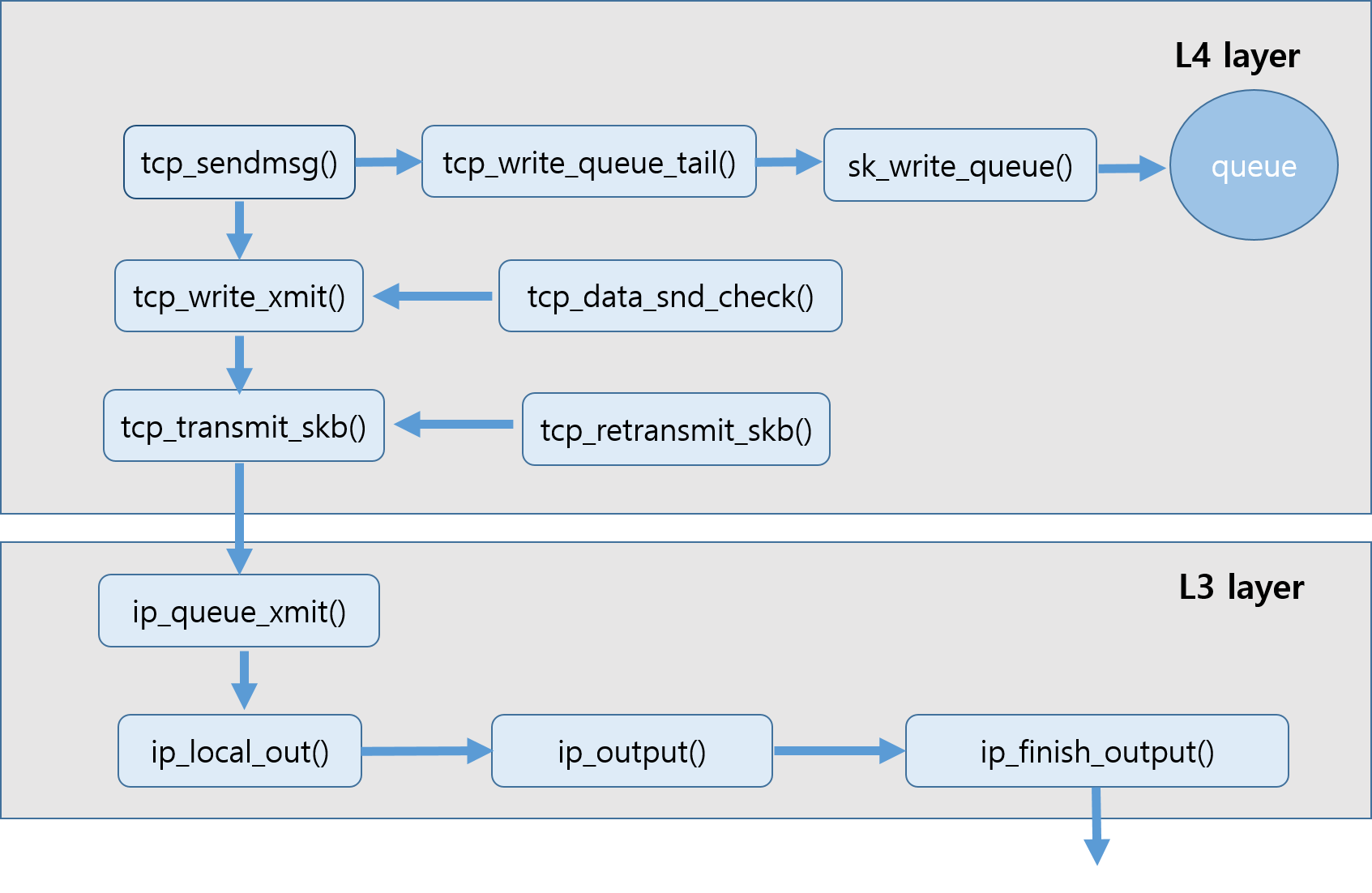
# 1. 개요

금주는 프로젝트를 진행하기 위한 기본 소스 코드 분석을 진행하였다. 분석한 소스 코드는 우리의 실함 대상인 OpenWrt의 원본 커널인 Linux kernel 3.10.49이다. 수업 시간 교수님께서 해 주셨던 조언에 따라, 앞으로의 구현 및 검증 과정에서도 AP 단의 동작 검증이 필수적인 상황이 아니면, Linux kernel이 설치된 노트북/데스크탑을 활용할 예정이다.

# 2. 분석 내용

* Fake TCP ACK generate 하기 위한 분석

우선 End device와 소켓 통신시 send하는 메시지에 대하여서 TCP layer에서는 tcp\_sendmsg() 함수가 패킷을 처리한다. 패킷 처리 flow를 간단히 나타내면 다음과 같다.



Socket 통신에서 write나 send를 통해 들어온 메시지에 대하여 tcp\_sendmsg() 함수가 이를 받아 처리하게 된다. tcp\_socket을 만들고 들어오는 메시지에 대하여 sk\_buffer라는 형식의 queue에 sk\_write\_queue()함수를 통해 저장한다.

앞서 보낸 데이터에 대한 ACK이 오면 tcp\_data\_snd\_check()를 통해 다음에 보내야 할 패킷의 유무를 확인한다. 만약 다음에 보내야 할 패킷이 존재한다면 tcp\_write\_xmit()함수를 호출한다. 이 함수에서는 앞서 queue에 넣었던 sk\_buffer 형식의 데이터를 하나 불러오고 이를 인자로 주어 tcp\_transmit\_xmit()함수를 호출한다. 혹은 tcp\_retransmit\_skb()를 통해 재전송 해야 할 패킷이 있는 것을 알았다면 이 패킷에 대하여 tcp\_transmit\_skb() 함수를 호출한다.

tcp\_transmit\_skb()함수에서 tcp header를 generate하여 패킷에 붙이는 작업이 이루어지고 다음 L3 layer로 패킷을 넘겨준다. tcp\_transmit\_skb에서 앞서 queue에 넣었던 데이터를 인자로 받게 된다. 그런데 이때 재전송 등의 요청에 따라 데이터가 다시 필요할 수도 있으므로 queue에 있는 데이터를 꺼내고 바로 지우는 것이 아니라 skb\_clone()함수를 통해 clone한 것을 사용한다.

tcp\_transmit\_skb에서 queue\_xmit() 함수를 호출하여 L3 layer로 패킷을 전달하여 주는데, queue\_xmit() 함수는 ip\_queue\_xmit() 함수에 매핑되어있다. 그리고 이 함수는 ip\_local\_out()을 호출하고 순서대로 ip\_output() 🡪 ip\_finish\_output() 함수가 실행되게 된다.

여기까지가 커널 코드에서 TCP/IP 패킷을 핸들링하는 플로우를 분석한 것이다.

패킷의 헤더를 어디서 generate하는지, 어디서 보내는지에 대한 정보를 알았으니 이를 바탕으로 Fake TCP ACK을 생성하고 보내도록 코드를 수정/추가하면 될 것으로 생각된다.

* Layer 2 (data link layer) 분석

우리가 layer 2의 네트워킹 동작을 분석하는 이유는 AP에 도착하는 ACK frame을 체크하여 TCP ACK을 보내는 시점을 결정하기 위해서이다. 이에 따라 layer 2에서 프레임의 TX/RX 동작을 전반적으로 분석하고, 우선 RX frame을 어떻게 처리해야 할지를 중점적으로 분석했다.

Linux kernel 내의 data link layer 소스코드는 크게 device dependent한 부분과 independent한 부분으로 나뉜다. 전자는 device driver를 관리하는 디렉토리에 위치하고, 후자는 802.11의 MAC을 관리하는 디렉토리가 존재한다. 두 디렉토리의 구체적인 위치는 아래와 같다.

* linux\_kernel\_dir/drivers/net/wireless/ath/ath9k
* linux\_kernel\_dir/net/mac80211

RX 프레임이 발생할 경우, device driver 단에서 들어온 프레임을 처리하기 위한 interrupt가 발생하고 ISR이 수행된다. ath9k의 경우 ath\_isr()이라는 함수가 수행되게 된다. 해당 ISR은 최종적으로 net\_rx\_action() 함수를 호출하게 되고, net\_rx\_action() 함수는 여러 함수를 호출하며 최종적으로 main IP receive routine인 ip\_rcv() 함수를 호출하게 된다.

아직 ACK frame을 device driver 단에서 어떻게 관리하는지에 대해 명확히 분석은 되지 않은 상태이나, 인터넷 상의 ath9k driver 관련 쓰레드에서 ath9k의 경우 하드웨어적으로 ACK 프레임을 처리한다는 문구를 확인하였다. (<https://www.mail-archive.com/ath9k-devel%40lists.ath9k.org/msg12280.html>) 이에 따라 ACK 프레임을 체크하는 알고리즘을 어느 시점에 어떻게 구현할지에 대한 추가적인 논의가 필요하다.

# 3. Role

* 이선호 – TCP layer 분석
* 이현민 – TCP layer ~ IP layer 연결 과정 분석
* 노순현 – Data link layer 분석 (802.11, ath9k driver)

# 4. 다음주 진행 계획

* Data link layer 추가 분석 (특히, ACK frame을 체크하는 부분을 어느 시점에 구현해야 할지에 대한 답을 얻는 것에 집중)
* Fake TCP ACK generate 구현 시작