Simple IP Implementation Report

2016160311 이재윤

Code Explanation

```
// If the next hop of found entry is NULL => Direct (next hop is destination IP)
if (rentry->next_hop == NULL) {
    next_hop = my_iph.dest;
}
// Else, use the next hop of found entry
else {
    next_hop = rentry->next_hop;
}
// !!! Fill the blank (Setup next hop variable)
```

라우팅 엔트리의 next hop은 rentry 구조체 내의 next_hop의 값에 해당하므로 이를 코드로 나타내면 rentry->next_hop에 해당한다. 이 값이 NULL이면 directly connected이므로 next hop이 destination IP가 되어야한다. 헤더 내부에 destination IP가 존재하고 이는 my_iph 구조체 내의 dest에 해당하므로 next_hop의 값을 my_iph.dest로 설정해주었다. 라우팅 엔트리의 next hop이 NULL이 아닐 경우 indirect forwarding이므로 엔트리에 명시된 next hop으로 설정해주면 된다. 따라서 next_hop의 값을 rentry->next_hop으로 설정해주었다.

```
// Retrieve ARP Cache to find the mac address for the next hop
if (strcmp(cache_ptr->dest, next_hop) == 0) {
    next_hop_mac = cache_ptr->dest_mac;
    break;
}
// !!! Fill the blank (Setup next_hop_mac variable)
```

ARP cache에 저장된 destination IP와 앞서 구한 next hop이 일치한다면 해당 destination의 MAC 주소를 구할 수 있다. ARP cache의 destination IP는 cache_ptr->dest에 해당하고 이 때의 MAC 주소는 cache_ptr->dest_mac에 해당하므로 위와 같이 구현해주었다. ARP cache 리스트에서 값을 찾으면 다른 리스트를 찾아볼 필요가 없으므로 break를 통해 while문을 빠져나온다.

```
// Adding My IP Header
// For the example, I leave some of the blank
buffer_ptr = buffer_data_ptr;

*((unsigned short *)buffer_ptr) = htons(my_iph.dest_len);
buffer_ptr += sizeof(unsigned short);
strncpy(buffer_ptr, my_iph.dest, my_iph.dest_len);
buffer_ptr += my_iph.dest_len;
*((unsigned short*)buffer_ptr) = htons(my_iph.source_len);
buffer_ptr += sizeof(unsigned short);
strncpy(buffer_ptr, my_iph.source, my_iph.source_len);
buffer_ptr += my_iph.source_len;
*((unsigned char*)buffer_ptr) = htons(my_iph.ttl);
buffer_ptr += sizeof(unsigned char);

// !!! Fill the blank (Set dest, source len, source, ttl)
```

버퍼에는 IP 헤더와 이더넷 헤더 정보가 담기게 된다. 위의 코드는 IP 헤더를 버퍼에 추가하는 코드이다. 버퍼에 순서대로 IP 헤더의 값인 destination IP length, destination IP, source IP length, source IP, TTL 값을 넣어준다. htons 함수를 통해 endian translation을 해주었다.

```
// TTL Check - If 0, Drop and stop further processing
my_iph.ttl--;
if (my_iph.ttl == 0) {
    printf("TTL is 0₩n");
    return -1;
}
// !!! Fill the blank
```

들어온 IP 헤더 내의 TTL 값을 하나 내린 뒤 TTL의 값이 0이라면 그 패킷을 버리고 진행을 멈춘다. IP 헤더의 TTL 값은 my_iph.ttl에 해당하므로 위의 코드와 같이 구현해주었다. 그리고 진행을 멈추기 위해 정수 -1을 반환함으로써 my_ip_forward 함수를 빠져나왔다.

```
// Get ethertype and dispatch payload part to proper process function
ethtype = eh.eth_type;
if (ethtype == htons(0xfffe)) {
    my_ip_receive(buffer + sizeof(eh));
}
else {
    printf("Ethertype Error₩n");
}
// Endian translation is needed
// !!! Fill the blank
```

전달받은 패킷의 이더넷 헤더를 열고 이더넷 타입이 0xfffe가 맞다면 정상적으로 진행한다. 앞서 IP 헤더와 이더넷 헤더를 추가할 때 endian translation을 해줬으므로 이더넷 헤더를 확인할 때에도 htons 함수를 이용하여 endian translation을 해주었다. 이더넷 헤더 확인은 끝났으므로 다음 진행을 위해 버퍼의 스택 포인터 값을 이더넷 헤더의 사이즈만큼 옮겨준 후 my_ip_receive 함수로 넘겨주었다.

Answers to Question 3 and 4

3. Without default gateway entry, how can you reduce the number of VM1's routing table entry?

→ VM1의 라우팅 테이블 엔트리에서 destination의 Sungbukgu Anamdong 1/20과 Sungbukgu Anamdong 2/20은 next hop과 output interface가 동일하므로 굳이 나눠서 나타낼 필요 없이 Sungbukgu Anamdong/18 하나로 나타내어도 matching이 가능하므로 문제 없이 패킷을 포워딩 할 수 있을 것이다. 마찬가지로 Sungbukgu Bomundong 1/21과 Sungbukgu Bomundong 2/21 역시 next hop과 output interface가 동일하므로 Sungbukgu Bomundong/19로 생략 가능하다. Sungbukgu Anamdong/18과 Sungbukgu Bomundong/19도 next hop과 output interface가 동일하므로 더 생략하고자 한다면 Sungbukgu/9 하나로 전부 생략할 수 있다.

4. How can you configure the default routing entry of VM1?

→ Network topology에 따르면 VM1 라우터는 interface a를 통해 VM2 라우터와 연결되어 있고 다른 인터페이스는 존재하지 않는다. 이미 설정한 라우팅 테이블은 Dongdaemungu와 Sungbukgu에 대한 IP는 전송을 보장하지만 그 외의 IP에 대해서는 default gateway가 존재하지 않아 포워딩 해줄 수 없다. Output interface는 a 하나뿐이므로 matching되지 않는 IP 역시 interface a를 통해 Dongdaemungu 2로 보내주도록 설정하면 된다. 실제 구현에서는 interface a가 enp0s3에 해당하므로 add_routing_entry("", "Dongdaemungu 2", "enp0s3")를 통해 default routing entry를 구현할 수 있을 것이다.