4.3.3 pcap API를 이용한 패킷 스니핑

Open *sniffc.c ~/Downloads/ch04_lab/volumes Save

```c
/*
이 프로그램은 기본적인 패킷 캡처 기능을 수행하며,
"icmp" 필터를 사용하여 ICMP 패킷만 캡처.
캡처된 패킷이 있을 때마다 단순히 "Got a packet" 메시지를 출력.
실제 응용 프로그램에서는 got_packet 함수에서 더 복잡한 패킷 처리 로직을 구현
*/

/*
표준 라이브러리(stdlib.h, stdio.h)와
패킷 캡처 라이브러리(pcap.h)를 포함
*/
#include <stdlib.h>
#include <stdio.h>
#include <pcap.h>

/*
패킷이 캡처될 때 호출되는 콜백 함수.
패킷이 캡처되면 "Got a packet"이라는 메시지를 출력.
args, header, packet은 각각 추가 인자, 패킷 헤더, 패킷 데이터를 의미
*/
void got_packet(u_char *args, const struct pcap_pkthdr *header,
                              const u_char *packet)
{
  printf("Got a packet\n");
}

```

sniff.c
실습 코드 설명

pcap_compile 함수

■ 함수 원형

- **int pcap_compile(pcap_t *p, struct bpf_program *fp, const char *str, int optimize, bpf_u_int32 netmask);**
- ***pcap_t p**
  - pcap_open_live 또는 pcap_open_offline 함수에 의해 반환된 pcap 세션 핸들.
  - 이 핸들은 패킷 캡처 세션을 나타내며, 필터가 적용될 인터페이스를 지정.
- ***struct bpf_program fp**
  - 컴파일된 BPF 프로그램을 저장할 구조체 포인터.
  - 이 구조체는 나중에 pcap_setfilter 함수에 의해 실제 필터로 사용.
  - 함수가 성공적으로 실행되면, fp는 컴파일된 필터 코드를 포함.
- ***const char str**
  - 필터 표현식을 나타내는 문자열.
  - 예제에서는 "icmp"이 사용되었으며, 이는 ICMP 패킷만 캡처하겠다는 의미.
  - 필터 표현식은 tcpdump의 필터 표현식 문법 사용.
- **int optimize**
  - 최적화 옵션을 나타내는 정수.
  - 0이면 최적화하지 않음을 의미하고, 1이면 최적화를 수행.
  - 일반적으로 1로 설정하여 필터를 최적화.
- **bpf_u_int32 netmask**
  - 네트워크 마스크를 나타내는 32비트 정수.
  - 네트워크 주소와 비교하여 필터 표현식을 해석하는 데 사용.
  - 로컬 네트워크의 서브넷 마스크를 지정.

스니핑 프로그램 단계

- **step 1: 라이브 pcap 세션을 연다(라인 ①).** 이 단계는 원시 소켓을 초기화하고 enp0s3 네트워크 장치를 무차별 모드로 설정하고(세 번째 매개변수의 값 1은 무차별 모드를 켠다) setsockopt()를 사용하여 소켓을 카드에 바인딩한다.

- **step 2: 필터를 설정한다(라인 ②와 ③).** pcap API는 부울 술어 표현식을 저수준 BPF 프로그램으로 변환하는 컴파일러를 제공한다.

- **step 3: 패킷 캡처(라인 ④).** pcap_loop()를 사용하여 패킷이 캡처되는 pcap 세션의 기본 실행 루프에 들어간다.

- **컴파일하기.** gcc를 사용하여 pcap을 사용하는 코드를 컴파일할 때 -lpcap 인수를 추가해야 한다

4.3.4 캡처된 패킷 처리하기 : Ethernet Header

```c
#include <stdio.h>
// 패킷 캡처 라이브러리인 pcap를 포함
#include <pcap.h>
// 인터넷 작업을 위한 라이브러리로, IP 주소 변환 함수 등이 포함
#include <arpa/inet.h>

/* ethernet headers are always exactly 14 bytes [1] */
// 이더넷 헤더의 크기를 정의
#define SIZE_ETHERNET 14
/* Ethernet addresses are 6 bytes */
//이더넷 주소의 길이를 정의
#define ETHER_ADDR_LEN  6
//최대 패킷 길이를 정의
#define PACKET_LEN   1500

/* Ethernet header */
/*
이더넷 헤더는 목적지 호스트 주소,
출발지 호스트 주소, 이더넷 타입을 포함
*/
struct ethheader {
  u_char  ether_dhost[ETHER_ADDR_LEN];    /* destination host address */
  u_char  ether_shost[ETHER_ADDR_LEN];    /* source host address */
  u_short ether_type;                     /* IP? ARP? RARP? etc */
};


/* IP Header */
```

sniff_improved.c 설명

캡처된 패킷 추가 처리하기

- TCP, UDP 및 ICMP 헤더를 출력하는 것과 같이 패킷을 추가로 처리하려는 경우 유사한 기술을 사용할 수 있다.
  - ➲ **포인터를 다음 헤더의 시작 부분으로 이동하고 유형 변환한다.**
  - ➲ **IP 헤더의 실제 크기를 계산하려면 IP 헤더의 헤더 길이 필드를 사용해야 한다.**

- 다음 예에서 다음 헤더가 ICMP라는 것을 알고 있으면 다음을 수행하여 ICMP 부분에 대한 포인터를 얻을 수 있다:

```
int ip_header_len = ip->iph_ihl * 4;
u_char *icmp = (struct icmpheader *)
               (packet + sizeof(struct ethheader) + ip_header_len);
```