10. Raw 소켓

- Raw 소켓을 사용하는 경우
- ICMP 프로토콜을 이용하는 ping 프로그램
  - ICMP는 호스트 서버와 인터넷 게이트웨이 사이에서 메시지를 제어하고 에러를 알려주는 프로토콜

Raw 소켓의 필요성

- 일반 소켓을 통해서 데이터를 전송
  - TCP, UDP, IP 등의 통신 프로토콜을 처리하기 위한 헤더가 커널에 의해 자동으로 만들어진다.
  - 프로그램 작성이 편리
- 각종 프로토콜 헤더를 직접 작성해야 하는 경우
  - ICMP ECHO 프로토콜을 이용하는 ping 프로그램
  - Raw 소켓을 이용하여 송수신
- Raw 소켓을 사용할 경우
  - 임의의 프로토콜 헤더를 만들어 직접 전송, IP 헤더는 커널에 의해 만들어진 것을 사용 가능
  - 해킹이나 네트워크 보안 프로그램에서는 IP 헤더도 프로그래머가 작성
    - Raw 소켓을 생성한 후 setsockopt()를 사용

Raw 소켓 생성

- **생성 함수**

```
int s;
s = socket(AF_INET, SOCK_RAW, protocol);
```

  - AF_INET : **주소체계**
  - SOCK_RAW : Raw **소켓을 생성**
  - protocol : IP **헤더의** protocol **필드에 입력되는 값**

```
// protocol 인자에 올 수 있는 값
IPPROTO_ICMP =1
IPPROTO_IGMP = 2
IPPROTO_TCP =6
IPPROTO_UDP = 17
IPPROTO_IPV6 = 41
IPPROTO_RAW = 255
```

- Raw **소켓을 통하여 데이터그램의 전송**
  - **송신** : sendto(), sendmsg()
  - **수신** : recvfrom(), recvmsg()

10.2. Ping 프로그램 구현

- Raw 소켓을 사용하여 ping과 유사한 프로그램 구현
- ping은 자신의 호스트가 다른 컴퓨터와 잘 연결되어 있는지를 확인하는 명령

10.2.1 프로그램 개요(myping.c)

- 다음의 순서로 동작
  - Raw 소켓을 개설, ping 요청을 보낼 목적지 호스트의 IP 주소를 설정
  - ICMP 헤더를 작성
  - sendto()로 ICMP ECHO 요청을 전송
  - 타임 아웃을 설정
  - recvfrom()로 ICMP ECHO 응답을 기다리다 도착하면 RTT(round trip time)을 측정하여 출력
    - RTT = 수신시간 – 송신시간
    - 응답이 도착하기 전에 타임 아웃되면 “Request time out”을 출력

소켓 개설과 목적지 주소 설정

- Raw **소켓 개설**

```
int sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
```

- **목적지 호스트 주소 설정**

```
struct  sockaddr_in peer;
sendaddr.sin_family = AF_INET;
inet_pton(AF_INET, argv[1], &sendaddr.sin_addr.s_addr);
sendaddr.sin_port = htons(0); // IP 계층 프로토콜은 포트번호를 지정하지 않음
```

사용자 IP 헤더 작성

- ICMP 프로토콜을 사용하므로 Raw 소켓을 개설한 후 ICMP 헤더를 직접 구성
- IP 헤더는 커널 혹은 사용자가 만들 수 있다.
- 사용자가 만들 경우 Raw 소켓에 대해 IP_HDRINCL 옵션을 지정

```
int on = 1;
sockfd = socket(AF_INET, SOCK_RAW, IPROTO_RAW);
If (setsockopt(sockfd, IPRPTO_IP, IP_HDRINCL, &on, sizeof(on) < 0){
    printf("error");
    exit(1);
}
```

- checksum 값을 계산
  - IP 헤더의 첵섬 필드를 제외한 모든 필드를 유효한 값으로 채운 후 계산
  - 만약 헤더의 필드를 수정하면 다시 계산

ICMP 에코 요청 전송

- ICMP 헤더 구조

Type (1)	Code (1)	Checksum (2)	Ident (2)	Seq Num (2)	Data (N byte)

  - Type : ICMP_ECHO(=8), Echo Reply(=0)
  - Code : 0
  - Checksum : **사용자가 계산**
  - Ident : ICMP **데이터의** ID, **보통** pid**를 사용**
  - Seq Num : **임의로 설정**
  - Data : **전송할 데이터**
- ECHO **요청 및 응답을 위한** ICMP **헤더는** icmphdr **구조체에 저장**

```
Struct icmphdr {
        u_int8_t type;                  //message type
        u_int8_t code;                  //type sub-code
        u_int16_t checksum;
        union {
                struct{ u_int16_t id;
                        u_int16_t sequence;
                } echo;                 // echo datagram
                u_int32_t gateway;              // gateway address
                struct { u_int16_t  __unused, mtu;} flag // path mtu discovery
        } un;
};
```

ICMP 에코 요청 전송(계속)

- ICMP ECHO 요청 헤더 구성 예

```
char sendbuf[1024];
icmp = (struct icmphdr *) sendbuf;
bzero((char *) icmp, sizeof(struct icmp));
icmp->code = 0;
icmp->type  = ICMP_ECHO; // ICMP_ECHO = 8
icmp->un.echo.sequence = seqnum++;// Ping 메시지 일련번호
icmp->un.echo.id = getpid();    // pid 를 ID로 설정
icmp->checksum = 0;        // checksum 계산 전 반드시 초기화
// 체섬을 계산하는 함수 : in_cksum
icmp->checksum=in_cksum((unsigned short *)icmp, sizeof(struct icmp));
len = sizeof(struct icmphdr);  // 8 byte
sendsize = sendto(rawsock, sendbuf, len, 0,
      (struct sockaddr*)&sendaddr, sizeof(struct sockaddr));
```

- Ping 데이터는 sendto() 함수로 전송

```
// “hello”를 전송할 경우 헤더가 8 바이트이므로 총 13 바이트를 전송
strncpy(&sendbuf[8], “hello”);
int len = 8 + strlen(“hello”);
sendsize = sendto(rawsock, sendbuf, len, 0,
                        (struct sockaddr *)&addr, sizeof(struct sockaddr));
```

ICMP ECHO 응답 수신

- ICMP **요청 패킷을 전송 후** 1**초간 타이머를 구동** : select()**의** timeout **인자 사용**
- **타임아웃될 때까지 응답이 오지 않으면** select()**는 리턴하고** ping**을 다시 보낸다**.
- 3**번 연속** ping**을 보내는 동안 응답을 수신못하면** "Request Timeout" **메시지 출력**

```
while(1) {
      FD_ZERO(&readset);  FD_SET(rawsock, &readset);
      tv.tv_sec =  1; // 1초 타이머
      tv.tv_usec = 0;
      send_ping();    // ping을 보냄
      ret = select(rawsock+1, &readset, NULL,NULL,&tv); // 타이머 설정
      if(ret == 0) {    // 타임아웃
         if(++notrecv == 3) {
            notrecv = 0; puts("Request Timeout ... ");
         }
         continue;
      }
      else if(ret <0) // select()의 에러
         errquit("select fail");
      // select()의 정상리턴, ping 응답을 읽음
      recvsize = recvfrom(rawsock, recvbuf, sizeof(recvbuf),0,
                                          (struct sockaddr*)&recvaddr, &addrlen);
      if(recvsize<0)
         errquit("recvfrom fail ");
      notrecv = 0; // 수신된 응답에 대한 처리
      prn_rcvping(recvbuf, recvsize);// ping 메시지 출력
      sleep(1);// 1초 간격으로 ping을 보냄
}
```

실행결과

- Root **권한으로 실행**

```
root> $ myping 210.115.36.128
[송신] 8 byte[icmp] (id:10363 seq:0 code:0 type:8)
          [수신] 8 byte (210.115.36.128) [icmp](id:10363 seq:0 code:0 type:8)

[송신] 8 byte[icmp] (id:10363 seq:1 code:0 type:8)
          [수신] 8 byte (210.115.36.128) [icmp](id:10363 seq:1 code:0 type:8)
.........
```

- **타임아웃 이내에** ping**에 대한 응답을 수신 못한 경우**

```
root> $ myping 210.115.36.128
[송신] 8 byte[icmp] (id:10363 seq:0 code:0 type:8)
[송신] 8 byte[icmp] (id:10363 seq:0 code:0 type:8)
[송신] 8 byte[icmp] (id:10363 seq:0 code:0 type:8)
Request Timeout . . .
.........
```