

Global KPU!

글로벌 경쟁력을 갖춘 산업기술 명문대학
세계를 향해 더 큰 미래를 펼쳐갑니다

12 Raw 소켓

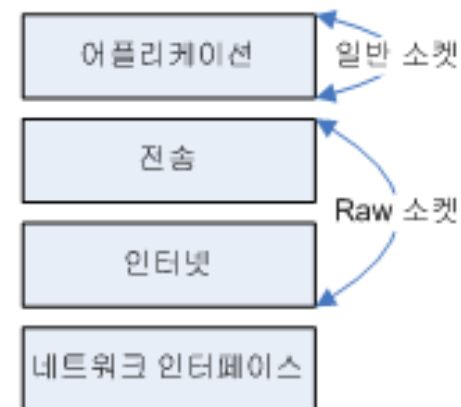
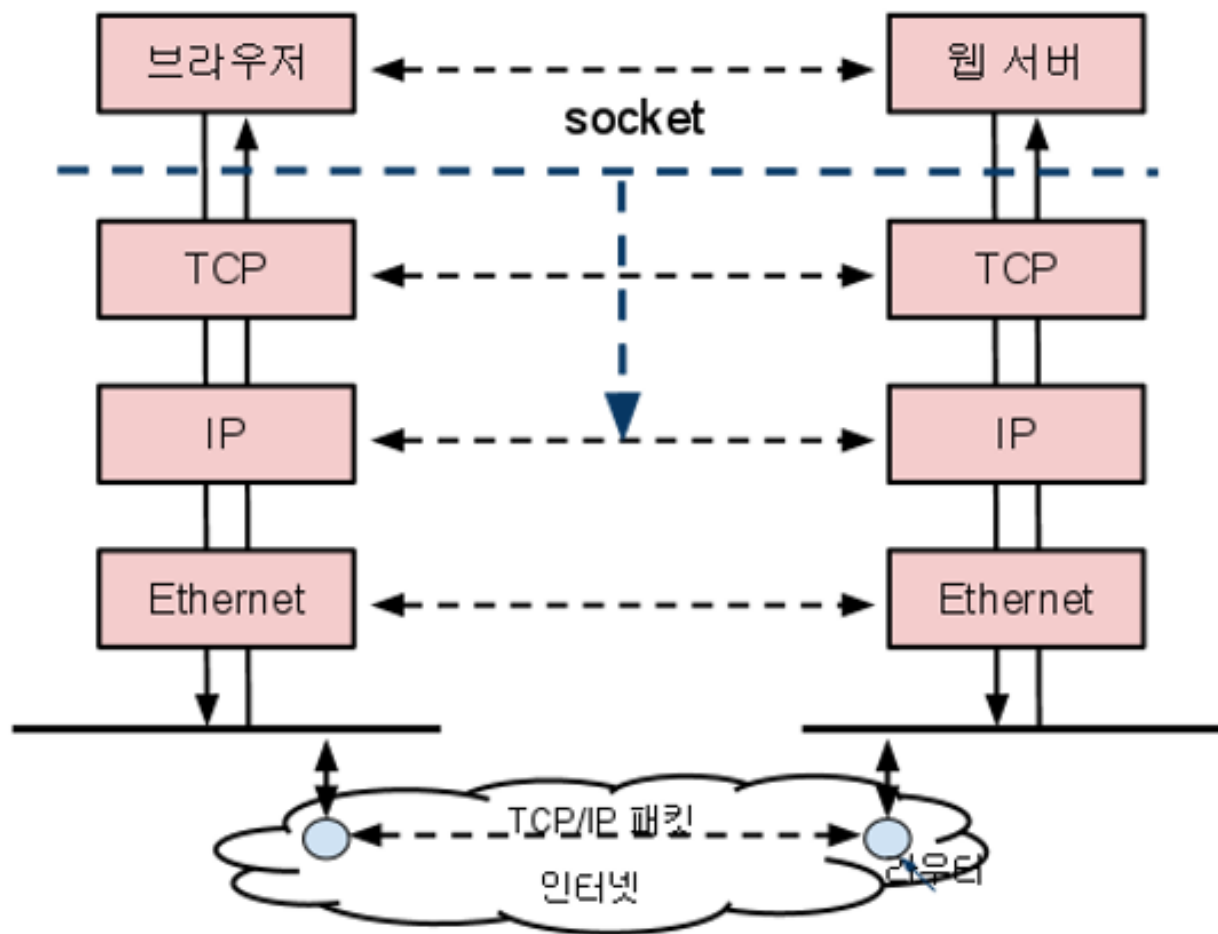
네트워크 게임 프로그래밍

- ❖ Raw 소켓의 특징과 사용 방법을 익힌다.
- ❖ Ping 프로그램의 동작 원리를 이해하고 작성 방법을 익힌다.
- ❖ Traceroute 프로그램의 동작 원리를 이해하고 작성 방법을 익힌다.
- ❖ ICMP.DLL이 제공하는 함수를 활용하는 방법을 익힌다.



❖ RAW 소켓

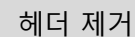
- TCP/IP 4계층의 Transport Layer 이하를 아래와 같이 추상화



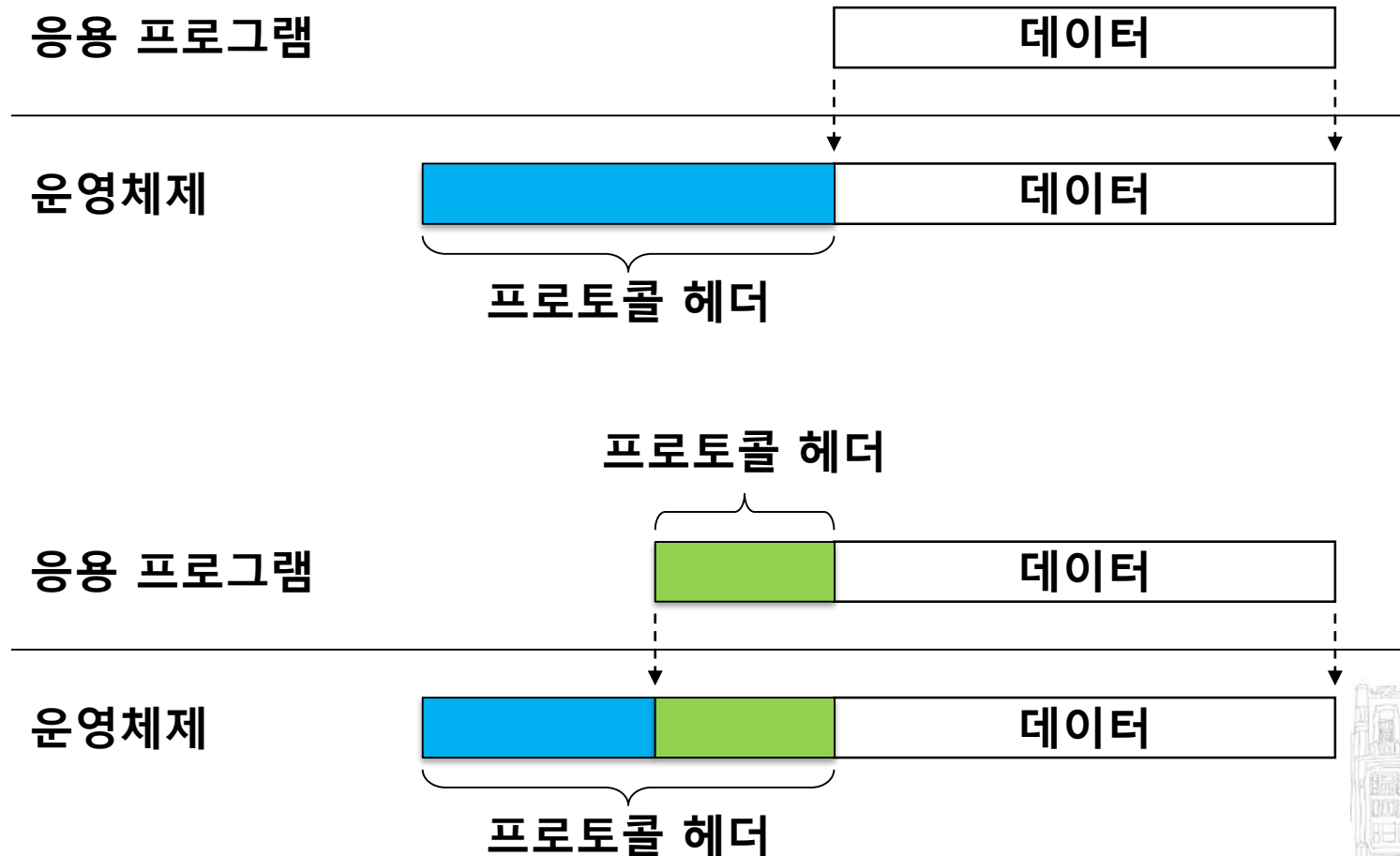
TCP/IP 4 계층



헤더 붙임



❖ TCP(또는 UDP) 소켓 vs. Raw 소켓



❖ Raw 소켓의 특징

- 응용 프로그램 수준에서 프로토콜 헤더를 직접 조작
 - ⇒ 기존의 TCP 또는 UDP 소켓을 사용하는 방식보다 세부적인 제어 가능
- 프로토콜 헤더의 구조와 동작 원리를 이해해야 하므로 프로그래밍이 상대적으로 어려움
- 해킹에 악용할 가능성이 있음



❖ Raw 소켓 생성 예

```
SOCKET sock = socket(AF_INET, SOCK_RAW, protocol);  
if(sock == INVALID_SOCKET) err_quit("socket()");
```

❖ Raw 소켓의 종류

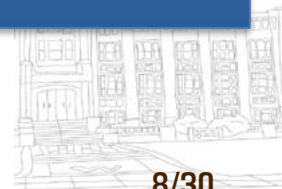
프로토콜 번호	응용 프로그램이 생성할 헤더	운영체제가 생성할 헤더
IPPROTO_ICMP	ICMPv4	IPv4
IPPROTO_IGMP	IGMPv4	
IPPROTO_ICMPV6	ICMPv6	IPv6



❖ IP_HDRINCL 또는 IPV6_HDRINCL 옵션 설정

```
#include <ws2tcpip.h>
...
// IPv4 헤더 포함 옵션 설정
BOOL optval = TRUE;
setsockopt(sock, IPPROTO_IP, IP_HDRINCL, (char *)&optval, sizeof(optval));
```

```
#include <ws2tcpip.h>
...
// IPv6 헤더 포함 옵션 설정
BOOL optval = TRUE;
setsockopt(sock, IPPROTO_IPV6, IPV6_HDRINCL, (char *)&optval, sizeof(optval));
```

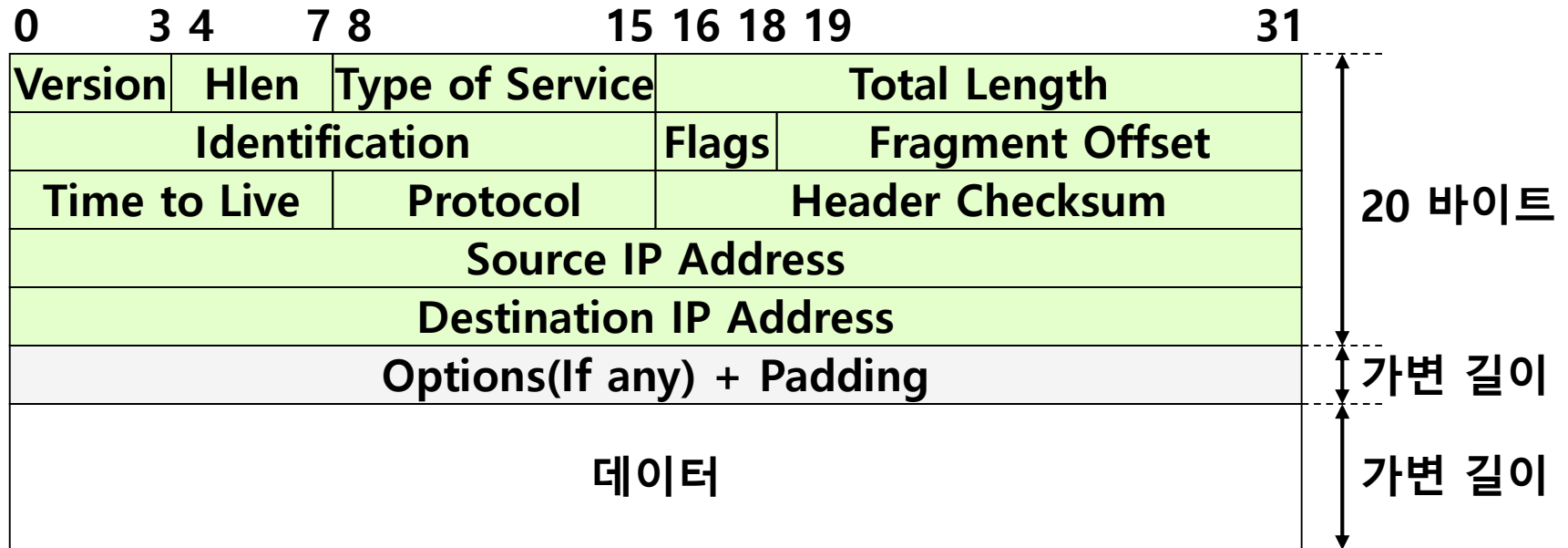


❖ Raw 소켓 생성 시 주의 사항

- 운영체제마다 Raw 소켓을 생성할 수 있는 권한이 다름
 - 윈도우 9x 계열(95/98/Me)
 - 모든 사용자
 - 윈도우 NT 계열(NT 4.0/2000/XP 이상)
 - 관리자 그룹에 속한 사용자
- 운영체제마다 생성할 수 있는 Raw 소켓의 종류가 다름
 - 윈도우 95/98/NT 4.0
 - IPPROTO_ICMP, IPPROTO_IGMP
 - 윈도우 Me/2000
 - 모든 종류의 IPv4 Raw 소켓
 - 윈도우 XP SP1 이상
 - 모든 종류의 IPv4와 IPv6 Raw 소켓



❖ IPv4 패킷 구조



- **version** : IP 버전을 나타내는 필드. IPv4패킷일 경우 4가 들어가고 IPv6일 경우 6이 들어 있음
- **header length** : IP헤더 길이를 4바이트 단위로 나타냄. 헤더 길이는 20~60바이트 까지로 가변적임. 이러한 헤더의 길이는 워드 단위로 나타내며 header length의 값이 5라면 $5 \times 4\text{바이트} = 20\text{바이트}$ 가 됨
- **Type of service** : IP패킷별로 요구되는 서비스 특성(우선순위, 지연시간등)을 나타냄. QOS등을 정의하는 필드이지만 IPv4에서는 사용되지 않음
- **Total length** : 헤더와 데이터 길이를 포함한 전체 길이를 바이트 단위로 나타냄.

❖ IPv4 패킷 구조 (이어서)

- **Identification** : 운영체제가 IP 패킷에 부여하는 고유한 번호. 패킷의 유일한 식별자로서 각각의 IP패킷을 유일하게 구분해 줌
- **Flags / Fragmentation offset** : IP 패킷 분할(fragmentation)과 재조립(reassembly)에 사용
- **Time to live** : Time to live는 줄여서 TTL이라 부르는데 TTL 값은 패킷이 네트워크 내에서 영원히 떠돌아 다니지 않도록 하기위해 패킷의 생존 기간을 설정하기 위한 필드. 처음 설정된 TTL값이 라우터 또는 호스트를 하나씩 지나갈때마다 1씩 감소하여 0이되면 해당 패킷을 받은 라우터는 패킷을 버리고 발신지로 ICMP패킷을 보내줌
- **Protocol** : IP패킷이 담고 있는 데이터가 어떤 프로토콜인지 나타내는 필드.
- **Header checksum** : IP패킷의 헤더가 정상적인지 검사하는데 사용되는 체크섬 값이 설정된 필드이다. 패킷을 받는 목적지 호스트는 체크섬을 확인하여 결과가 다르다면 패킷을 버린다.
- **Source IP Address** : 출발지 IP주소 설정
- **Destination IP Address** : 목적지 IP주소 설정



Raw 소켓 입출력 (1)

❖ 참고

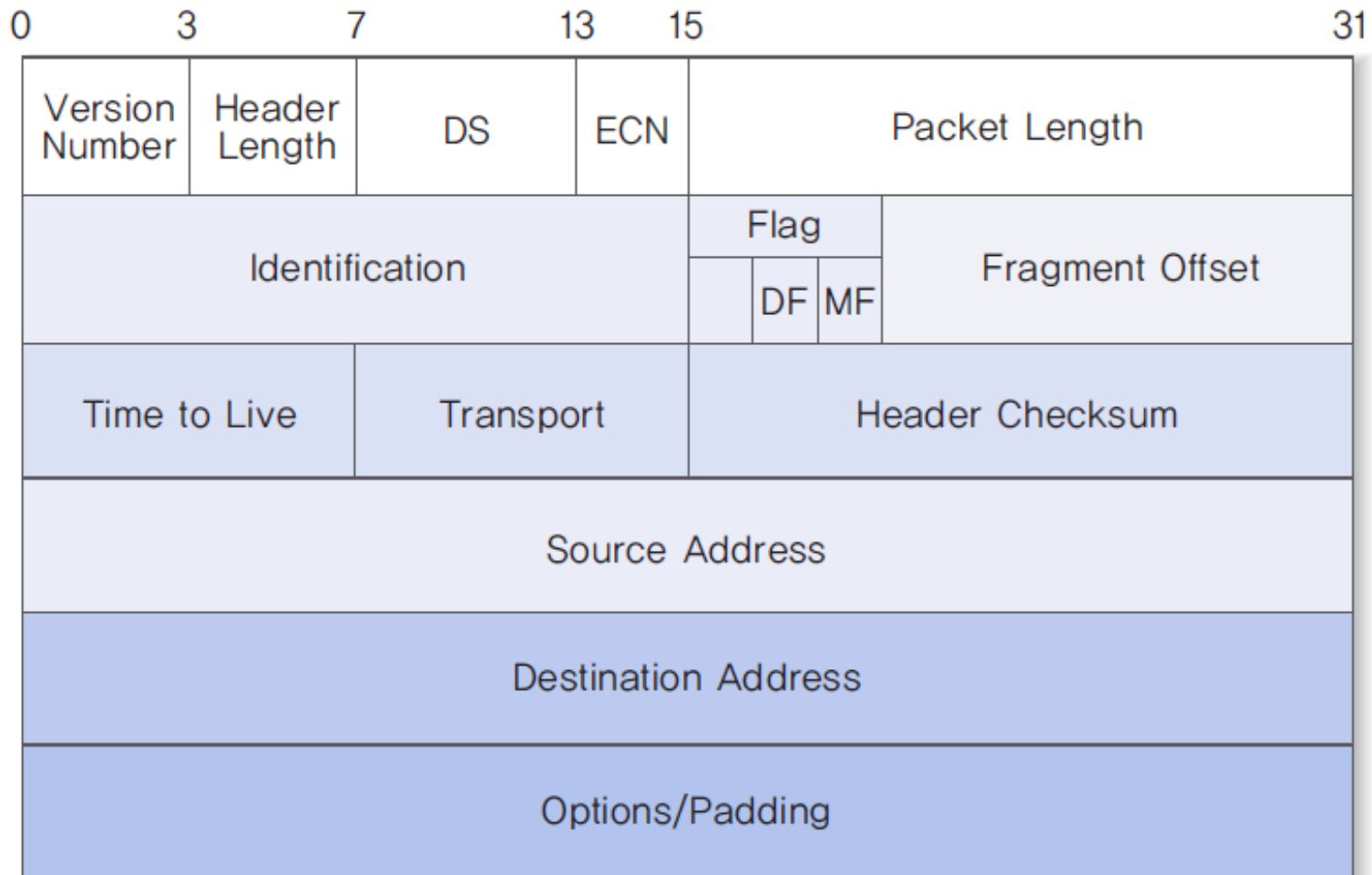


그림 7-12 IP 헤더의 구조



❖ IPv4 패킷 정의

```
typedef struct _IPHEADER
{
    u_char    ip_hl;         // header length
    u_char    ip_v;         // version
    u_char    ip_tos;       // type of service
    short     ip_len;       // total length
    u_short   ip_id;        // identification
    short     ip_off;       // flags & fragment offset field
    u_char    ip_ttl;      // time to live
    u_char    ip_p;        // protocol
    u_short   ip_cksum;     // checksum
    IN_ADDR   ip_src;      // source address
    IN_ADDR   ip_dst;      // destination address
} IPHEADER;
```



❖ Raw 소켓 출력(IPv4 기준)

- 일반적으로 **sendto()** 함수를 사용한다. 이때 목적지 주소로 브로드캐스트 주소나 멀티캐스트 주소를 사용할 수도 있다.
- IP_HDRINCL 옵션을 설정하지 않은 경우, **socket()** 함수의 세 번째 인자에 해당하는 프로토콜 헤더를 생성하고 여기에 응용 프로그램 데이터를 덧붙여 보낸다. 이때 IPv4 헤더는 운영체제가 자동으로 생성해 덧붙인다.
- IP_HDRINCL 옵션을 설정한 경우, **IPv4 헤더와 socket() 함수의 세 번째 인자에** 해당하는 프로토콜 헤더를 생성하고 여기에 응용 프로그램 데이터를 덧붙여 보낸다. 즉, 응용 프로그램이 IPv4 헤더를 포함한 패킷 전체를 생성해 보내는 것이다.
- IP_HDRINCL 옵션을 설정했더라도 IPv4 헤더의 **Identification**을 0으로 채우면 운영체제가 자동으로 Identification 번호를 설정해준다.



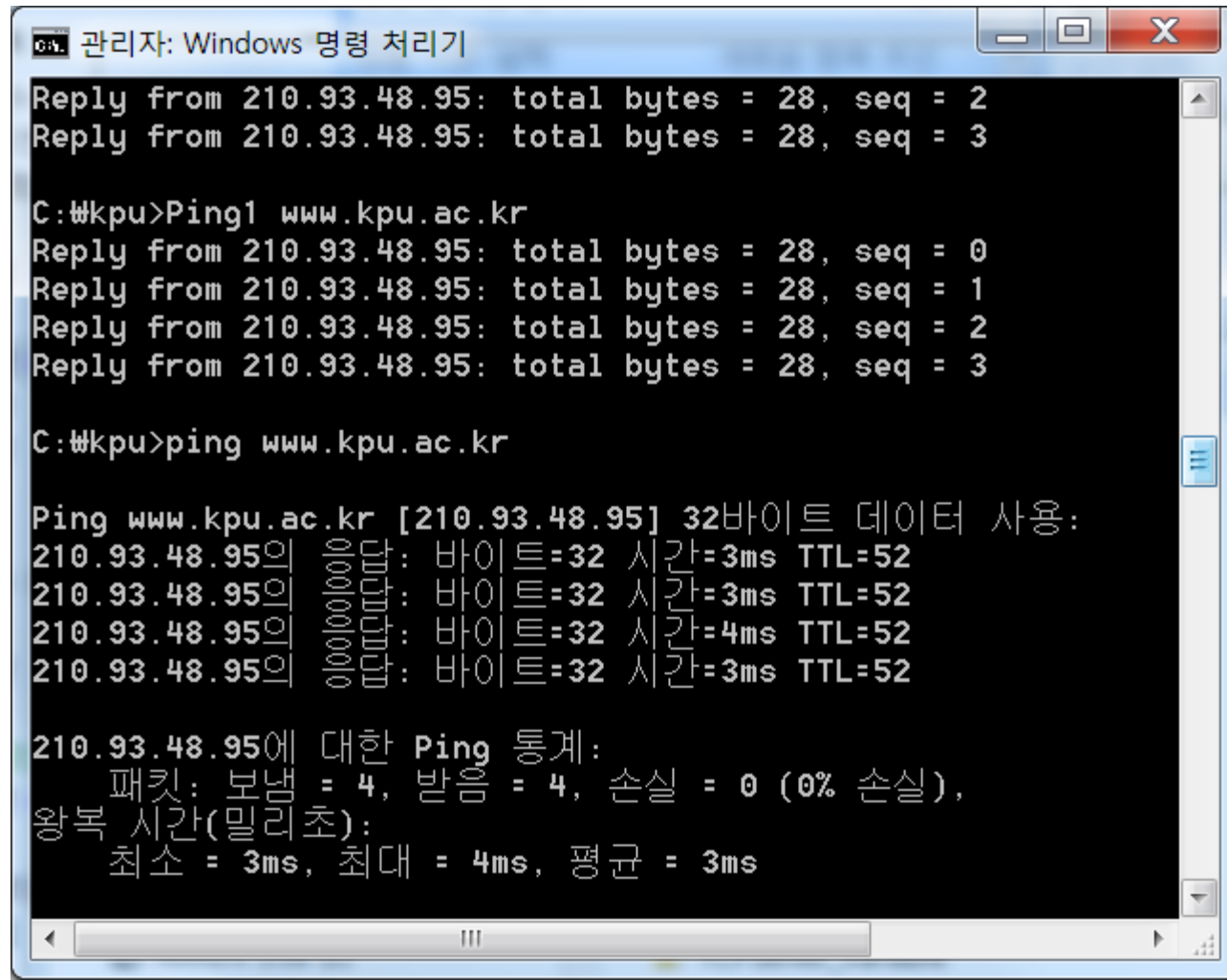
❖ Raw 소켓 입력

- 일반적으로 **recvfrom()** 함수를 사용한다.
- IPv4의 경우에는 IP_HDRINCL 옵션 사용 여부와 관계없이 데이터 맨 앞쪽에 항상 IPv4 헤더가 포함되어 읽힌다. IPv6의 경우에는 IPV6_HDRINCL 옵션 사용 여부와 관계없이 항상 IPv6 헤더 다음 부분 부터만 읽을 수 있다.
- 운영체제는 IP 패킷을 받으면 프로토콜이 일치하는 모든 Raw 소켓에 이 패킷을 전달한다. 여기서 프로토콜이 일치한다는 것은 IP 패킷의 Protocol 부분과 Raw 소켓 생성 시 socket() 함수에 전달한 세 번째 인자가 일치한다는 뜻이다.



❖ Ping 응용 프로그램

- 호스트나 라우터의 작동 여부를 확인할 때 사용
- ICMP 프로토콜을 이용하여 구현



```
관리자: Windows 명령 처리기

Reply from 210.93.48.95: total bytes = 28, seq = 2
Reply from 210.93.48.95: total bytes = 28, seq = 3

C:\wkpu>Ping1 www.kpu.ac.kr
Reply from 210.93.48.95: total bytes = 28, seq = 0
Reply from 210.93.48.95: total bytes = 28, seq = 1
Reply from 210.93.48.95: total bytes = 28, seq = 2
Reply from 210.93.48.95: total bytes = 28, seq = 3

C:\wkpu>ping www.kpu.ac.kr

Ping www.kpu.ac.kr [210.93.48.95] 32바이트 데이터 사용:
210.93.48.95의 응답: 바이트=32 시간=3ms TTL=52
210.93.48.95의 응답: 바이트=32 시간=3ms TTL=52
210.93.48.95의 응답: 바이트=32 시간=4ms TTL=52
210.93.48.95의 응답: 바이트=32 시간=3ms TTL=52

210.93.48.95에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 3ms, 최대 = 4ms, 평균 = 3ms
```



❖ ICMP

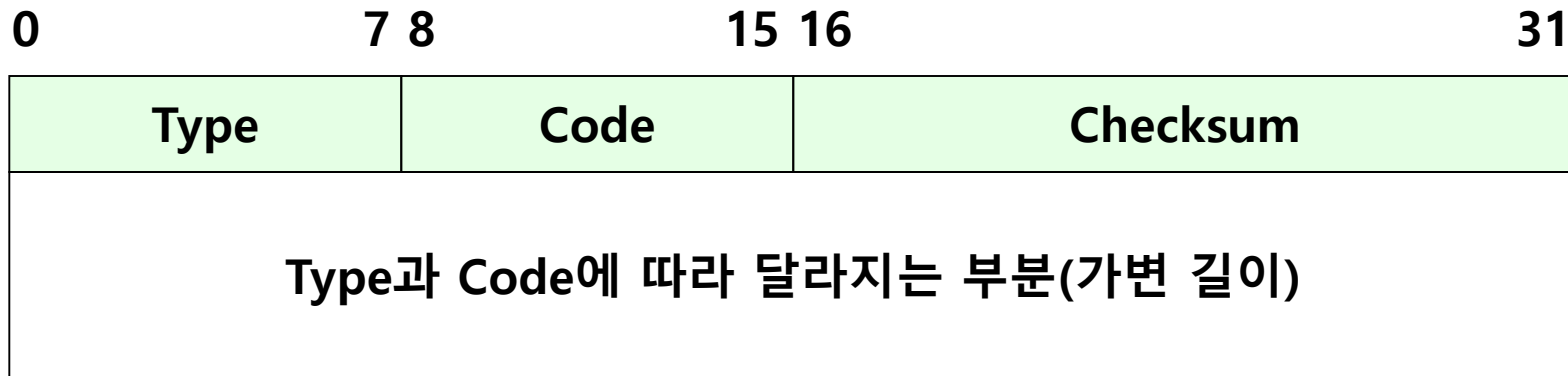
- 인터넷에 연결된 호스트나 라우터 간에 유용한 정보(오류 발생, 라우팅 정보 등)를 알리는 목적으로 사용
- 항상 IP 패킷에 포함된 형태로 전송되며 TCP/IP 프로토콜 동작에 필수 역할을 함

IP 헤더

ICMP 메시지



❖ ICMP 메시지 구조



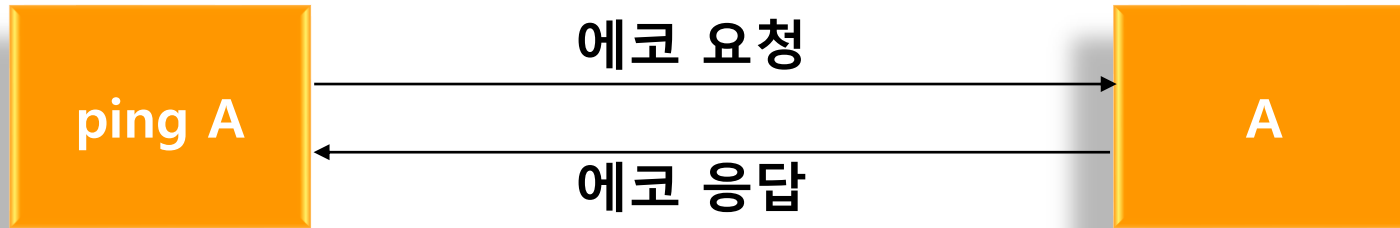
❖ ICMP 메시지 정의

```
typedef struct _ICMPMESSAGE
{
    u_char icmp_type;      // type of message
    u_char icmp_code;      // type sub code
    u_short icmp_cksum;    // checksum
    ...
} ICMPMESSAGE;
```



Ping (4)

❖ Ping 응용 프로그램 동작 원리



❖ 에코 요청, 에코 응답 ICMP 메시지

0	7 8	15 16
Type(8 또는 0)	Code(0)	Checksum
Identifier		Sequence Number
옵션 데이터(가변 길이)		



❖ 에코 요청, 에코 응답 ICMP 메시지 정의

```
typedef struct _ICMPMESSAGE
{
    u_char  icmp_type;           // type of message
    u_char  icmp_code;          // type sub code
    u_short icmp_cksum;          // checksum
    u_short icmp_id;             // identifier
    u_short icmp_seq;            // sequence number
    ...                          // 옵션 데이터
} ICMPMESSAGE;
```



Traceroute (1)

❖ Traceroute 응용 프로그램

- 호스트나 라우터까지의 IP 패킷 전달 경로를 확인
- ICMP 프로토콜을 이용하여 구현

```
관리자: Windows 명령 처리기
C:\>tracert www.copyright.com
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
^C
C:\>tracert www.kpu.ac.kr
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
[오류] Request timed out!
^C
C:\>
```

```
관리자: Windows 명령 처리기
배치 파일이 아닙니다.

C:\>tracert www.kpu.ac.kr

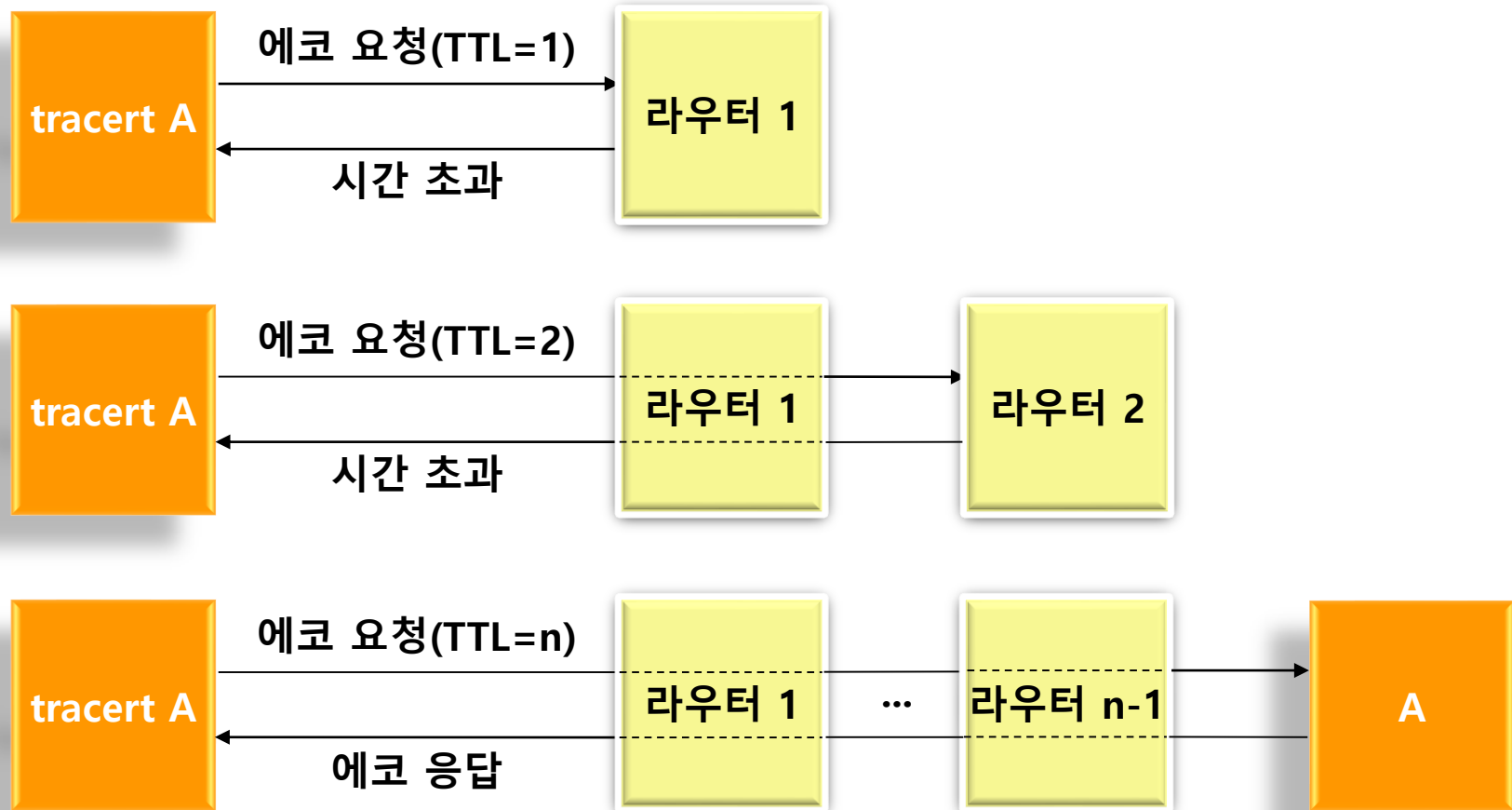
최대 30홉 이상의
www.kpu.ac.kr [210.93.48.95](으)로 가는 경로 추적:

  1  <1 ms    <1 ms    1 ms  10.0.0.1
  2   4 ms     5 ms     5 ms  14.39.186.254
  3   2 ms     2 ms     2 ms  118.33.12.201
  4   7 ms     9 ms    19 ms  112.188.58.101
  5   2 ms     1 ms     2 ms  112.188.53.89
  6   4 ms     3 ms     3 ms  220.73.149.37
  7   3 ms     2 ms     2 ms  220.73.153.210
  8   4 ms     3 ms     2 ms  112.189.71.58
  9   3 ms     4 ms     3 ms  112.189.161.254
 10   3 ms     2 ms     4 ms  222.121.143.11
 11   *        *        *      요청 시간이 만료되었습니다.
 12   *        *        ^C

C:\>tracert www.kpu.ac.kr
[오류] Request timed out!
```

Traceroute (2)

❖ Traceroute 동작 원리



❖ IP_TTL 또는 IPV6_UNICAST_HOPS 옵션 설정

```
#include <ws2tcpip.h>
```

```
...
```

```
// IPv4의 TTL 변경
```

```
int optval = TTL 값;
```

```
setsockopt(sock, IPPROTO_IP, IP_TTL, (char *)&optval, sizeof(optval));
```

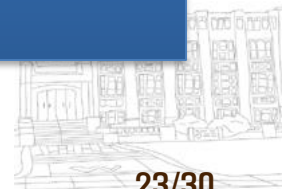
```
#include <ws2tcpip.h>
```

```
...
```

```
// IPv6의 TTL 변경
```

```
int optval = TTL 값;
```

```
setsockopt(sock, IPPROTO_IPV6, IPV6_UNICAST_HOPS,  
            (char *)&optval, sizeof(optval));
```



❖ IcmpCreateFile() 함수

```
HANDLE IcmpCreateFile(void) ;  
성공: 핸들, 실패: INVALID_HANDLE_VALUE
```



❖ IcmpSendEcho() 함수

```
DWORD IcmpSendEcho (  
    HANDLE IcmpHandle,  
    ULONG DestinationAddress,  
    LPVOID RequestData,  
    WORD RequestSize,  
    PIP_OPTION_INFORMATION RequestOptions,  
    LPVOID ReplyBuffer,  
    DWORD ReplySize,  
    DWORD Timeout  
);
```

성공: ReplyBuffer에 저장된 ICMP_ECHO_REPLY 구조체 개수
실패: 0



❖ 관련 구조체

```
typedef struct {  
    unsigned char Ttl;           // Time To Live  
    unsigned char Tos;          // Type Of Service  
    unsigned char Flags;        // IP header flags  
    unsigned char OptionsSize;  // Size in bytes of options data  
    unsigned char *OptionsData; // Pointer to options data  
} IP_OPTION_INFORMATION, *PIP_OPTION_INFORMATION;
```

```
typedef struct {  
    DWORD Address;               // Replying address  
    unsigned long Status;        // Reply status  
    unsigned long RoundTripTime; // RTT in milliseconds  
    unsigned short DataSize;     // Echo data size  
    unsigned short Reserved;     // Reserved for system use  
    void *Data;                 // Pointer to the echo data  
    IP_OPTION_INFORMATION Options; // Reply options  
} IP_ECHO_REPLY, *PIP_ECHO_REPLY;
```



❖ IcmpCloseHandle() 함수

```
BOOL IcmpCloseHandle (  
    HANDLE IcmpHandle  
);
```

성공: TRUE, 실패: FALSE



❖ DLL 조작 함수

```
HMODULE LoadLibrary (  
    LPCTSTR lpFileName // DLL 파일 이름  
);
```

성공: 핸들, 실패: NULL

```
FARPROC GetProcAddress (  
    HMODULE hModule,  
    LPCSTR lpProcName  
);
```

성공: 함수 주소, 실패: NULL

```
BOOL WINAPI FreeLibrary(  
    HMODULE hModule  
);
```

성공: 0이 아닌 값, 실패: 0



❖ ICMP.DLL 활용

- 기존 ping 예제와 결과는 비슷
- 방화벽이 있거나 관리자 레벨(그룹) 사용자 타입이 아닐경우 제한이 있을수 있음





Thank You !

oasis01@gmail.com / rhqudtn75@nate.com