TCP UDP 프로토콜에 대한 이해 & 주소 체계 실습

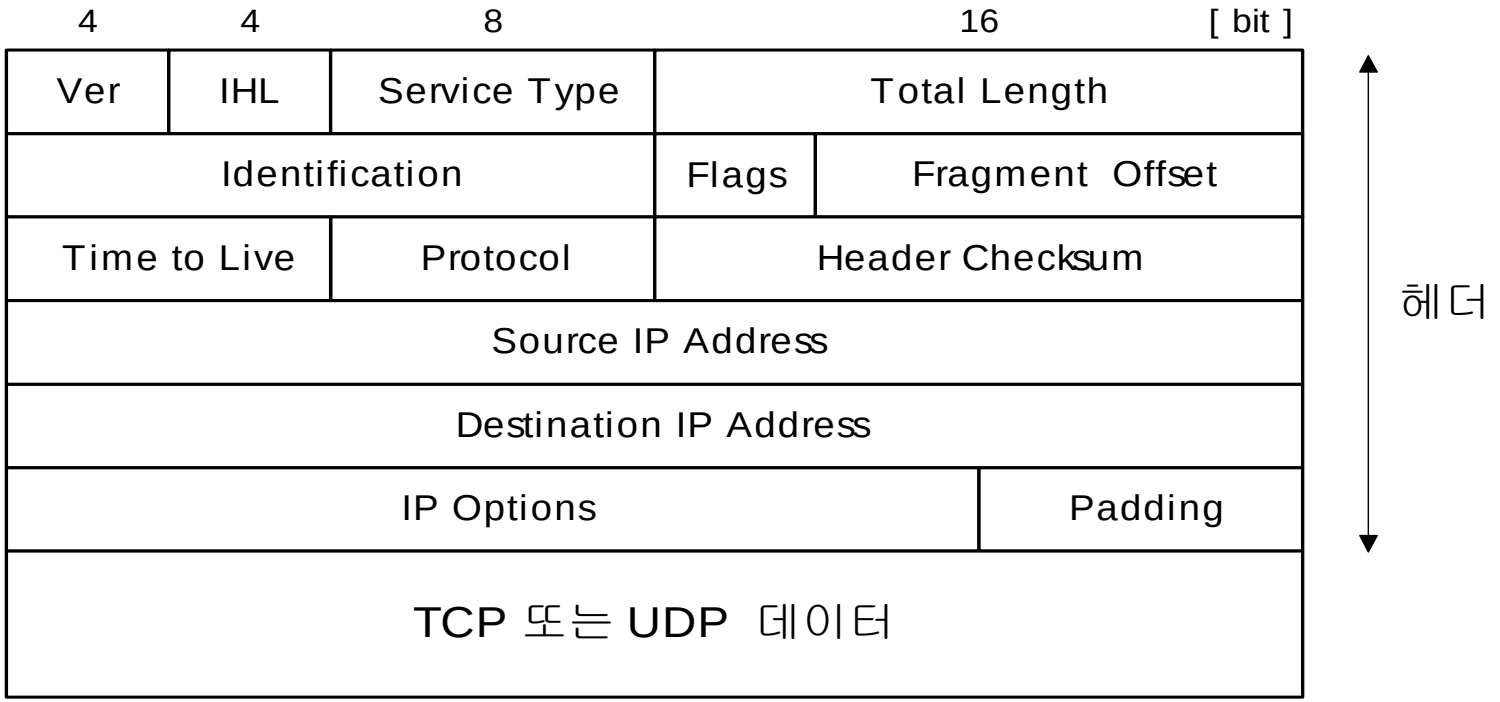
인하공업전문대학 컴퓨터정보과
최효현 교수

주요사항

- IP 패킷 및 특징
- TCP
- UDP

- 소켓 주소 부여 방법
- 주소 정보 (sockaddr_in 구조체)
- 네트워크 바이트 순서 vs. 호스트 바이트 순서

IP 프로토콜 동작



IHL : IP Header Length
Ver : Version

IP 패킷 헤더의 기능

필드 명		길이(비트)	기 능
Ver(Version)		4	IP 버전 값을 표시(현재는 4임)
IHL		4	4바이트 단위로 헤더 길이를 표시(최소값은 5)
Service Type		8	서비스 클래스 지정(보통 0으로 지정)
Total Length		16	IP 패킷 크기(바이트 단위, 최대값은 65535)
Identification		16	수신 측에서 패킷을 재조립할 때 사용하는 고유번호
Flags	미사용	1	미사용(항상 0)
	DF	1	세분화 금지 플래그(0:허용, 1:금지)
	More	1	패킷을 중간 라우터가 재조립할 때 사용 (0:마지막패킷, 1:연속되는패킷)
Fragment Offset		13	전체 메시지 중 이 패킷의 위치를 표시(8바이트 단위)
TTL(Time To Live)		8	패킷이 통신망 내에서 계속 돌아다니는 것을 방지하기 위하여 사용 보통 hop counter 값을 사용
Protocol		8	데이터를 전달할 상위 계층 프로토콜을 지정(1:ICMP, 6:TCP, 17:UDP)
Header Checksum		16	헤더 부분의 오류 검출
Source IP Address		32	송신지 IP 주소
Destination IP Address		32	수신지 IP 주소
IP Options		가변	옵션 선택
Padding		가변	32비트 단위로 헤더의 길이를 맞춤

IP 프로토콜 동작

- Maximum Transmission Unit(MTU)
  - ✓ 통신망이 한번에 전달할 수 있는 패킷의 최대크기

- Path MTU
  - ✓ 두 호스트 사이의 최소 MTU
  - ✓ 상위 계층으로부터 받은 메시지 크기가 path MTU보다 큰 경우 여러 개의 IP 패킷으로 분할

Network type	MTU(bytes)
Hyperchannel	65535
Token ring(IBM)	17914
Token ring(IEEE 802.5)	4464
FDDI	4352
Ethernet	1500
IEEE 802.3/802.2	1492
X.25	576
Point-to-Point	296

트랜스포트 계층

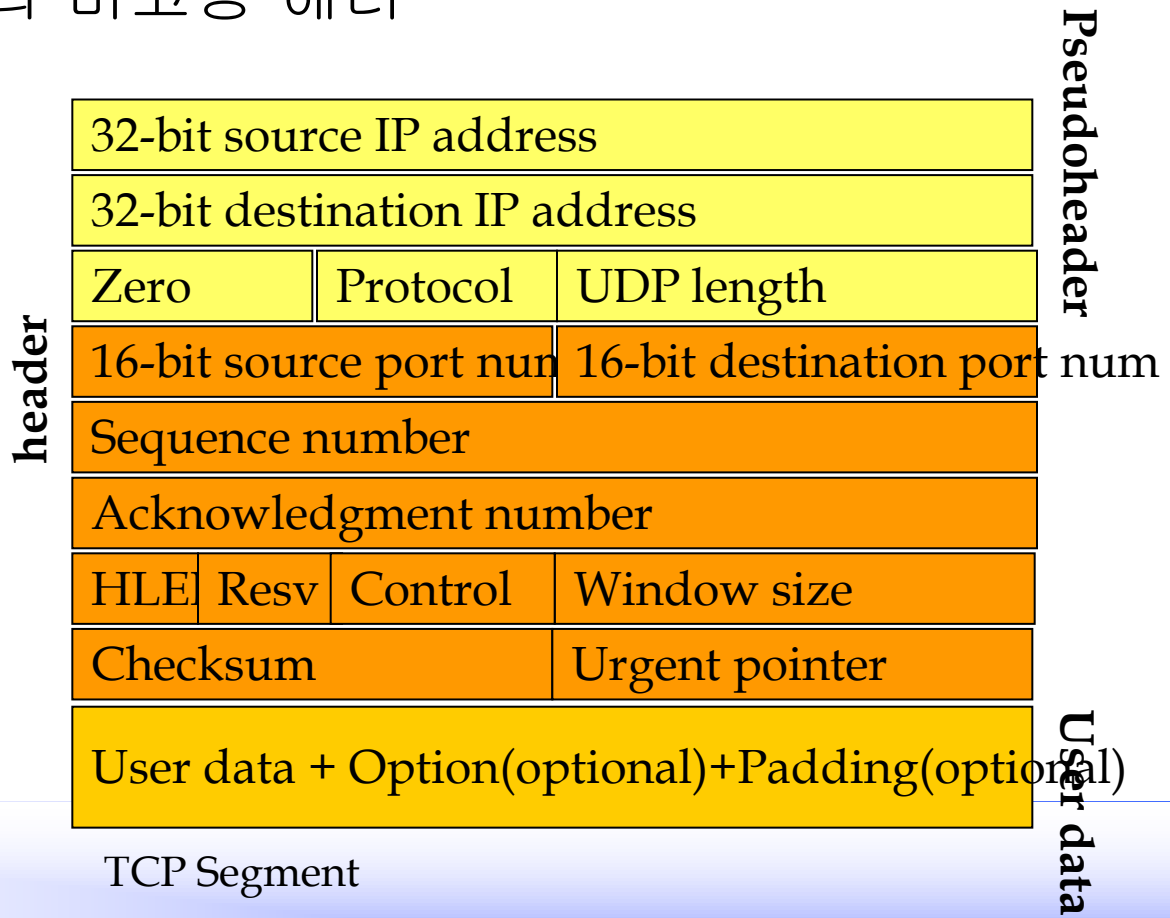
- **TCP : 연결형 서비스**
  - ✓ 종점간의 연결 개설, 오류 발생시 패킷 재전송, 패킷 전달 순서 확인, 중복 패킷 제거, 흐름제어, 네트워크 오동작시 보고 등을 제공하는 서비스
  - ✓ 연속된 흐름(Stream)의 데이터 송수신이 가능
  - ✓ 큰 파일 전송시 : 데이터가 중간에 끊어지지 않음
  - ✓ 한번에 많은 양의 데이터를 전송할 때나 신뢰성 있는 통신이 필요할 때, 또는 데이터의 순서 보장이 필요할 때 사용
  - ✓ ftp, telnet, http에서 TCP를 사용

트랜스포트 계층

- UDP : 비연결형 서비스
  - ✓ 연결형 서비스를 제공하지 않고 단순히 패킷을 하나씩 목적지 주소로 전송만 함
  - ✓ 프로토콜 헤더의 크기가 작고 연결 지연이 없으므로 간단한 패킷을 주고받는 경우에 유리
  - ✓ UDP를 사용해야만 하는 경우
    - ➢ 서버 프로그램이 UDP만을 사용하도록 작성되어 있는 경우
    - ➢ 패킷을 브로드캐스트 또는 멀티캐스트해야 하는 경우
    - ➢ TCP 오버헤드가 커서 TCP로 처리할 시간이 부족한 경우(실시간 서비스 등)

TCP 헤더

- TCP segment
  - ✓ 20-60 bytes의 비고정 헤더



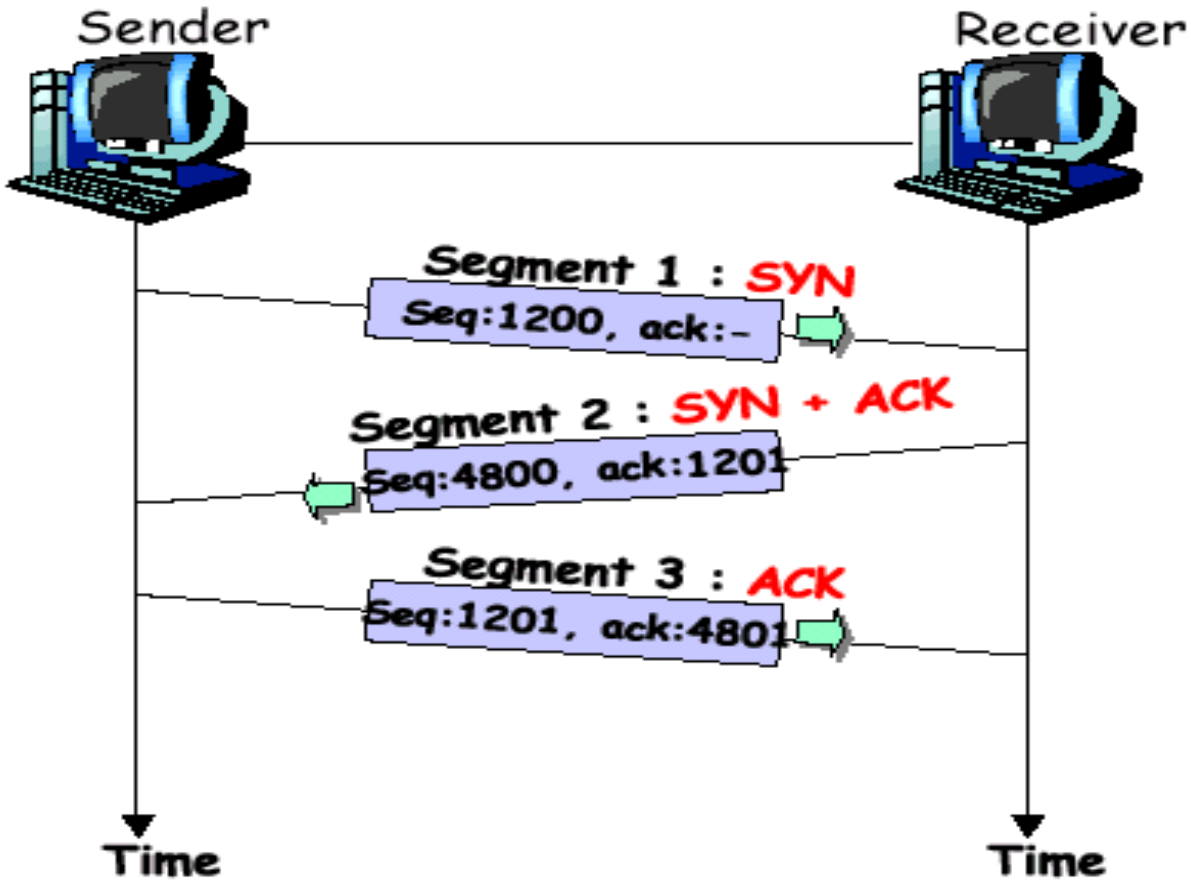
TCP 헤더의 각 필드 기능

필 드 명		길이(비트)	기 능
Source Port		16	송신측의 응용 프로세스를 구분하는 포트 번호
Destination Port		16	수신측의 응용 프로세스를 구분하는 포트 번호
Sequence Number		32	송신 데이터의 순서 번호(바이트 단위)
Ack Number		32	수신한 데이터의 순서 번호 + 1 (다음에 받고자 하는 순서 번호)
Header Length		4	헤더 크기(4바이트 단위)로 보통 5임
Rsvd		6	현재는 사용하지 않으며 0으로 세트
Code Bits	URG	1	긴급 데이터임을 표시(이때, Urgent Pointer 값이 유효)
	ACK	1	Ack Number 값이 유효함을 표시
	PSH	1	Push 기능을 표시
	RST	1	연결을 리셋하는 데 사용
	SYN	1	연결 요청시 사용되며, Sequence Number가 초기값임을 알림
	FIN	1	데이터 전송 종료
Window		16	흐름제어용 윈도우 크기(바이트 단위)
Checksum		16	TCP PDU 전체와 IP 계층의 헤더 중 후반부 12 바이트(송수신자 IP 주소 등)에 대한 오류 검출 코드
Urgent Pointer		16	긴급 데이터가 들어있는 위치를 표시

TCP 연결 설정

- TCP : 3-way handshake 사용
  - ✓ 2-way handshake
    - ➢ OSI 표준인 X.25의 네트워크 계층은 2-way handshake 사용
    - ➢ 연결 요청에 대하여 상대방이 확인을 하면 연결이 이루어짐
  - ✓ 3-way handshake
    - ➢ 연결 요청에 대하여 상대방도 이를 확인하면서 다시 새로운 연결 요청을 하고 이에 대해 처음의 연결 요구측에서 확인을 보내야 연결이 이루어짐

3-way handshake



TCP 데이터 전송

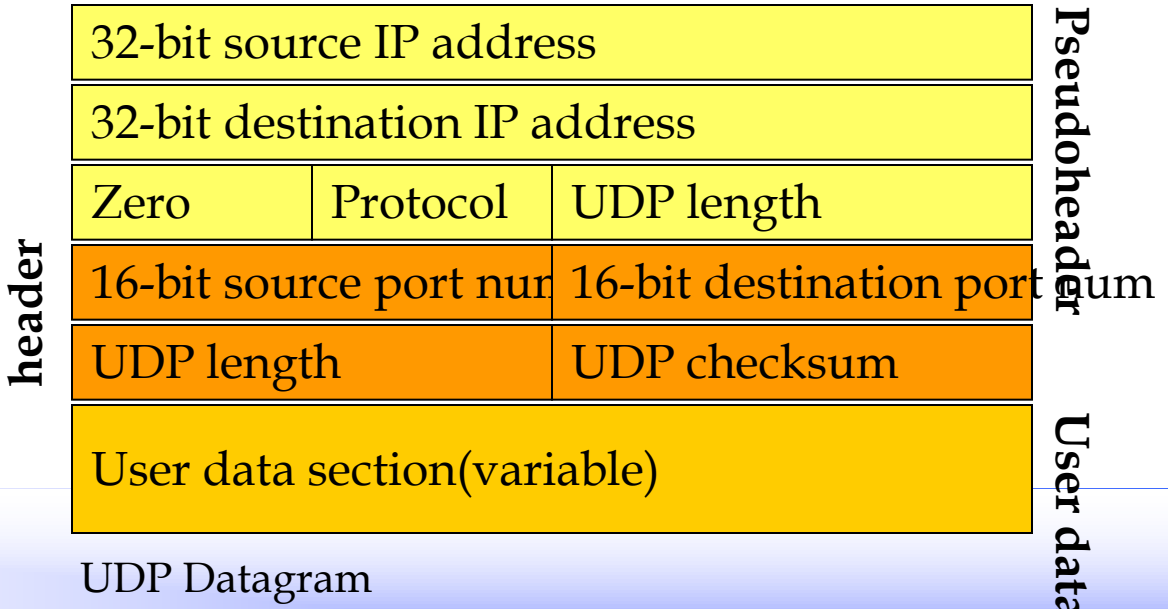
- Window
  - ✓ 상대방이 최근에 보내온 TCP 헤더의 Window 크기 내에서만 TCP 데이터를 전송
  - ✓ TCP는 자신의 데이터 처리 능력과 네트워크 사정에 따라 Window 값을 바꿈으로써 흐름제어
- **포트 번호**
  - ✓ 호스트 내에서 TCP/IP를 이용하는 응용 프로세스들을 구별
  - ✓ Well-known 포트 번호 : 1023번 이하의 번호
  - ✓ 응용 프로그램에서 임의로 정하여 사용하는 포트 번호 : 1024번 이상의 번호

Port numbers

- Port number
  - Well-known port
    - 서버의 반영구적인 port
    - 각 프로토콜마다 IANA에 의해서 유일하게 할당됨
    - 주로, 1-1023까지 해당
  - Ephemeral port
    - 클라이언트의 임시적인 port
    - 클라이언트가 서버와 통신하기 위해서 생성
    - 주로, 1024부터 5000까지 해당
      - TCP, UDP는 32768부터 사용
  - Reserved port
    - Unix 에서 사용되는 port

UDP PDU

- User datagram
  - ✓ 8 bytes 고정 크기 헤더
  - ✓ Pseudoheader(optional) + UDP header + User data
  - ✓ Checksum은 헤더와 데이터를 함께 처리
    - ➢ IP헤더의 checksum은 헤더만 처리



UDP 특징

- Connectionless service
  - ✓ 전달하기 이전에 connection 관리가 이루어지지 않으며, datagram에 번호(number)가 존재하지 않으며, 다른 경로를 통하여 datagram이 전송
- Flow & error control 없음
  - ✓ 비신뢰적 전송 프로토콜
  - ✓ Checksum을 제외한 어떠한 메커니즘도 존재하지 않음
- Applications
  - ✓ TFTP(Trivial File Transfer Protocol)
  - ✓ Multicasting, broadcasting
  - ✓ SNMP(Simple Network Management Protocol), RIP(Routing Information Protocol)

소켓 주소

소켓의 주소는?

주소 정보 = IP address + port 번호

www.inhatc.ac.kr , web 서버

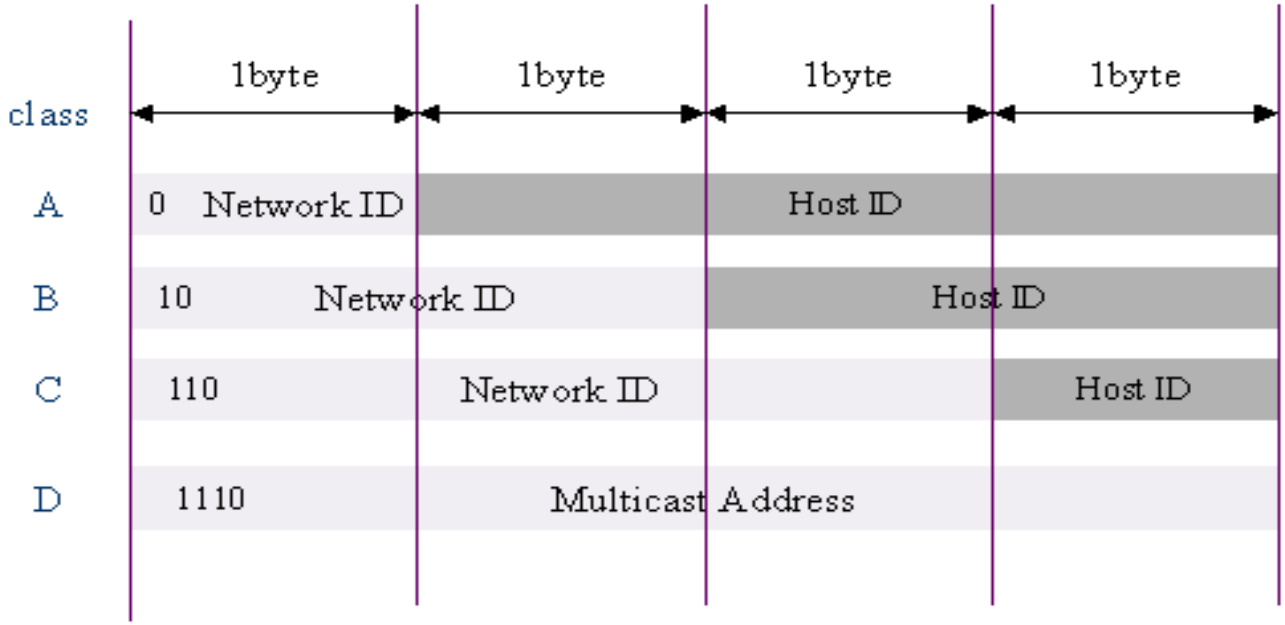
221.154.90.151 port 번호: 80

0xdd9a5a97 0x0050

0x975a9add 0x5000

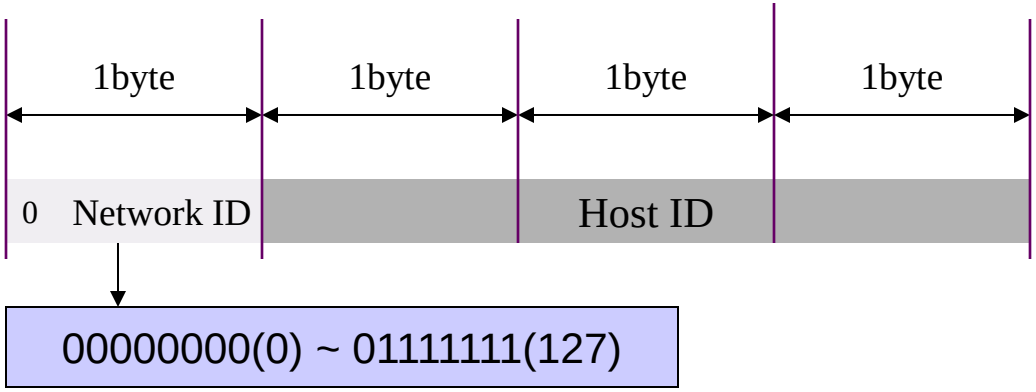
- **Internet Address**

1. 인터넷상에 존재하는 호스트를 구분하기 위한 **32**비트 주소 체계

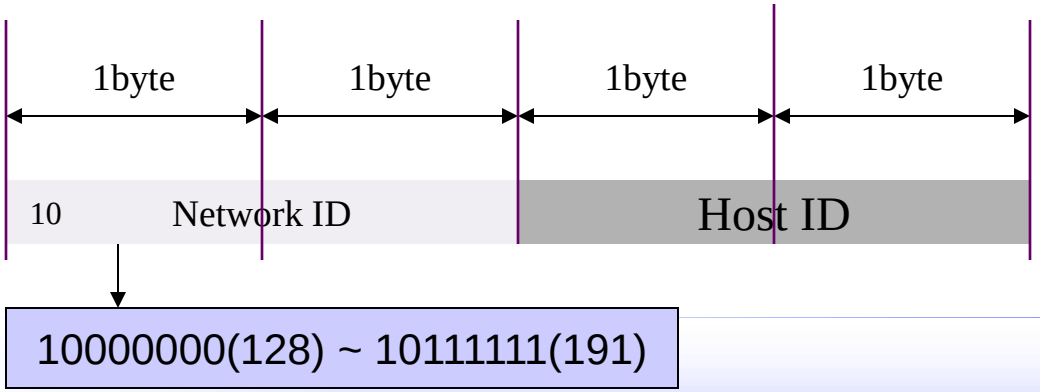


2. 클래스

- class A

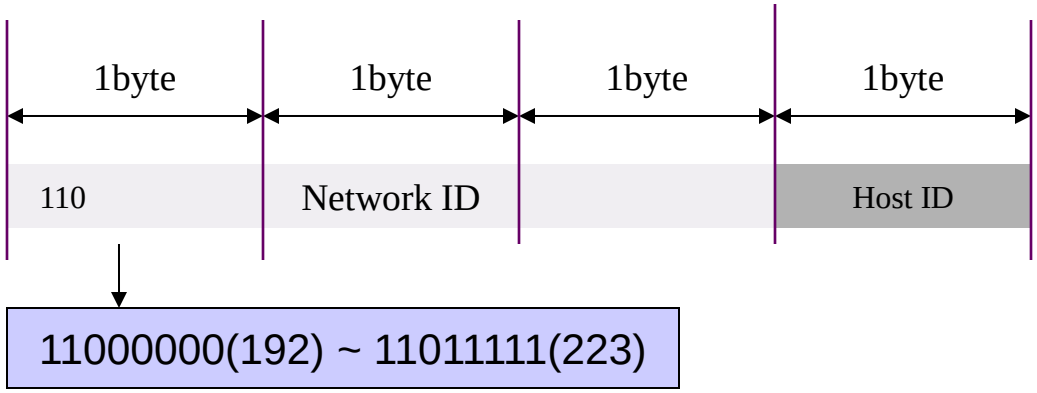


- class B



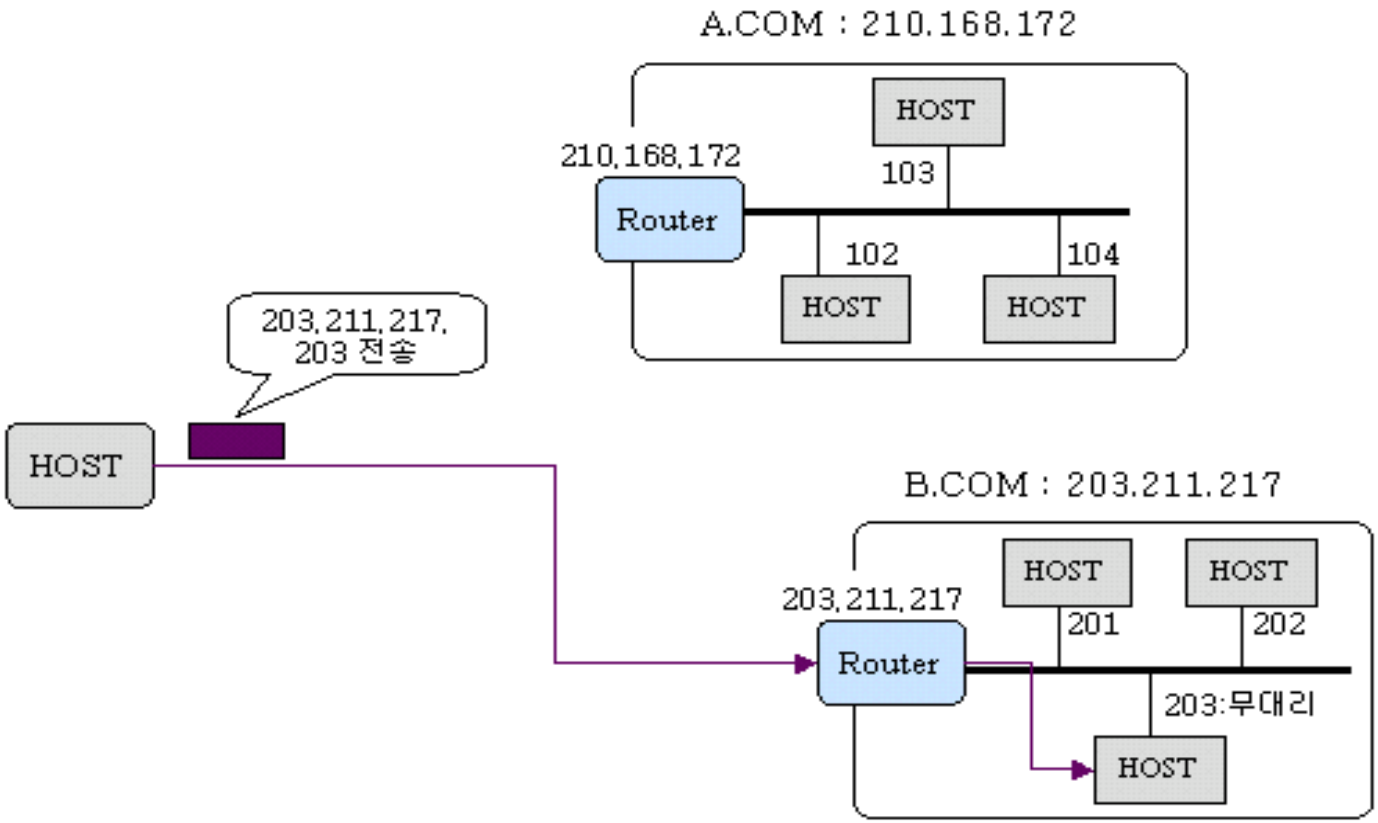
3.1 인터넷 주소개요

- class C



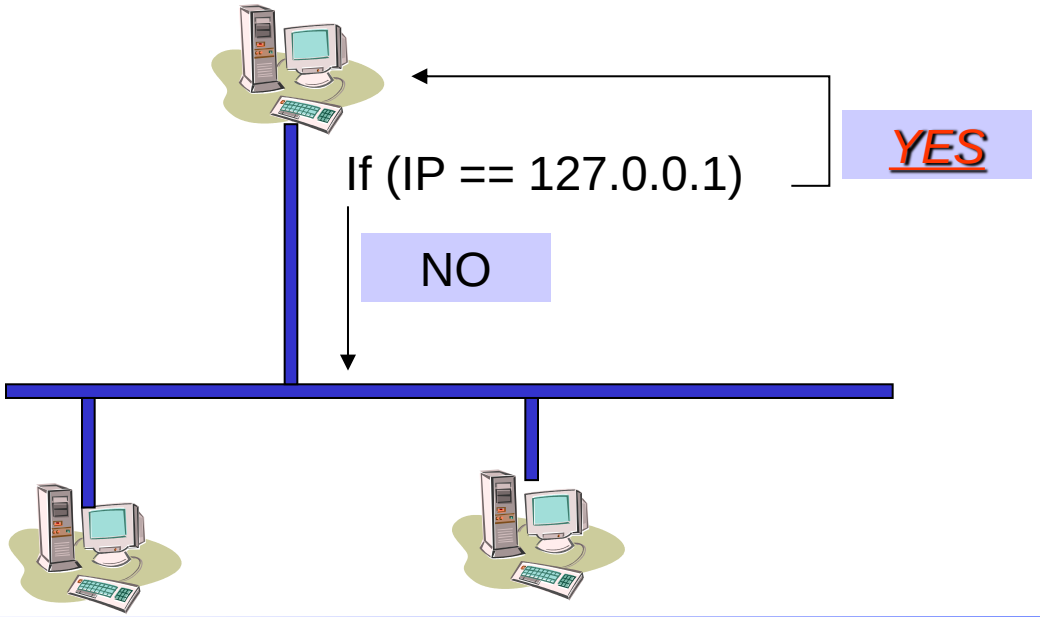
3.1 인터넷 주소개요

3. 네트워크 주소(네트워크 ID) + 호스트 주소(호스트 ID)



4. 루프백 주소(loop back address)

- 127.0.0.1
- 한 PC에서 서버와 클라이언트 모두를 프로그래밍 가능



3.1 인터넷 주소개요

- **Port란 무엇인가?**

1. 호스트 내에서 실행되고 있는 프로세스**(Process)**를 구분 짓기 위한 **16** 비트의 논리적 할당**(**소켓에 할당된다**).**
   - 16비트이므로 값의 범위가 0 ~ 65535 이다.

2. **Well-known ports : 0 ~ 1023,** 사용자는 **1023**번 이후 대 사용

3. 포트는 중복될 수 없으나**, TCP** 소켓과 **UDP** 소켓은 **Port**를 공유 하지 않으므로 중복되어도 무방함
   (예**, TCP** 소켓 **:** 포트번호 **9190,** 다른 **TCP** 소켓 **:** 포트번호 **9191,**
   UDP 소켓 **:** 포트번호 **9190** 사용가능 함**)**

3.2 주소정보의 표현

- **IPv4의 주소체계를 나타내는 구조체**
  - **sockaddr 구조체 생성과정**

```
struct sockaddr {
        sa_family_t         sin_family;
        char                sa_data[14];
};
```

범용적인 주소 구조체
(page 86 참조)

- 프로토콜 체계에 따라 주소체계가 다름
- socket(), bind() 함수는 특정 프로토콜을 위한 함수가 아님
- 따라서 범용적인 구조체가 필요

3.2 주소정보의 표현

- **IPv4의 주소체계를 나타내는 구조체**

Unsigned 32bit int형 (sys/types.h에 선언)

```
struct in_addr {
        uint32_t        s_addr;      /* 32비트 IP 주소정보 */
};
```

```
struct sockaddr_in {
        sa_family_t     sin_family;  /* 주소 체계(AF_INET) */
        uint16_t        sin_port;    /* port 정보, 16비트 */
        struct in_addr  sin_addr;    /* 32 비트 IP 주소정보 */
        char            sin_zero[8]; /* 사용되지 않음 */
};
```

(자료형은 표 3-1 참조)

☛주의 : 모든 데이터는 네트워크 바이트 순서로 저장 해야 한다.

3.2 주소정보의 표현

- **sockaddr_in의 구조체 정보**

1. sin_family;
   - 프로토콜체계마다 주소체계가 다름
   - 주소체계(address family) : *AF_INET - IPv4 주소체계*
     AF_INET6 - IPv6 주소체계
     AF_LOCAL - local 통신 unix 주소체계
2. sin_addr;
   - 32비트 IP 주소정보, 네트워크 바이트 순서로 저장
3. sin_port;
   - 16 비트 포트정보, 네트워크 바이트 순서로 저장
4. sin_zero;
   - 특별한 의미 없음, 0으로 padding.

3.3 네트워크 바이트 순서

- **sockaddr_in에 주소 대입시 주의점**

  - 네트워크 바이트 순서로 대입해야 함

  - 바이트 순서(byte order)?
    : 시스템이 내부적으로 데이터를 표현하는 방법

  - 2가지 방법
    : Big-endian방식과 Little-endian방식

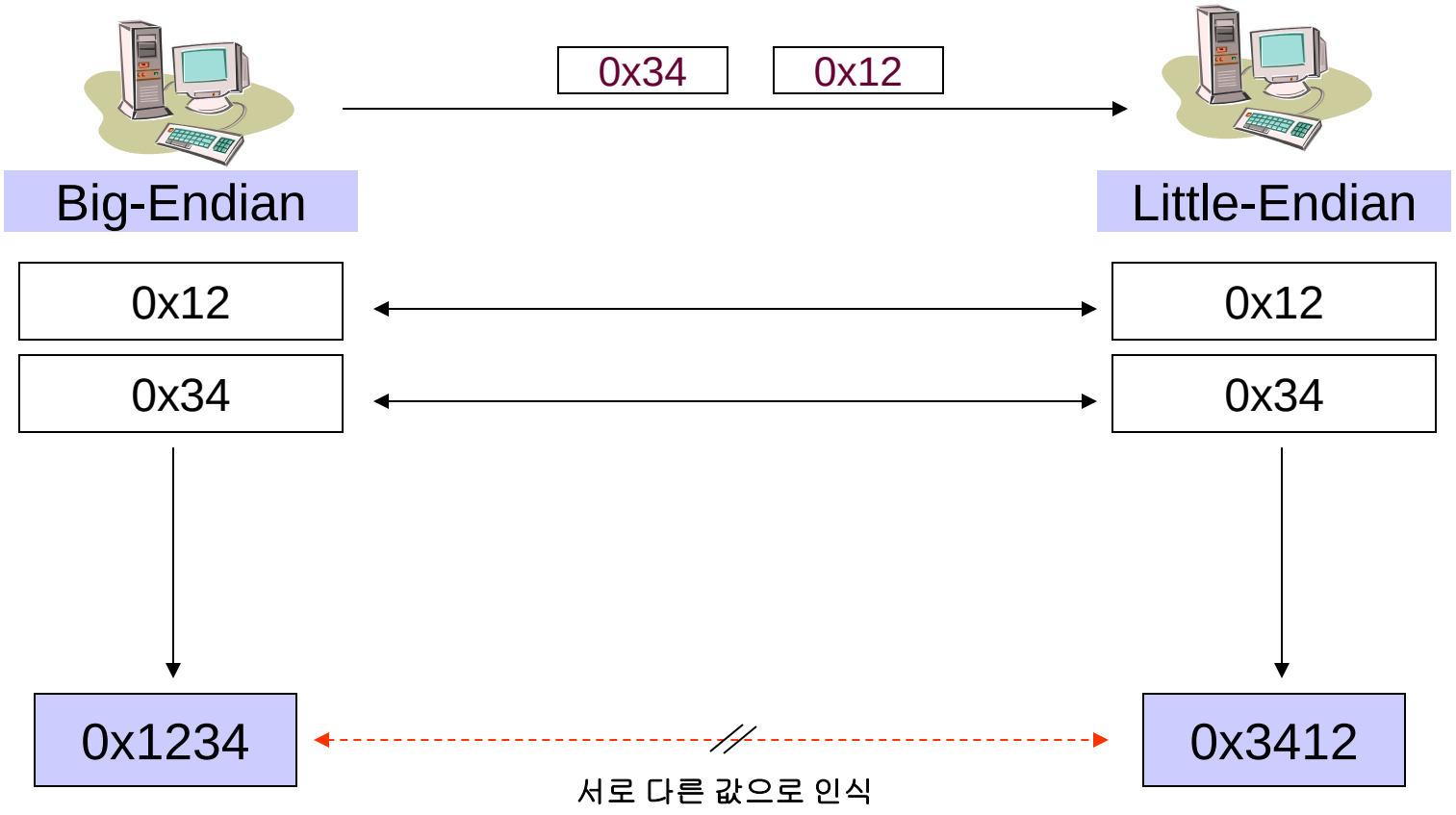
    **** 보통 사용하는 Intel CPU를 사용하는 PC는 거의다 Little-endian 방식**
    **** 반면, network order는 Big-endian 방식**

3.3 네트워크 바이트 순서

- 호스트 바이트 순서(**Host Byte Order**)
  - 시스템이 내부적으로 데이터를 처리하는 방식으로 **CPU**에 따라 다름

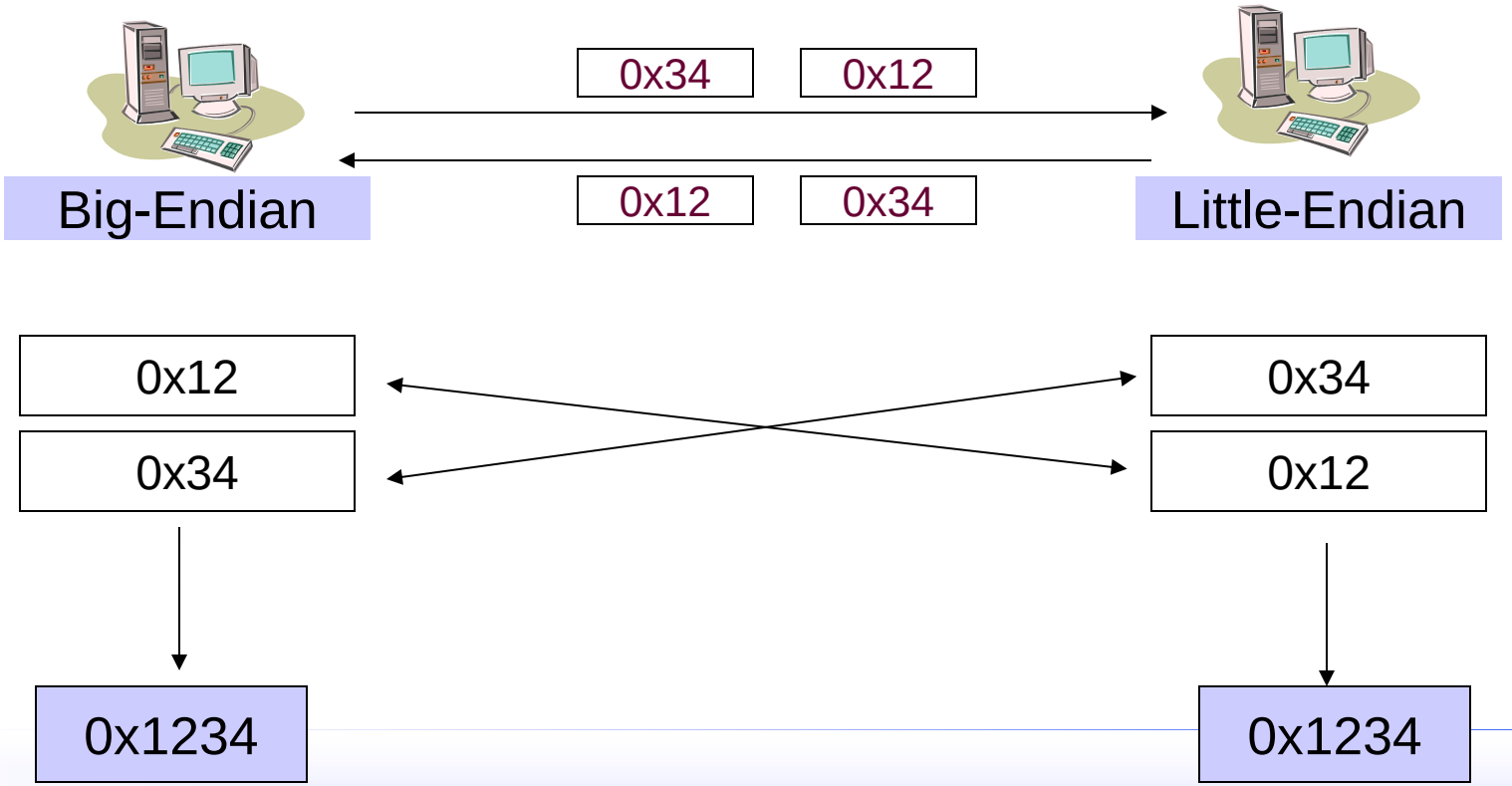
3.3 네트워크 바이트 순서

- 데이터 표현 방식에 따른 문제점



3.3 네트워크 바이트 순서

- 네트워크 바이트 순서(**Network Byte Order**)
  - 네트워크 바이트 순서는 **Big-Endian** 방식을 적용하기로 약속



3.3 네트워크 바이트 순서

- **바이트 순서 변환 함수(Endian Conversion)**
  - 데이터를 전송전에, sockaddr_in구조체에 값을 채우기 전에 Big-endian 방식으로 변환

```
unsigned short  htons(unsigned short);
unsigned short  ntohs(unsigned short);
unsigned long  htonl(unsigned long);
unsigned long  ntohl(unsigned long);
```

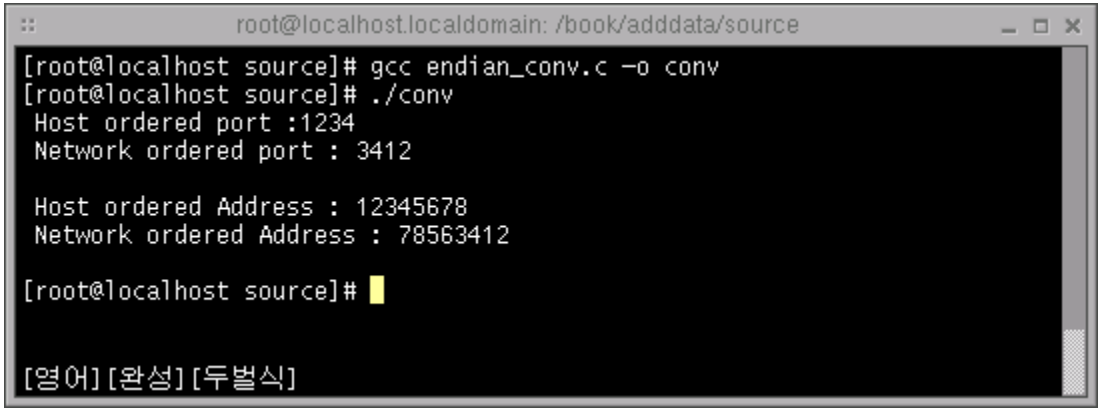
‘h’ : host byte order ‘n’ : network byte order
‘s’ : short (16 bit –포트정보) ‘l’ : long (32 bit – IP주소정보)

3.3 네트워크 바이트 순서

- 예제 확인

1. 프로그램 예제
  - endian_conv.c

2. 실행결과 : 두 출력결과가 다르다면, **Little-endian** 시스템임(대부분)



Q&A