바이트 순서(Byte Order)

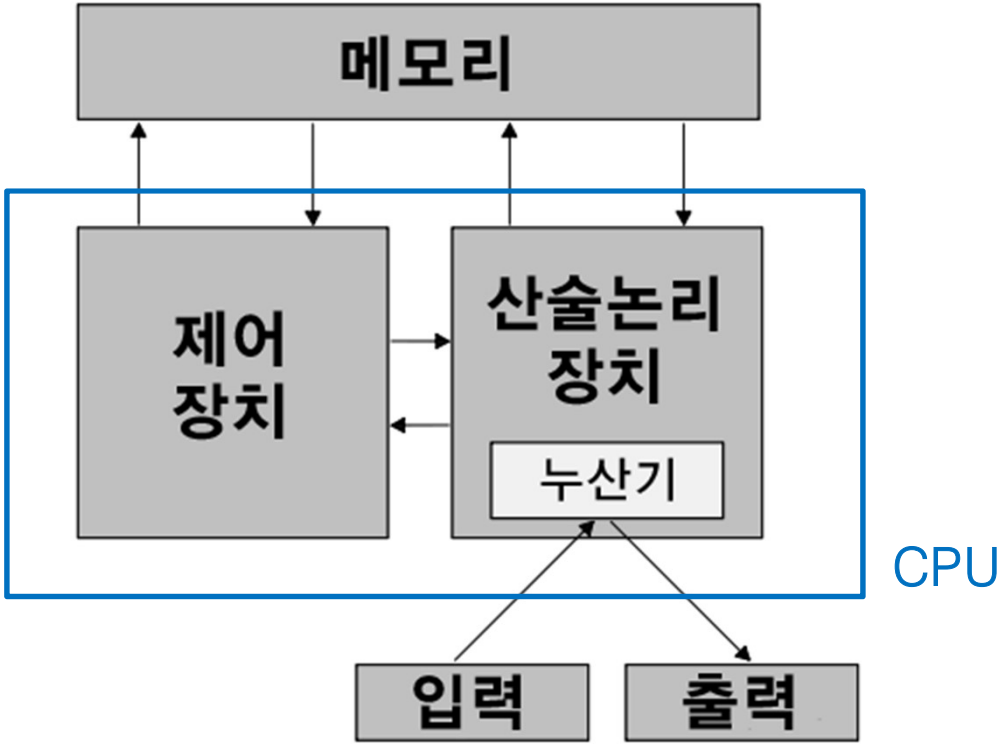
뇌를 자극하는 TCP/IP 소켓 프로그래밍

6장. 네트워크 바이트 순서

- 컴퓨터 기종 (CPU) 간 데이터 순서의 차이가 존재한다.
- 데이터를 읽고 쓸때의 순서를 바이트 순서라고 한다.
- 바이트 순서를 이해하고, 데이터 통신에 미치는 영향과 해결 방법을 찾는다.

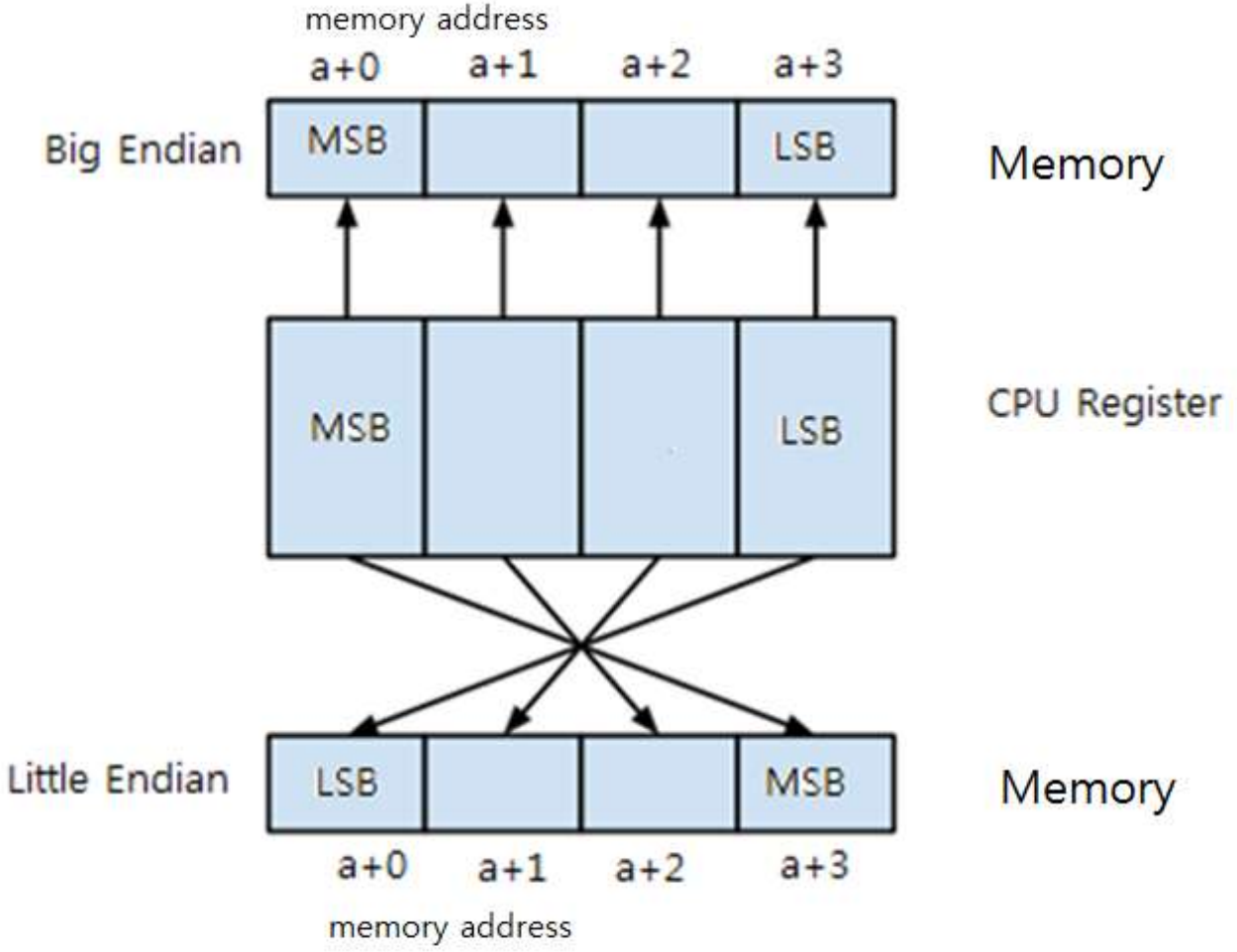
폰 노이만 구조(von Neuman)

- 존 폰 노이만이 고안
- 현재와 같은 CPU, 메모리, 프로그램 구조를 갖는 범용 컴퓨터 구조의 확립
- 내장 메모리 순차처리 방식
  - 데이터 메모리와 프로그램 메모리가 구분되어 있지 않음



바이트 오더(byte order)

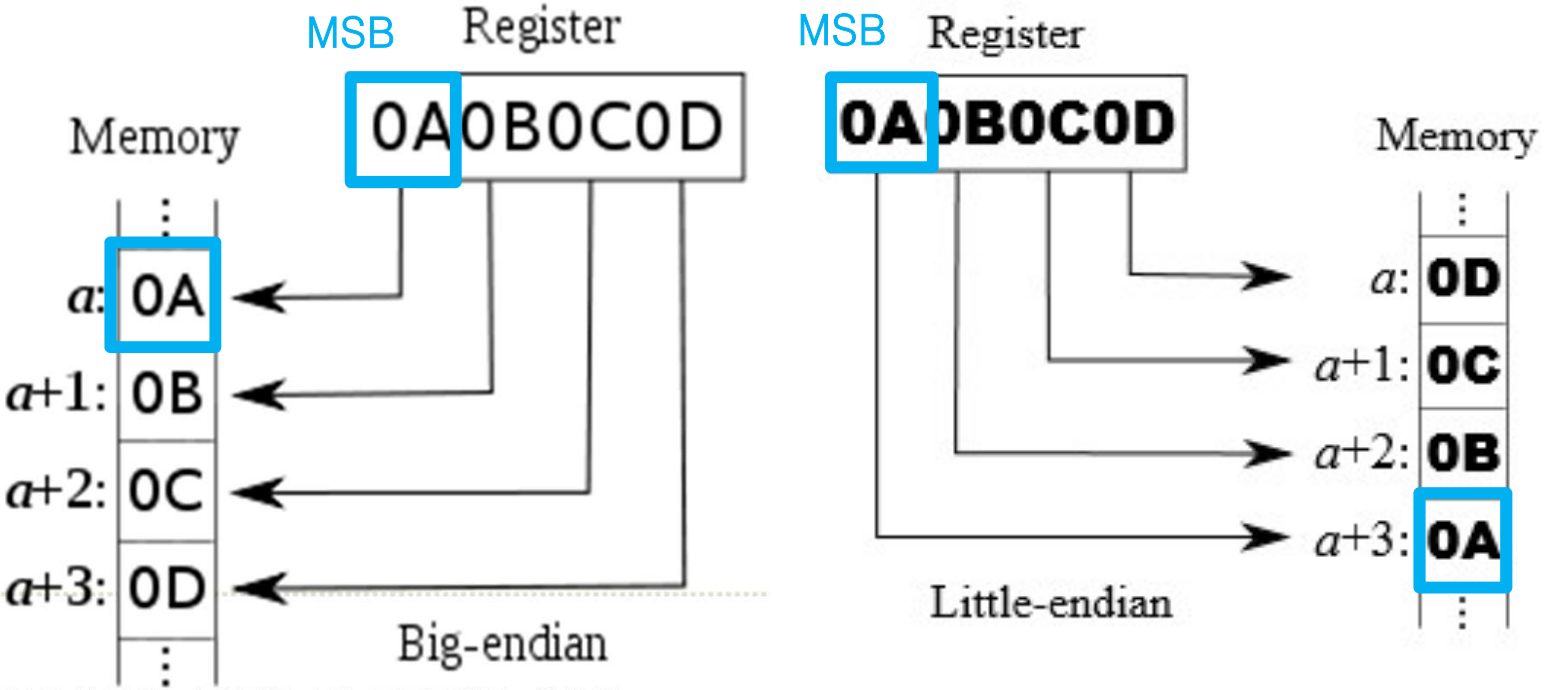
- CPU는 바이트 단위로 주소 (**byte addressable**) 가 주어지며 메모리에 데이터를 읽거나/쓴다.
- 이때 읽고/쓰는 순서에 따른 차이가 있다.
- 이것을 바이트 오더 혹은 바이트 순서라고 한다.



바이트 오더(byte order)

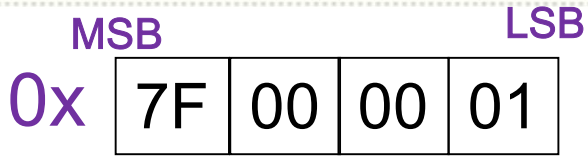
- “빅 엔디안(**Big Endian**)” 방식
  - 어떤 CPU는 가장 낮은 주소의 바이트 부터 먼저 읽음, 즉 메모리에서 가장 낮은 주소의 바이트가 CPU 레지스터의 **MSB(Most Significant Byte)**로 감
  - RISC Processor - Sparc, Motorola

- “리틀 엔디안(**Little Endian**)” 방식
  - 어떤 CPU는 가장 높은 주소의 바이트 부터 먼저 읽음, 즉 메모리에서 가장 낮은 주소의 바이트가 CPU 레지스터의 **LSB(Least Significant Byte)**로 감
  - CISC Processor - Intel 계열

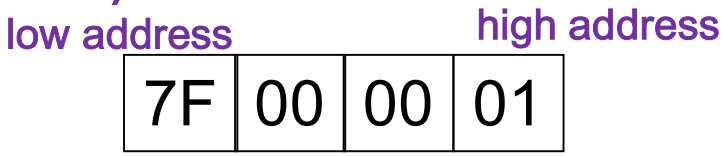


바이트 오더(byte order)

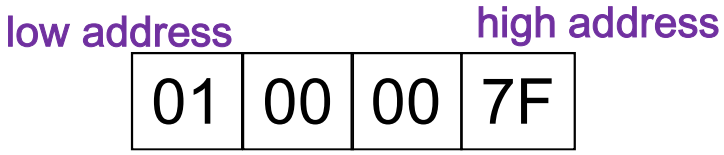
- **CPU register**
  - 127.0.0.1



- **Memory “빅 엔디안(Big Endian)” 방식**

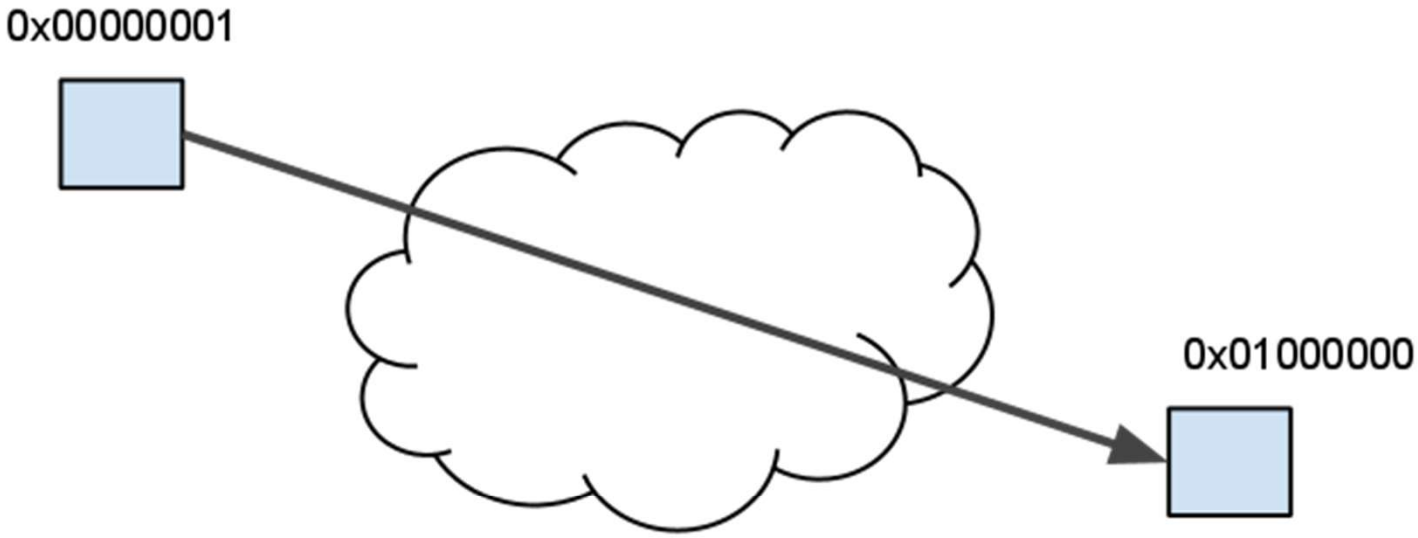


- **Memory “리틀 엔디안(Little Endian)” 방식**



바이트 오더(byte order)

- 인터넷은 다양한 CPU를 가진 컴퓨터로 구성 돼 있다.
- 서로 다른 엔디안 방식을 가진 CPU가 데이터를 주고 받을 경우 바이트를 역전해서 읽을 수 있는 경우를 고려 해야 한다.

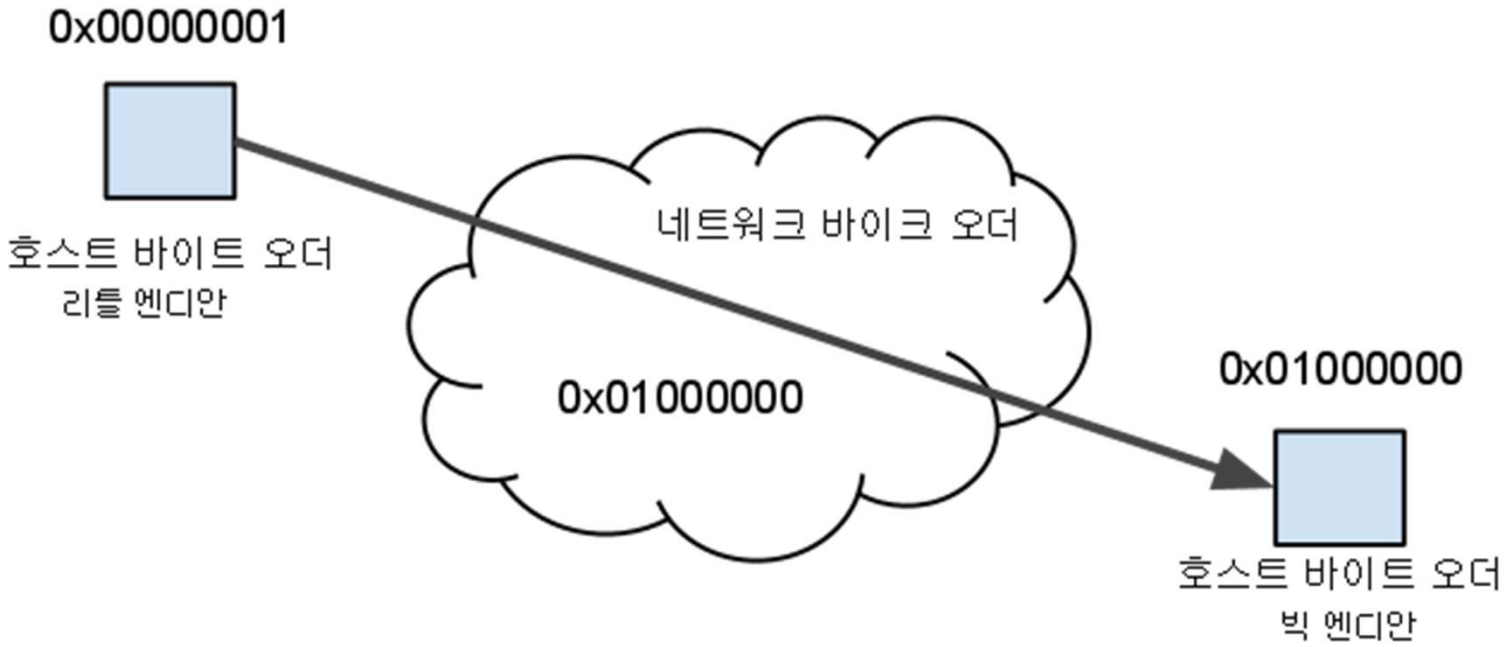


- 한쪽에서 0x00000001 데이터를 인터넷을 통하여 다른 컴퓨터에 전송 했다.
- 수신 컴퓨터와 송신 컴퓨터의 바이트 오더가 다를 경우
- 송신 측은 수신 데이터를 0x10000000으로 읽을 것이다.
- 전혀 엉뚱한 값을 재현하게 된다.

네트워크 바이트 오더

- 바이트 오더를 통일 : 네트워크 바이트 오더로 통일
- 네트워크 바이트 오더는 빅 엔디안을 따른다.

- 데이터를 전송 할 때, 빅엔디언 컴퓨터든 리틀 엔티언 컴퓨터든, 네트워크 바이트 오더로 변환후 송신
- 받은 측에서도, 빅엔디언 컴퓨터든 리틀 엔티언 컴퓨터든, 수신후 호스트 바이트 오더로 변환 후 응용에서 사용



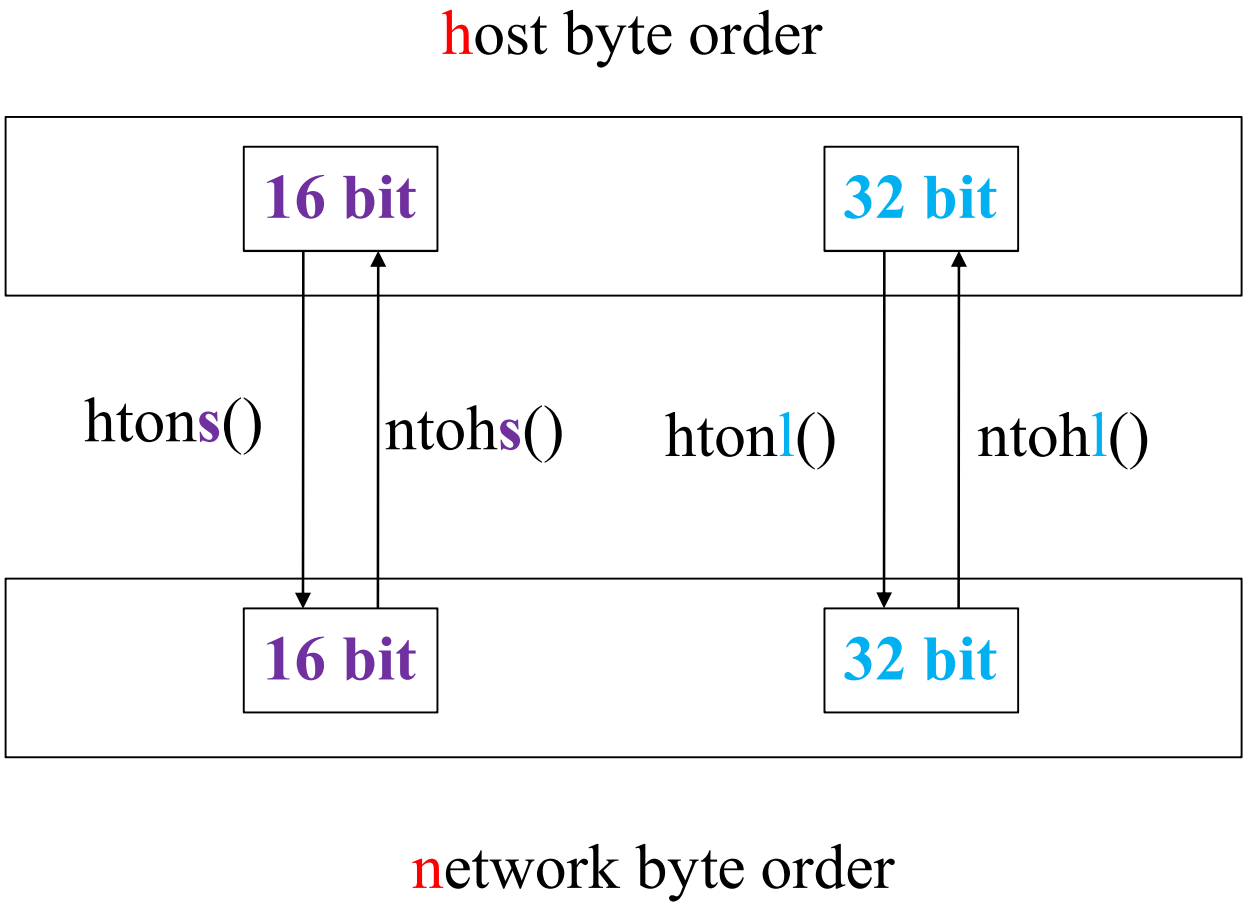
바이트 오더 변환 함수

- 바이트 오더는 2byte 이상의 데이터에서 고려해야 한다.
- 소켓은 바이트 오더를 위한 4종의 함수를 제공한다.

ntohs	2byte	network to host
htons	2byte	host to network
ntohl	4byte	network to host
htonl	4byte	host to network

- __s 계열 함수는 2byte 데이터에 사용한다.
- __l 계열 함수는 4byte 데이터 사용한다.

바이트 오더 변환 함수


host byte order
16 bit
32 bit
htons()
ntohs()
htonl()
ntohl()
16 bit
32 bit
network byte order

바이트 오더 변환 함수

- 바이트 변환 함수 예
- myinfo 구조체를 보낼 때, short, int 데이터에 대해서 바이트 오더 함수를 사용한다.
- name은 1byte 크기의 char array 이므로 바이트 오더에 신경쓰지 않아도 된다.

```
struct myinfo
{
    char name[80];
    short int age;
    int birthday;
};

myinfo.age = htons(myinfo.age);
myinfo.birthday = htonl(myinfo.birthday);

write(sockfd, &myinfo, sizeof(myinfo);
```

네트워크 주소와 바이트 오더 함수

- 인터넷으로 통신하는 모든 데이터에 대해서 바이트 오더를 신경써야 한다.
- 소켓 함수에 사용하는 여러 구조체의 데이터도 마찬가지.
- sin_port의 값에 대해소 htons를 하지 않으면 전혀 다른 포트로의 연결을 시도할 것이다.
- inet_addr과 같은 소켓 전용 함수들은 네트워크 바이트 오더를 따르는 값을 반환 한다.

```
struct sockaddr_in addr;

adrr.sin_family=AF_INET;
addr.sin_addr.s_addr = inet_addr("211.11.11.11")
addr.sin_port = htons(8080);

connect(sockfd, &addr, sizeof(addr));
```

Thank You !

뇌를 자극하는 TCP/IP 소켓 프로그래밍