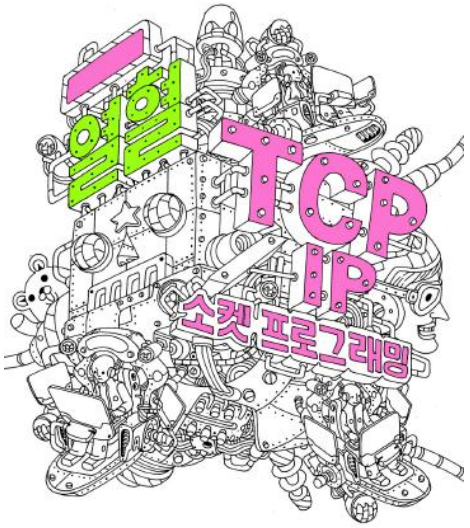
윤성우의 열혈 TCP/IP 소켓 프로그래밍

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

Chapter 09. 소켓의 다양한 옵션



Chapter 09-1. 소켓의 옵션과 입출력 버퍼의 크기

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

다양한 소켓의 옵션

Protocol Level	Option Name	Get	Set
SOL_SOCKET	SO_SNDBUF	O	O
	SO_RCVBUF	O	O
	SO_REUSEADDR	O	O
	SO_KEEPALIVE	O	O
	SO_BROADCAST	O	O
	SO_DONTROUTE	O	O
	SO_OOBINLINE	O	O
	SO_ERROR	O	X
	SO_TYPE	O	X
IPPROTO_IP	IP_TOS	O	O
	IP_TTL	O	O
	IP_MULTICAST_TTL	O	O
	IP_MULTICAST_LOOP	O	O
	IP_MULTICAST_IF	O	O
IPPROTO_TCP	TCP_KEEPALIVE	O	O
	TCP_NODELAY	O	O
	TCP_MAXSEG	O	O

소켓의 특성을 변경시킬 때 사용하는 옵션 정보들이다. 이러한 소켓의 옵션은 계층별로 분류된다. IPPROTO_IP 레벨의 옵션들은 IP 프로토콜에 관련된 사항들이며, IPPROTO_TCP 레벨의 옵션들은 TCP 프로토콜에 관련된 사항들이다. 그리고 SOL_SOCKET 레벨의 옵션들은 소켓에 대한 가장 일반적인 옵션들로 생각하면 된다.

옵션정보의 참조에 사용되는 함수

```
#include <sys/socket.h>

int getsockopt(int sock, int level, int optname, void *optval, socklen_t *optlen);
```

➜ 성공 시 0, 실패 시 -1 반환

- sock — 옵션확인을 위한 소켓의 파일 디스크립터 전달.
- level — 확인할 옵션의 프로토콜 레벨 전달.
- optname — 확인할 옵션의 이름 전달.
- optval — 확인결과의 저장을 위한 버퍼의 주소 값 전달.
- optlen — 네 번째 매개변수 optval로 전달된 주소 값의 버퍼크기를 담고 있는 변수의 주소 값 전달, 함수호출이 완료되면 이 변수에는 네 번째 인자를 통해 반환된 옵션정보의 크기가 바이트 단위로 계산되어 저장된다.

앞서 표에서 제시한 Protocol Level과 Option Name이 두 번째, 세 번째 인자로 전달되어, 해당 옵션의 등록 정보를 얻어온다.

옵션정보의 설정에 사용되는 함수

```
#include <sys/socket.h>

int setsockopt(int sock, int level, int optname, const void *optval, socklen_t optlen);
```

➜ 성공 시 0, 실패 시 -1 반환

- sock 옵션변경을 위한 소켓의 파일 디스크립터 전달.
- level 변경할 옵션의 프로토콜 레벨 전달.
- optname 변경할 옵션의 이름 전달.
- optval 변경할 옵션정보를 저장한 버퍼의 주소 값 전달.
- optlen 네 번째 매개변수 optval로 전달된 옵션정보의 바이트 단위 크기 전달.

앞서 표에서 제시한 Protocol Level과 Option Name 이 두 번째, 세 번째 인자로 전달하고, 해당 옵션의 등록정보를 변경한다.

소켓의 타입정보(TCP or UDP)의 확인

예제 sock_type.c의 일부

```
tcp_sock=socket(PF_INET, SOCK_STREAM, 0);
udp_sock=socket(PF_INET, SOCK_DGRAM, 0);
printf("SOCK_STREAM: %d \n", SOCK_STREAM);
printf("SOCK_DGRAM: %d \n", SOCK_DGRAM);

state=getsockopt(tcp_sock, SOL_SOCKET, SO_TYPE, (void*)&sock_type, &optlen);
if(state)
    error_handling("getsockopt() error!");
printf("Socket type one: %d \n", sock_type);

state=getsockopt(udp_sock, SOL_SOCKET, SO_TYPE, (void*)&sock_type, &optlen);
if(state)
    error_handling("getsockopt() error!");
printf("Socket type two: %d \n", sock_type);
```

소켓의 타입정보는 변경이 불가능하기 때문에, 옵션 SO_TYPE은 확인만 가능하고 변경이 불가능한 옵션이다.

실행결과

```
root@my_linux:/tcpip# gcc sock_type.c -o socktype
root@my_linux:/tcpip# ./socktype
SOCK_STREAM: 1
SOCK_DGRAM: 2
Socket type one: 1
Socket type two: 2
```

소켓의 입출력 버퍼 크기 확인

예제 get_buf.c의 일부

```
sock=socket(PF_INET, SOCK_STREAM, 0);
len=sizeof(snd_buf);
state=getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void*)&snd_buf, &len);
if(state)
    error_handling("getsockopt() error");

len=sizeof(rcv_buf);
state=getsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void*)&rcv_buf, &len);
if(state)
    error_handling("getsockopt() error");

printf("Input buffer size: %d \n", rcv_buf);
printf("Output buffer size: %d \n", snd_buf);
```

입력버퍼의 크기를 참조 및 변경할 때에는 SO_SNDBUF, 출력버퍼의 크기를 참조 및 변경할 때에는 SO_RCVBUF 를 사용한다.

```
root@my_linux:/tcpip# gcc get_buf.c -o getbuf
root@my_linux:/tcpip# ./getbuf
Input buffer size: 87380
Output buffer size: 16384
```

실행결과

소켓의 입출력 버퍼 크기 변경

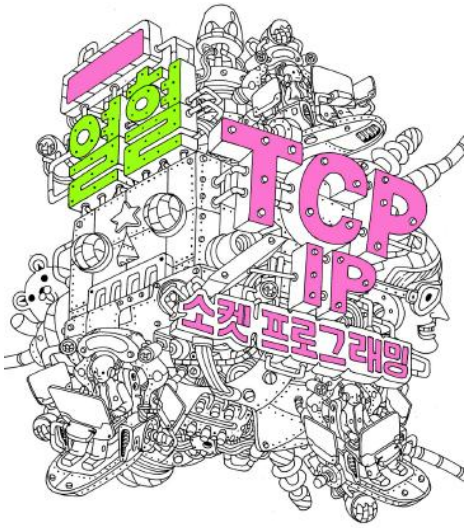
예제 get_buf.c의 일부(에러 처리 코드는 생략)

```
sock=socket(PF_INET, SOCK_STREAM, 0);
state=setsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void*)&rcv_buf, sizeof(rcv_buf));
state=setsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void*)&snd_buf, sizeof(snd_buf));
len=sizeof(snd_buf);
state=getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void*)&snd_buf, &len);
len=sizeof(rcv_buf);
state=getsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void*)&rcv_buf, &len);
printf("Input buffer size: %d \n", rcv_buf);
printf("Output buffer size: %d \n", snd_buf);
```

실행결과

```
root@my_linux:/tcpip# gcc set_buf.c -o setbuf
root@my_linux:/tcpip# ./setbuf
Input buffer size: 6144
Output buffer size: 6144
```

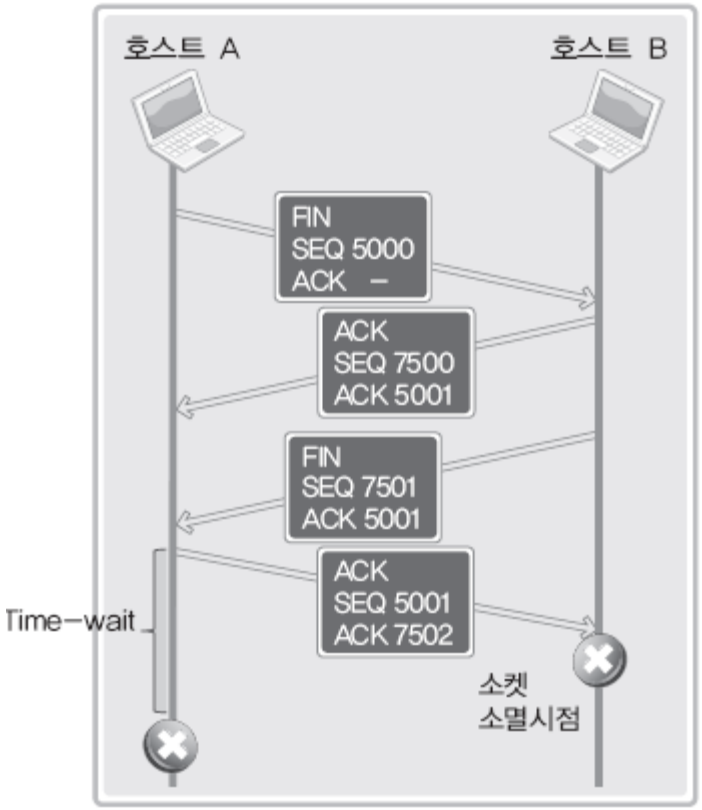
입출력 버퍼는 상당히 주의 깊게 다뤄져야 하는 영역이기 때문에, 실행결과에서 보이듯이 코드에서 요구하는 바가 완벽히 반영되지는 않는다.



Chapter 09-2. SO_REUSEADDR

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

주소할당 에러의 원인 time-wait



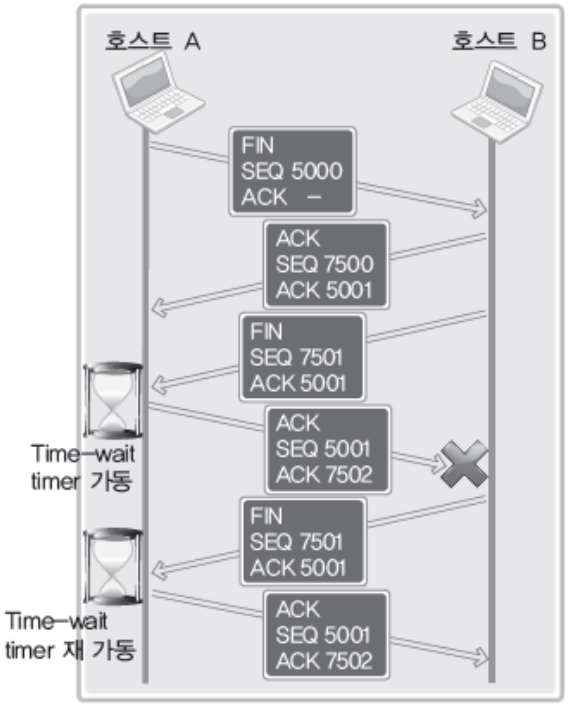
Time-wait의 이해

서버, 클라이언트에 상관없이 TCP 소켓에서 연결의 종료를 목적으로 Four-way handshaking의 첫 번째 메시지를 전달하는 호스트의 소켓은 Time-wait 상태를 거친다. Time-wait 상태 동안에는 해당 소켓이 소멸되지 않아서 할당 받은 Port를 다른 소켓이 할당할 수 없다.

Time-wait의 존재이유

왼쪽 그림에서 호스트 A의 마지막 ACK가 소멸되는 상황을 대비해서 Time-wait 상태가 필요하다. 호스트 A의 마지막 ACK가 소멸되면, 호스트 B는 계속해서 FIN 메시지를 호스트 A에 전달하게 된다.

주소의 재할당

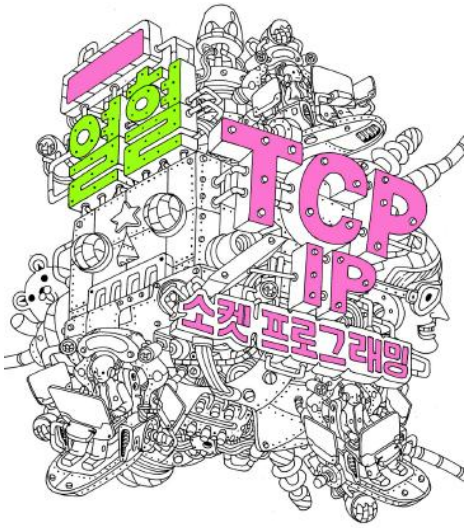


Time-wait은 길어질 수 있다

Time-wait은 필요하다. 그러나 실 서비스중인 서버의 경이 Time-wait이 문제가 될 수 있다. 그러한 경우에는 Time-wait 상태에 있는 Port의 할당이 가능하도록 코드를 수정해야 한다.

Port 할당이 가능하도록 옵션의 변경

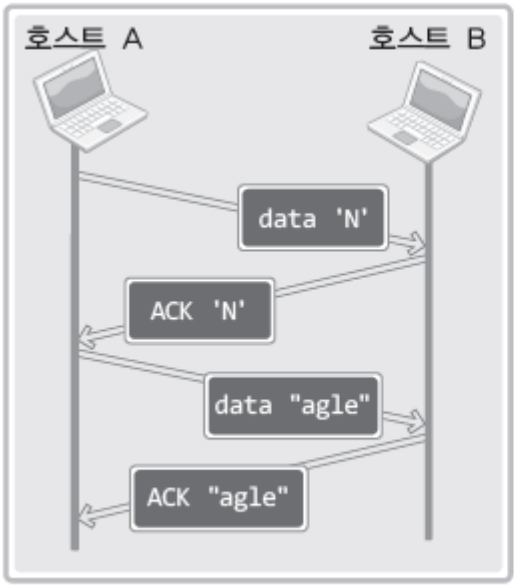
```
optlen=sizeof(option);
option=TRUE;
setsockopt(serv_sock, SOL_SOCKET, SO_REUSEADDR, (void*)&option, optlen);
```



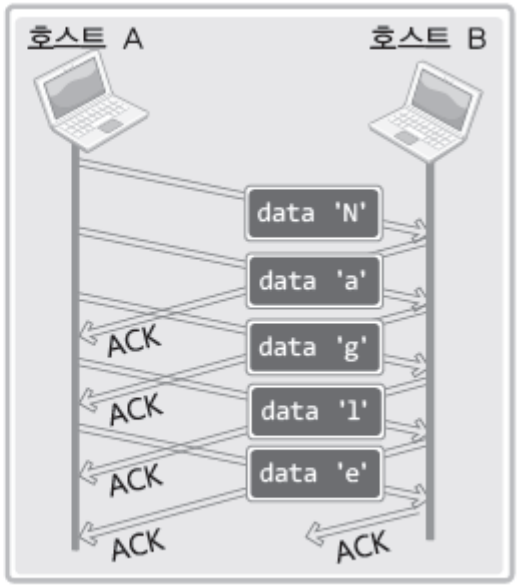
Chapter 09-3. TCP_NODELAY

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

Nagle 알고리즘



Nagle 알고리즘 ON



Nagle 알고리즘 OFF

인터넷의 과도한 트래픽과 그로 인한 전송속도의 저하를 막기 위해서 디자인 된 알고리즘이 Nagle 알고리즘이다.
이러한 Nagle 알고리즘은 그 목적이 명확한 경우가 아니면 중단하지 말아야 하며, 소켓은 기본적으로 Nagle 알고리즘을 적용해서 데이터를 송수신한다.

"Nagle 알고리즘은 앞서 전송한 데이터에 대한 ACK 메시지를 받아야만, 다음 데이터를 전송하는 알고리즘이다!"

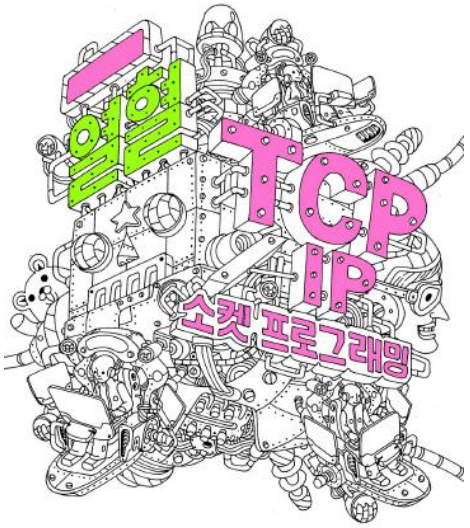
Nagle 알고리즘의 중단

Nagle 알고리즘의 중단을 명령하는 코드

```
int opt_val=1;
setsockopt(sock, IPPROTO_TCP, TCP_NODELAY, (void*)&opt_val, sizeof(opt_val));
```

Nagle 알고리즘의 설정상태 확인하는 코드

```
int opt_val;
socklen_t opt_len;
opt_len=sizeof(opt_val);
getsockopt(sock, IPPROTO_TCP, TCP_NODELAY, (void*)&opt_val, &opt_len);
```



Chapter 09-4. 윈도우 기반으로 구현하기

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

소켓의 옵션정보를 확인하는 getsockopt 함수

```
#include <winsock2.h>

int getsockopt(SOCKET sock, int level, int optname, char * optval, int * optlen);
```

➡ 성공 시 0, 실패 시 SOCKET_ERROR 반환

- sock — 옵션확인을 위한 소켓의 핸들 전달.
- level — 확인할 옵션의 프로토콜 레벨 전달.
- optname — 확인할 옵션의 이름 전달.
- optval — 확인결과의 저장을 위한 버퍼의 주소 값 전달.
- optlen — 네 번째 매개변수 optval로 전달된 주소 값의 버퍼 크기를 담고 있는 변수의 주소 값 전달, 함수호출이 완료되면 이 변수에는 네 번째 인자를 통해 반환된 옵션정보의 크기가 바이트 단위로 계산되어 저장된다.

리눅스의 getsockopt 함수와 다르지 않다.

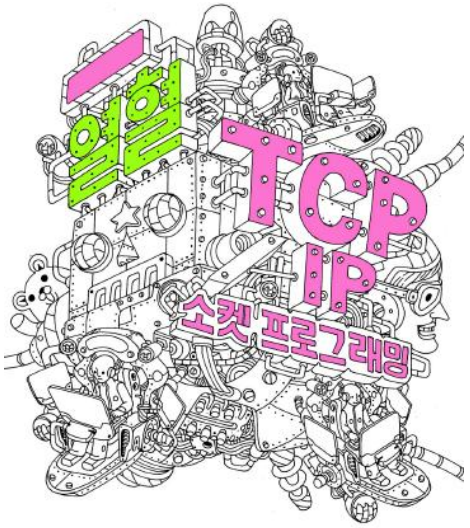
소켓의 옵션정보를 확인하는 setsockopt 함수

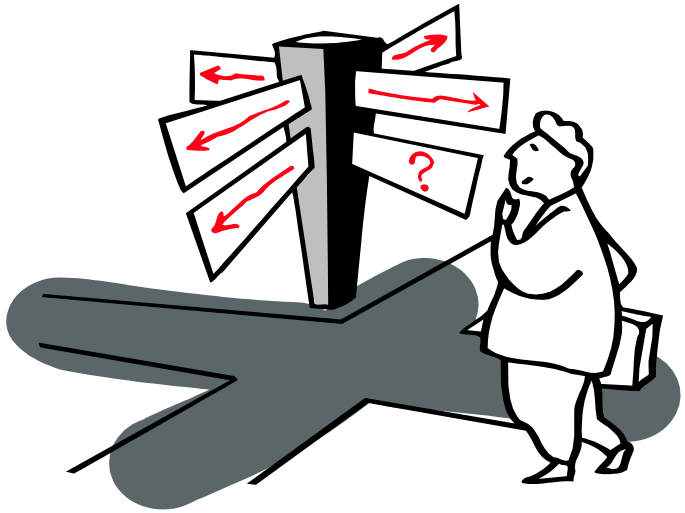
```
#include <winsock2.h>

int setsockopt(SOCKET sock, int level, int optname, const char* optval, int optlen);
```

➜ 성공 시 0, 실패 시 SOCKET_ERROR 반환

- sock 옵션변경을 위한 소켓의 핸들 전달.
- level 변경할 옵션의 프로토콜 레벨 전달.
- optname 변경할 옵션의 이름 전달.
- optval 변경할 옵션정보를 저장한 버퍼의 주소 값 전달.
- optlen 네 번째 매개변수 optval로 전달된 옵션정보의 바이트 단위 크기 전달.

리눅스의 setsockopt 함수와 다르지 않다.





Chapter 09이 끝났습니다. 질문 있으신지요?