

Global KPU!

글로벌 경쟁력을 갖춘 산업기술 명문대학
세계를 향해 더 큰 미래를 펼쳐갑니다

1

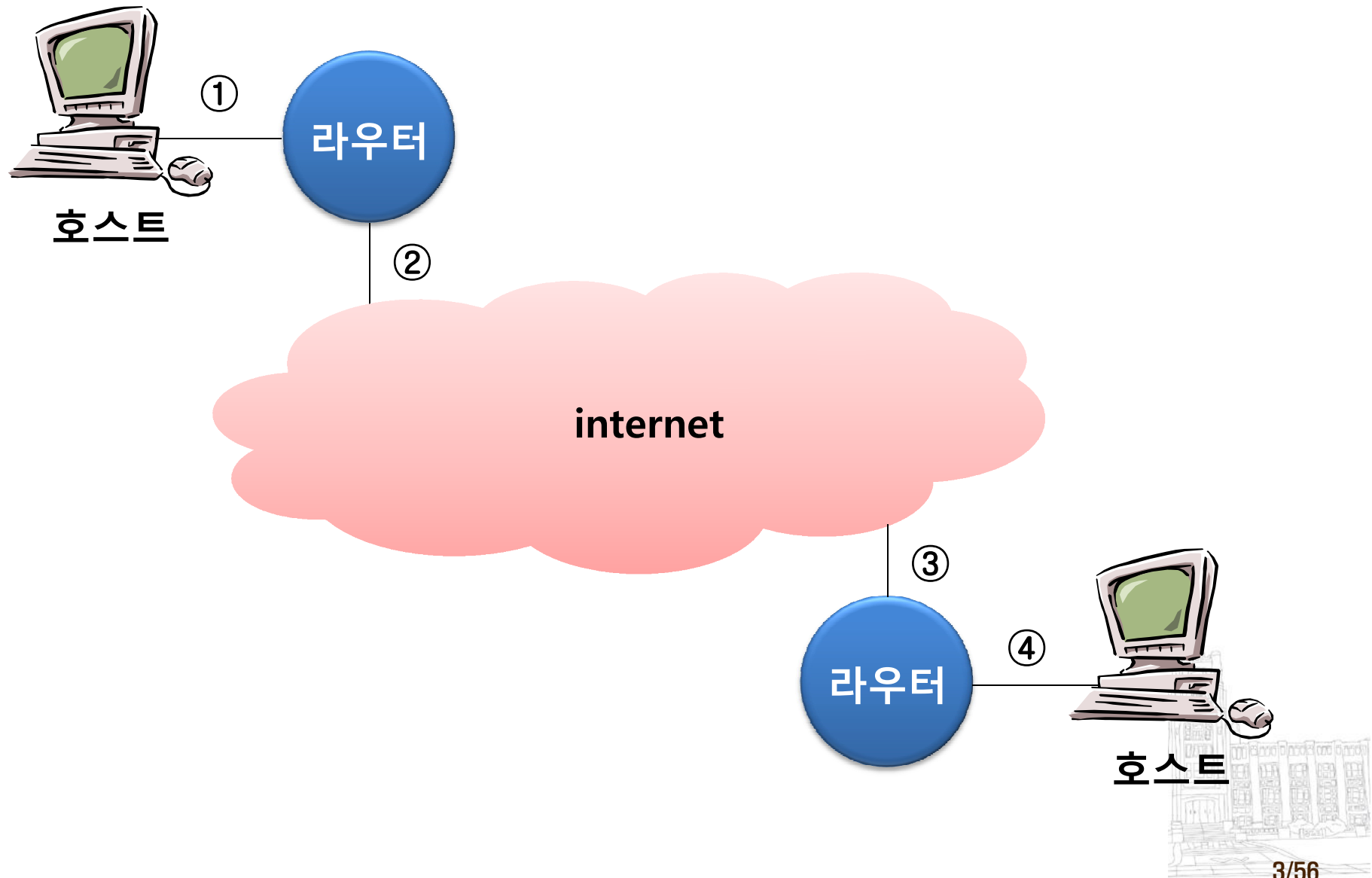
네트워크와 소켓 프로그래밍

네트워크 게임 프로그래밍

- ❖ TCP/IP 프로토콜의 동작 원리를 살펴본다.
- ❖ 소켓의 기본 개념을 이해한다.
- ❖ 윈도우 소켓의 역사와 특징을 살펴본다.
- ❖ 윈도우 소켓 프로그램을 작성하고 실행하는 과정을 이해한다.



인터넷 구성 요소 (1)



인터넷 구성 요소 (2)

❖ 호스트

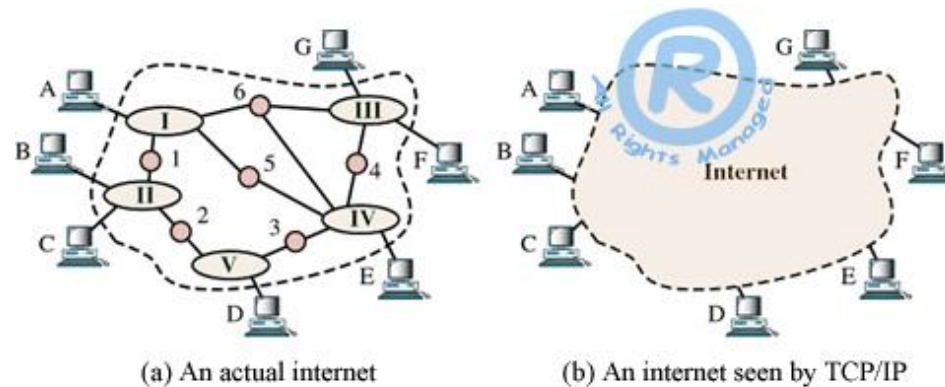
- 최종 사용자의 응용 프로그램을 수행하는 주체

❖ 라우터

- 호스트에서 생성된 데이터를 여러 네트워크를 거쳐 전송함으로써 서로 다른 네트워크에 속한 호스트 간에 데이터를 교환할 수 있게 하는 장비

❖ 통신 프로토콜

- 호스트와 라우터, 라우터와 라우터, 호스트와 호스트가 통신하기 위한 정해진 절차와 방법



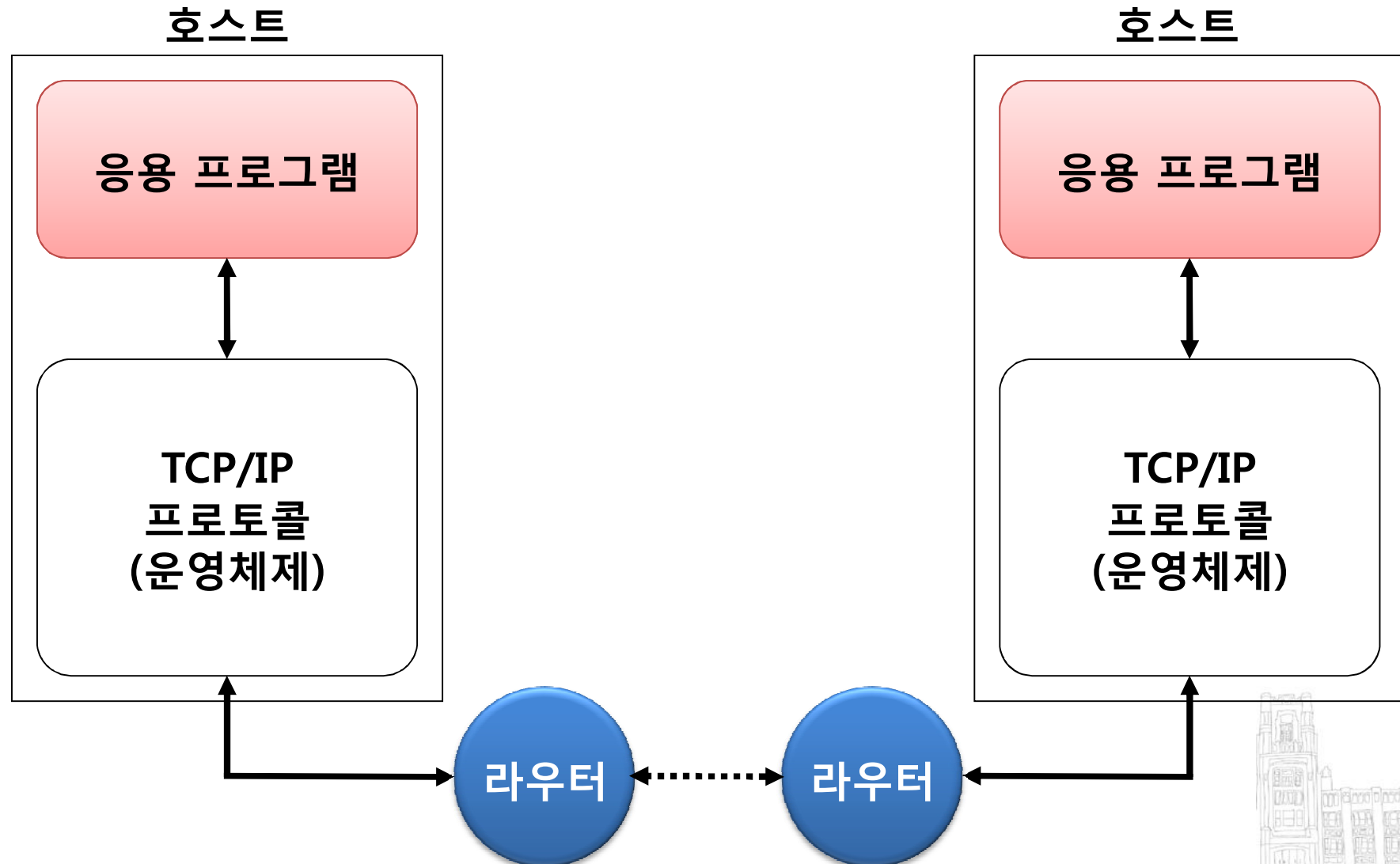
TCP/IP 프로토콜 (1)

❖ TCP/IP 프로토콜

- 인터넷의 핵심 프로토콜인 TCP와 IP를 비롯한 각종 프로토콜
- 운영체제의 일부로 구현되며, 응용 프로그램은 운영체제가 제공하는 TCP/IP 프로토콜의 서비스를 사용하여 통신



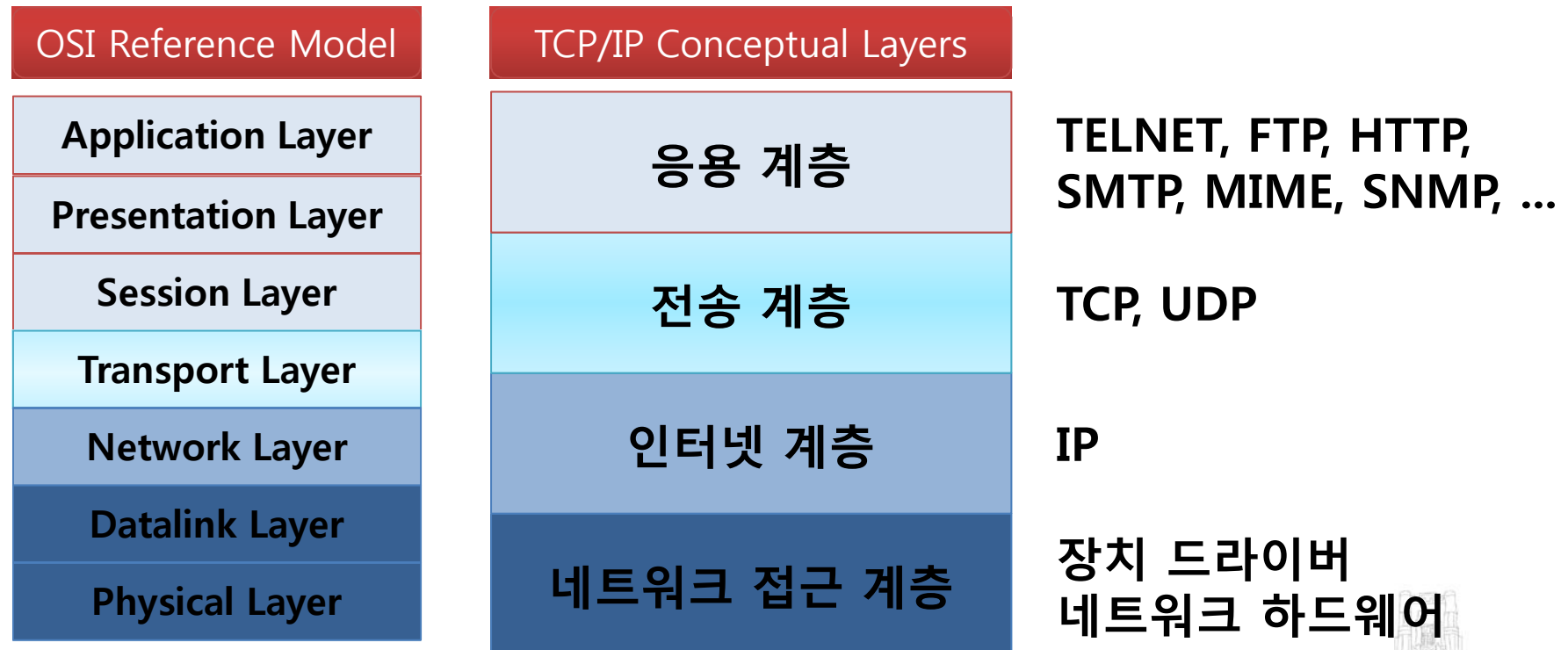
TCP/IP 프로토콜 (2)



TCP/IP 프로토콜 구조 (1)

❖ TCP/IP 프로토콜 구조

■ 계층적 구조



TCP/IP 프로토콜 구조 (2)

❖ 네트워크 접근 계층 (데이터링크 계층 / 물리 계층)

■ 역할

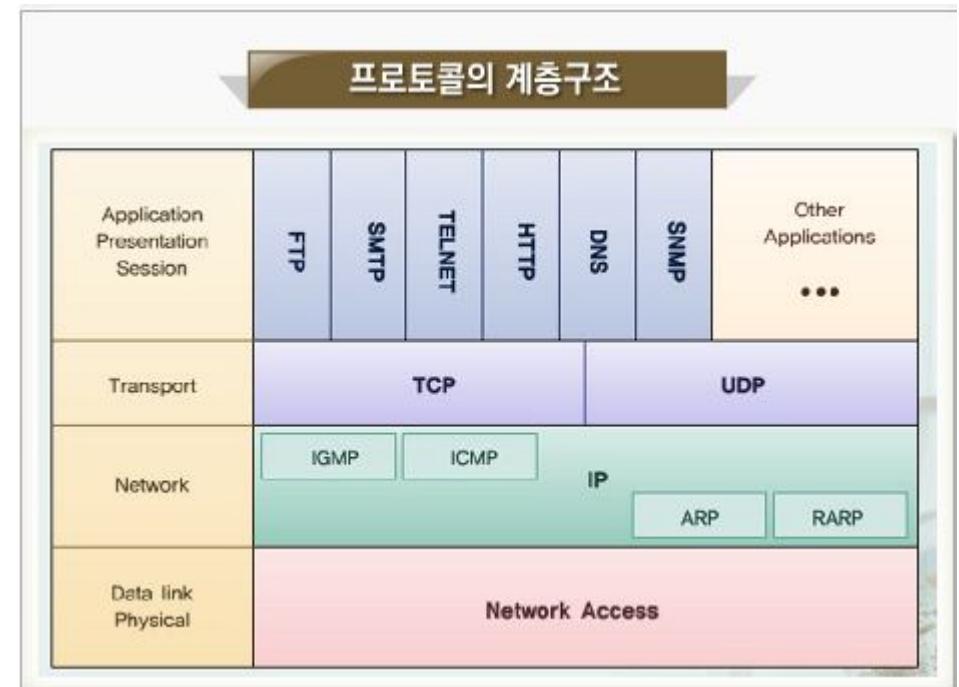
- 물리적 네트워크를 통한 데이터 송수신
- 데이터를 전송하는 케이블에 프레임을 송수신

■ 구성 요소

- 네트워크 하드웨어 + 장치 드라이버

■ 주소 지정 방식

- 물리 주소
 - 예) 이더넷: 48비트 물리 주소
 - 예) 무선 LAN



TCP/IP 프로토콜 구조 (3)

❖ 인터넷 계층 (네트워크 계층)

■ 역할

- 네트워크 접근 계층의 도움을 받아 데이터를 목적지 호스트까지 전달
- 송신지에서 목적지가 있는 네트워크로 데이터를 전송하는 역할
- IP의 주소를 관리하고 패킷의 IP보고 라우팅하는 역할

■ 구성 요소

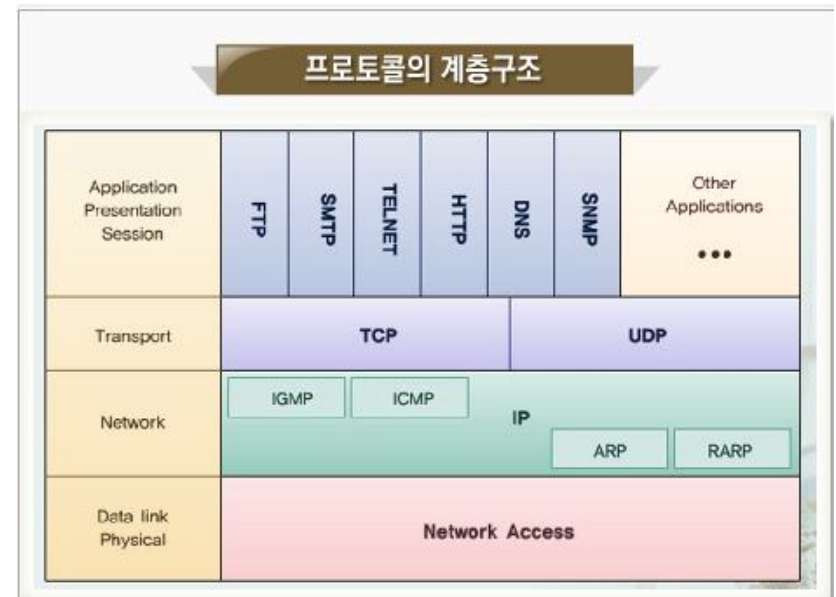
- IP 주소 + 라우팅(or 라우터)

■ 주소 지정 방식

- IP 주소
 - 소프트웨어적으로 정의된 논리 주소
 - 전 세계적인 유일성과 하드웨어 독립성을
 - 신뢰성이 없고 비연결형인 프로토콜
 - 오류 검사나 추적을 하지 않음

■ 라우팅

- 데이터를 목적지까지 전달하는 일련의 작업
 - 라우팅에 필요한 정보 수집
 - 라우팅 정보를 기초로 데이터 전달



TCP/IP 프로토콜 구조 (4)

❖ 전송 계층 (전송 계층)

■ 역할

- 최종 통신 목적지(응용 프로그램)를 지정하고, 오류 없이 데이터를 전송
 - 데이터 손실 또는 손상을 검출해 잘못된 데이터가 목적지에 전달되는 일을 방지
- 신뢰성 있는 데이터전송을 위해 흐름제어, 오류제어 기능
- 응용프로그램 사이의 종단 간 신뢰성 제공

■ 주소 지정 방식

- 포트 번호

■ 대표 프로토콜

- TCP :
 - Handshaking 절차 수행, 안전하고 순서있는 데이터 전송과 서로간의 흐름제어를 수행
 - 수신측으로부터 확인 응답을 요구할 필요가 있을때 신뢰성 있는 전송을 위해서 사용
- UDP
 - Handshaking 절차 미수행, 데이터 전송의 안전성을 보장하지 못하고, TCP 보다 속도가 빠르며 전송량은 작음.
 - 패킷의 정확한 전달을 보장하지 않고 송수신의 책임은 상위의 어플리케이션이 가지도록 함



TCP/IP 프로토콜 구조 (5)

❖ TCP와 UDP

TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
연결형(connection-oriented) 프로토콜 - 연결 설정 후 통신 가능	비연결형(connectionless) 프로토콜 - 연결 설정 없이 통신 가능
신뢰성 있는 데이터 전송 - 데이터를 재전송함	신뢰성 없는 데이터 전송 - 데이터를 재전송하지 않음
일대일 통신(unicast)	일대일 통신(unicast), 일대다 통신(broadcast, multicast)
패킷을 관리할 필요가 없음 UDP보다 전송속도가 느림	패킷을 관리해주어야 함 TCP보다 전송속도가 빠름
데이터 경계 구분 안 함 - 바이트 스트림(byte-stream) 서비스	데이터 경계 구분함 - 데이터그램(datagram) 서비스
관련 클래스 Socket ServerSocket	관련 클래스 DatagramSocket DatagramPacket MulticastSocket



TCP/IP 프로토콜 구조 (6)

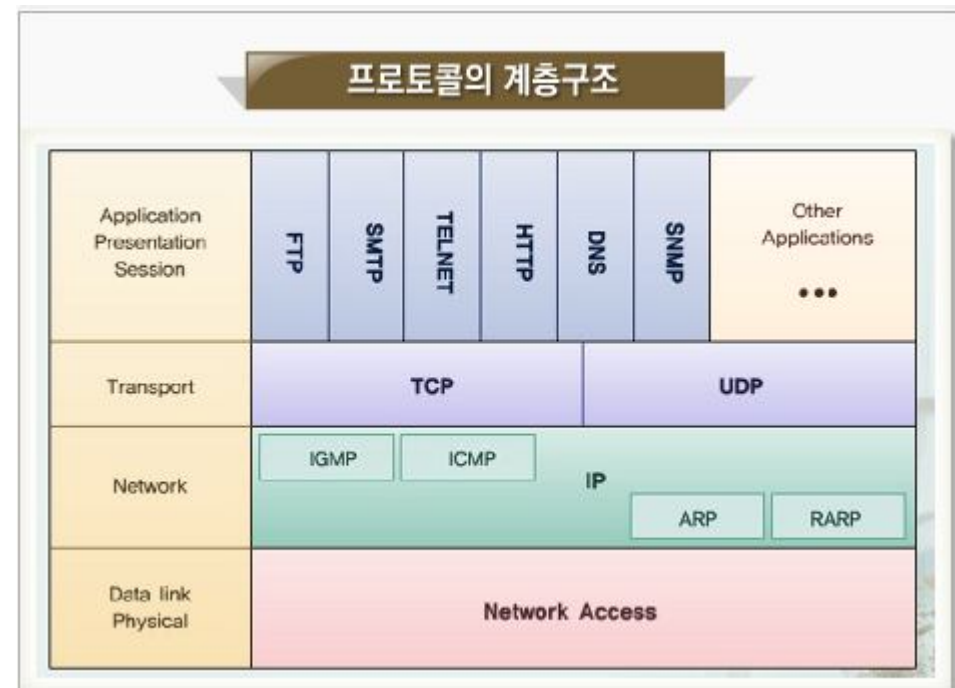
❖ 응용 계층

■ 역할

- 전송 계층을 기반으로 한 다수의 프로토콜과 이 프로토콜을 사용하는 응용 프로그램을 포괄
- 어플리케이션이 네트워크에 접근 가능하도록 해주는 역할 담당

■ 대표 프로토콜

- Telnet, FTP, HTTP, SMTP, ...



❖ 패킷(Packet)이란?

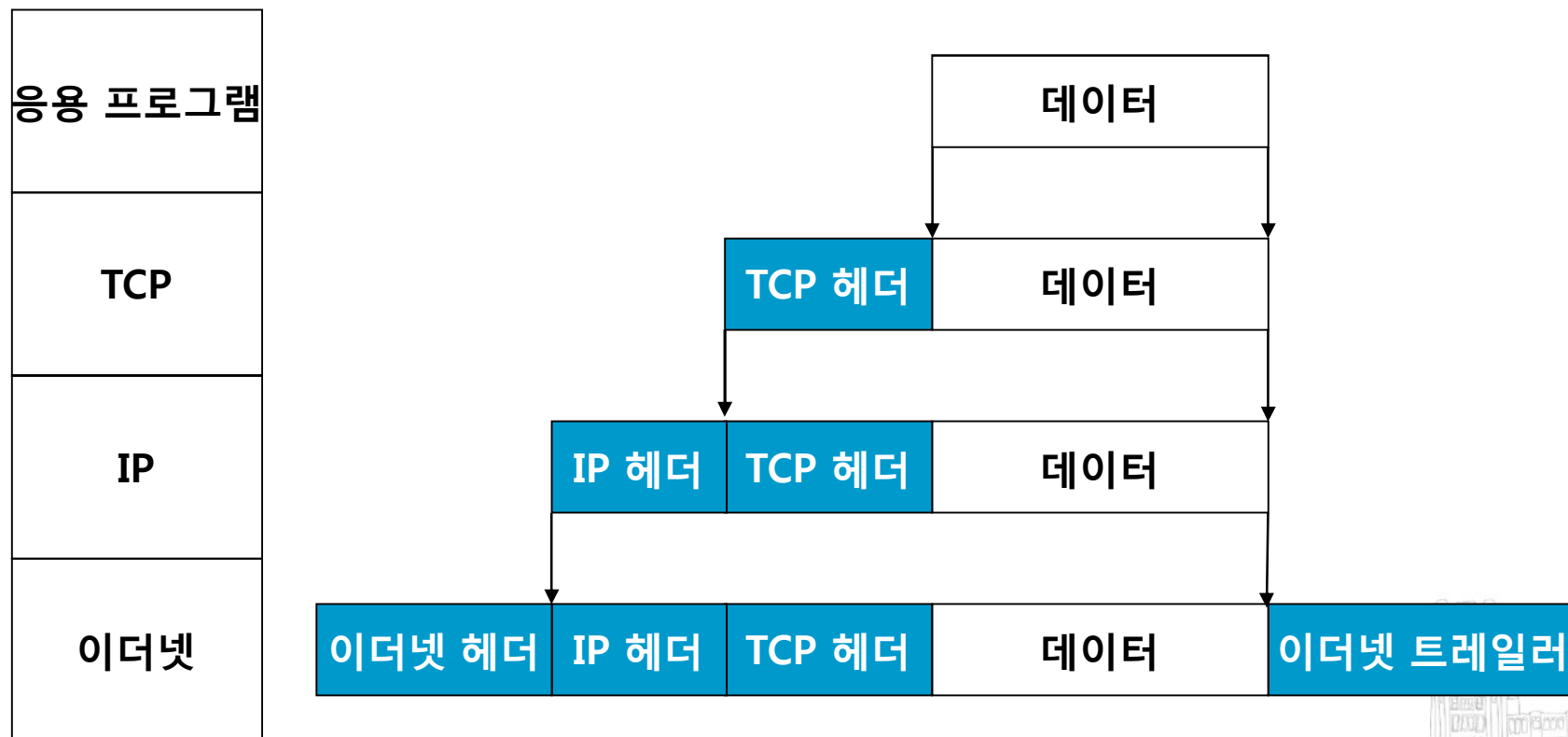
- 각 프로토콜에서 정의한 제어 정보(IP 주소, 포트 번호, 오류 체크 코드 등) + 데이터
- 제어 정보의 위치에 따라 앞쪽에 붙는 헤더(*header*)와 뒤쪽에 붙는 트레일러(*trailer*)로 구분
- 우체국은 라우터 택배는 패킷



데이터 전송 원리 (2)

❖ 패킷 전송 형태

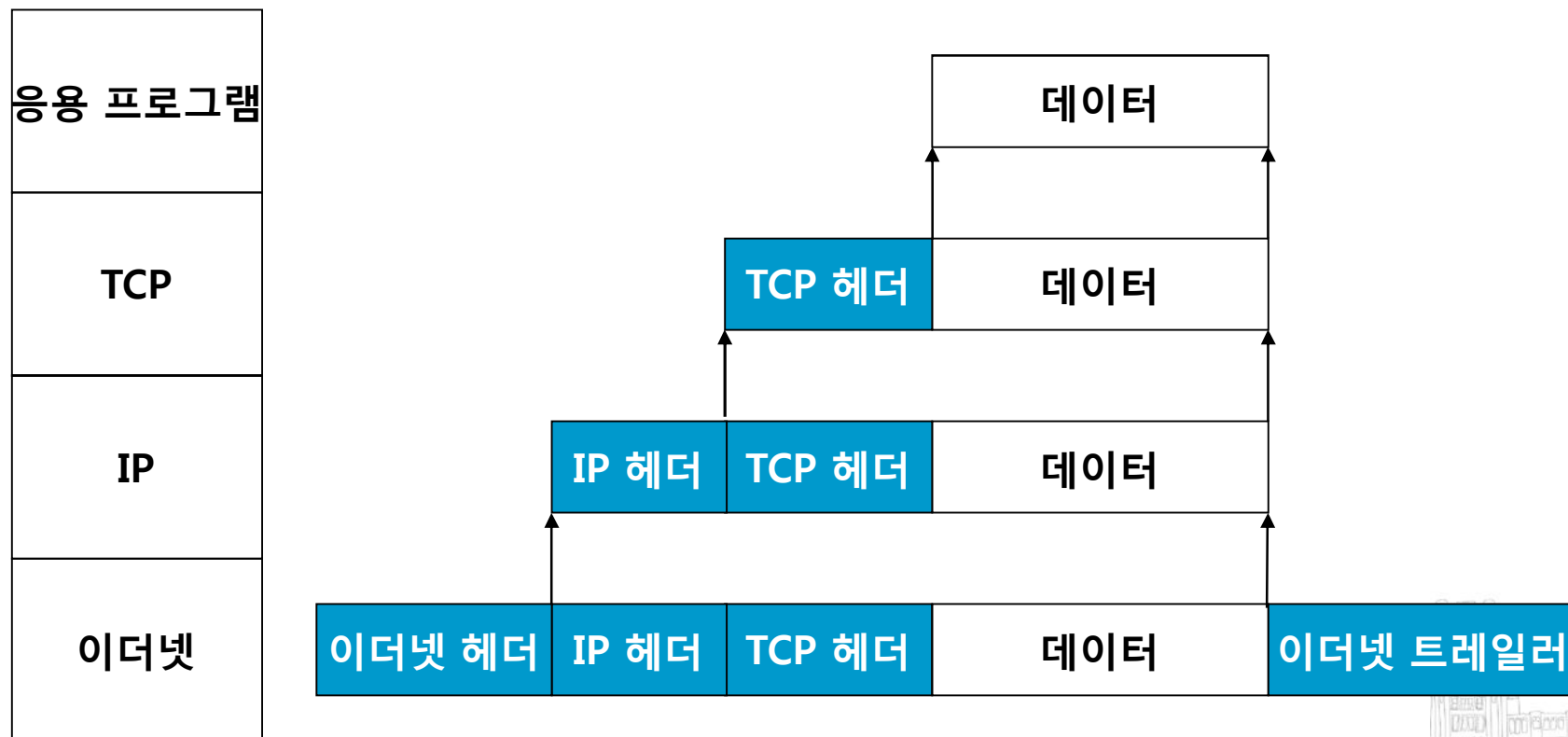
■ 송신측



데이터 전송 원리 (3)

❖ 패킷 전송 형태

■ 수신측

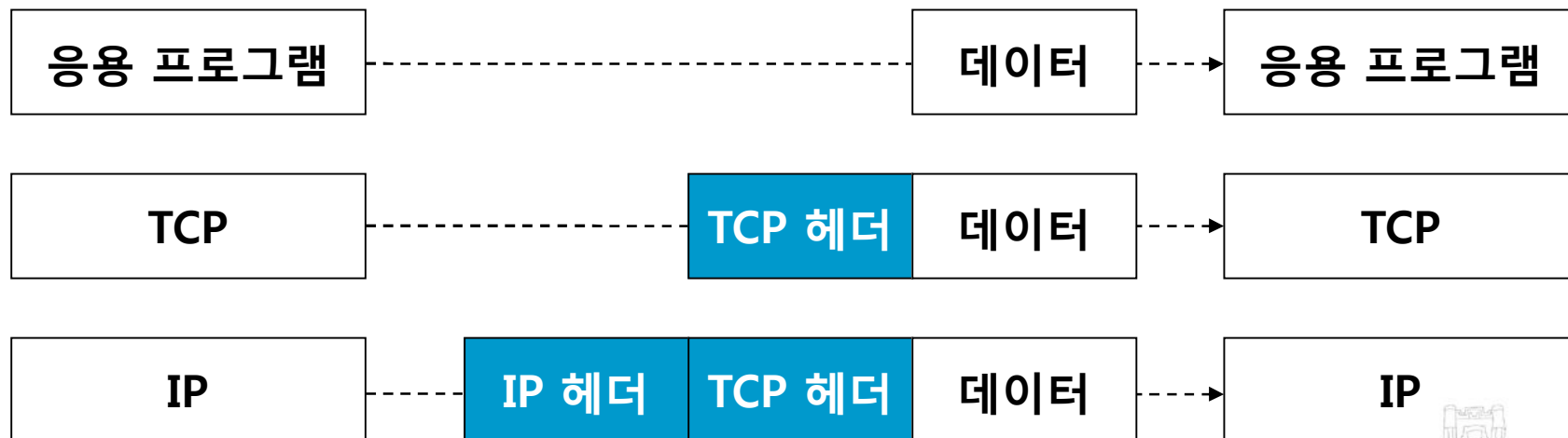


데이터 전송 원리 (4)

❖ 패킷 전송 형태

■ 계층별

- 각 계층은 동일 위치의 상대 계층과 통신하는 것으로 간주

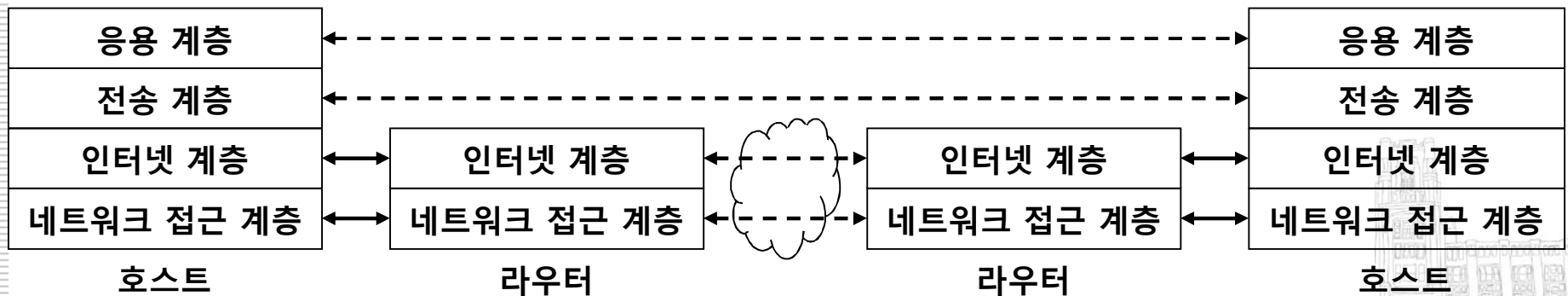


데이터 전송 원리 (5)



❖ 패킷 전송 형태

■ 인터넷

- 응용 계층, 전송 계층
 - 하부 계층이 제공하는 가상적인 연결을 사용해 동작
- 인터넷 계층
 - IP 주소와 라우팅 기능을 이용해 패킷 전송 경로 결정
- 네트워크 접근 계층
 - 물리 주소를 사용해 실제 패킷 전송



❖ IP 주소

- 인터넷에 있는 호스트와 라우터의 식별자
 - 폐쇄된 네트워크거나 IP를 공유하는 경우가 아니면 전 세계적으로 값이 유일
- IPv4는 32비트, IPv6는 128비트 사용
- IPv4는 8비트 단위로 .(dot)로 구분하여 10진수 4개로 표기 
dotted-decimal notation
 - 예) 147.46.114.70
- IPv6는 16비트 단위로 :(colon)으로 구분하여 16진수 8개로 표기 
colon-hexadecimal notation
 - 예) 2001:0230:abcd:ffab:0023:eb00:ffff:1111



❖ 포트 번호

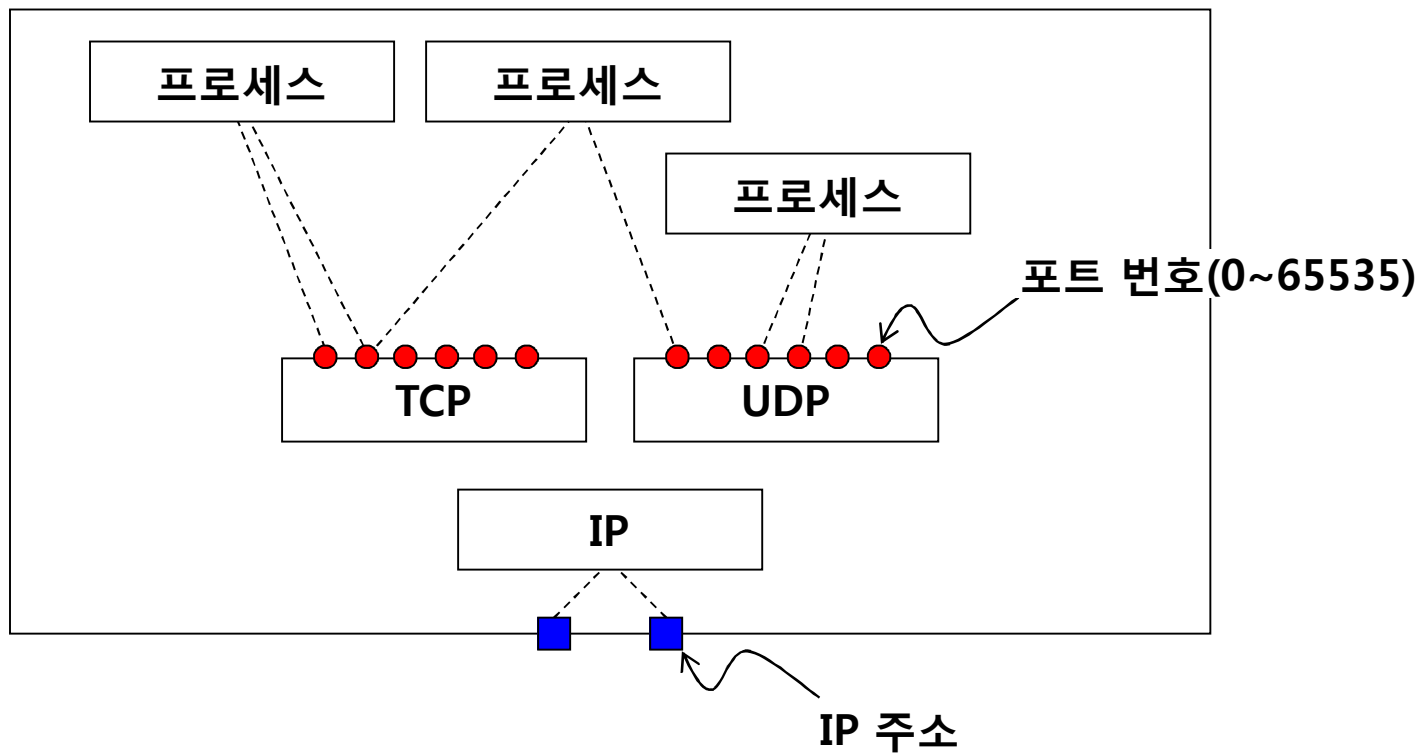
- 인터넷 통신의 종착점(하나 혹은 여러 프로세스)을 나타내는 식별자
- TCP와 UDP는 포트 번호로 부호 없는 16비트 정수를 사용하므로 0~65535 범위가 가능
- 영역별 포트 번호

포트 번호	분류
0 ~ 1023	알려진 포트(well-known ports)
1024 ~ 49151	등록된 포트(registered ports)
49152 ~ 65535	동적/사설 포트(dynamic and/or private ports)



IP 주소, 포트 번호 (3)

❖ IP 주소와 포트 번호



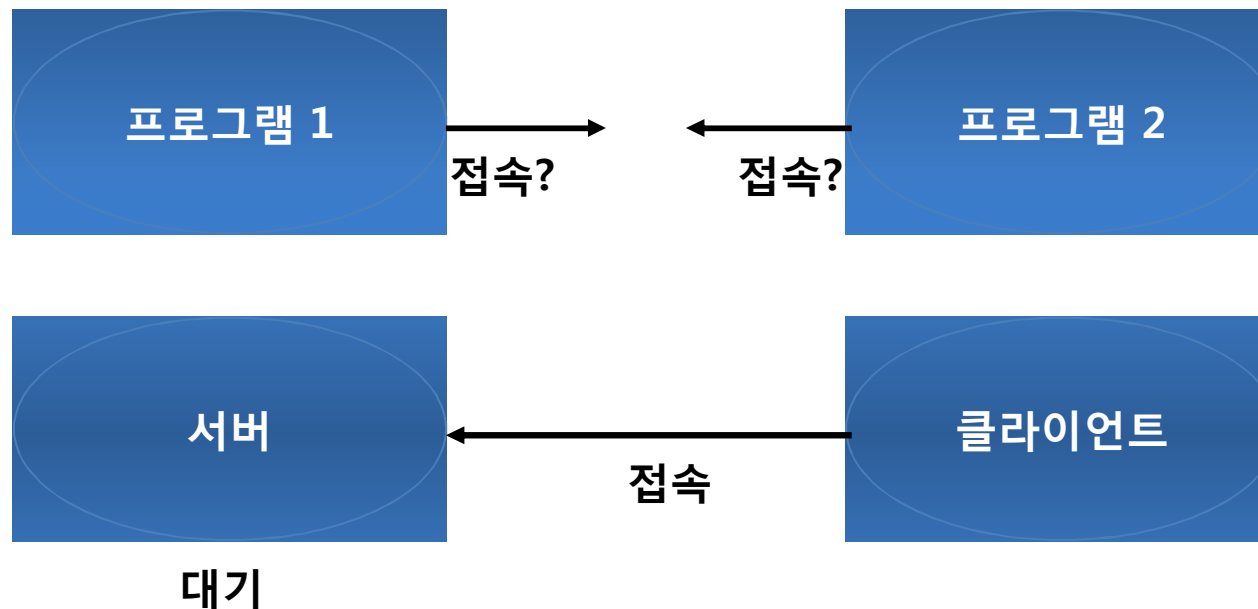
❖ 도메인 이름

- IP 주소에 대한 (기억하기 쉬운) 별명
- 실제 통신할 때는 IP 주소로 변환해야 함



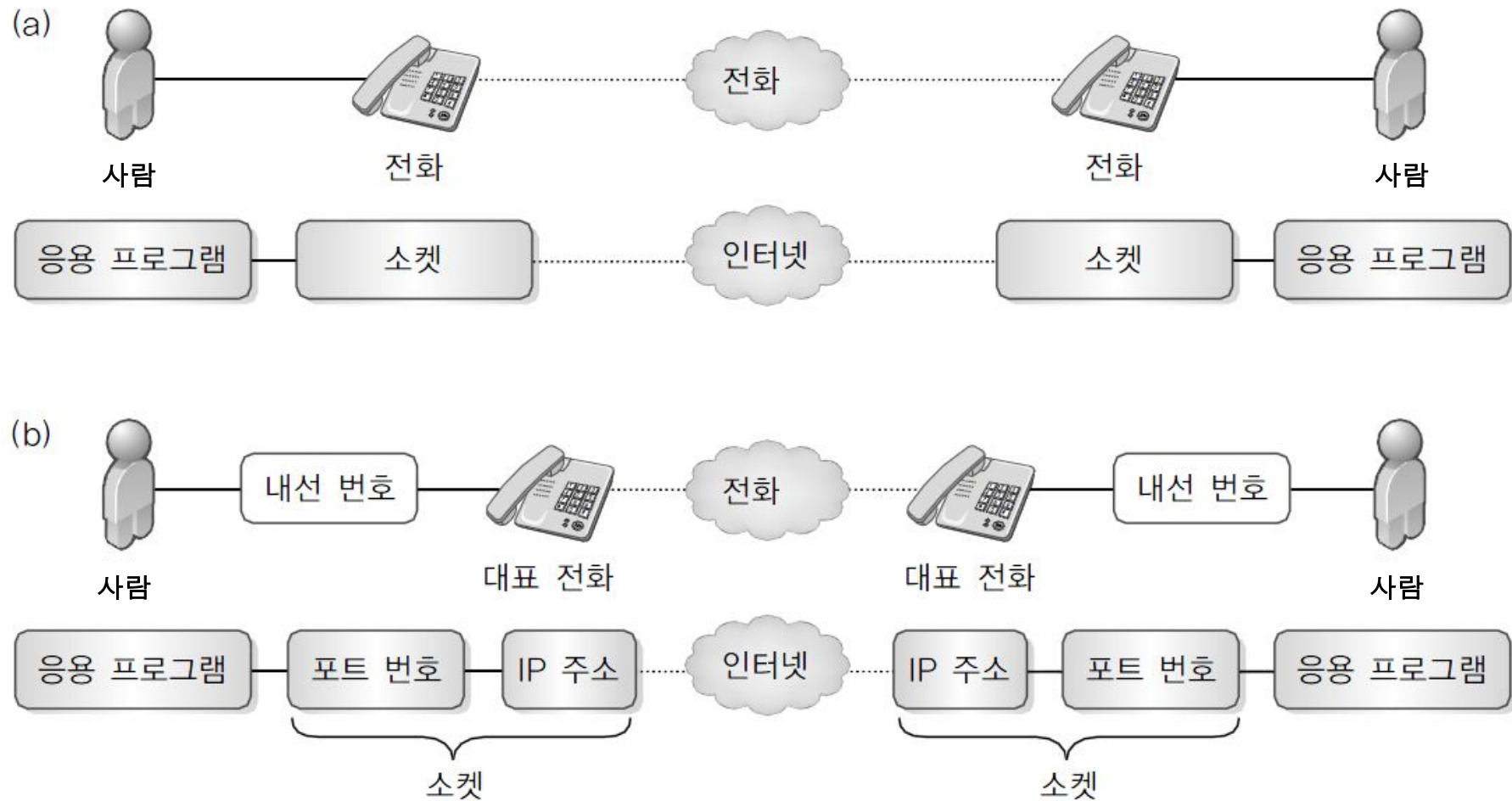
❖ 클라이언트-서버(client/server) 모델

- 두 프로그램이 상호 작용하는 방식을 나타내는 용어
- 서비스를 요청하는 쪽은 클라이언트(client), 클라이언트가 요청하는 서비스를 처리하는 쪽은 서버(server)



소켓의 개념 (1)

❖ 전화 통신과 소켓 통신 비교



소켓의 개념 (2)

❖ 세 가지 관점

- ① 데이터 타입
- ② 통신 종단점
- ③ 네트워크 프로그래밍 인터페이스



소켓의 개념 (3)

❖ 데이터 타입

- 파일 디스크립터 혹은 핸들과 유사한 개념
- 생성과 설정 과정이 끝나면 운영체제의 통신 관련 정보를 참조해 다양한 작업을 편리하게 할 수 있는 데이터 타입

```
// 파일 생성  
int fd = open("myfile", ...);  
...  
read(fd, ...) // 데이터 읽기  
write(fd, ...) // 데이터 쓰기
```



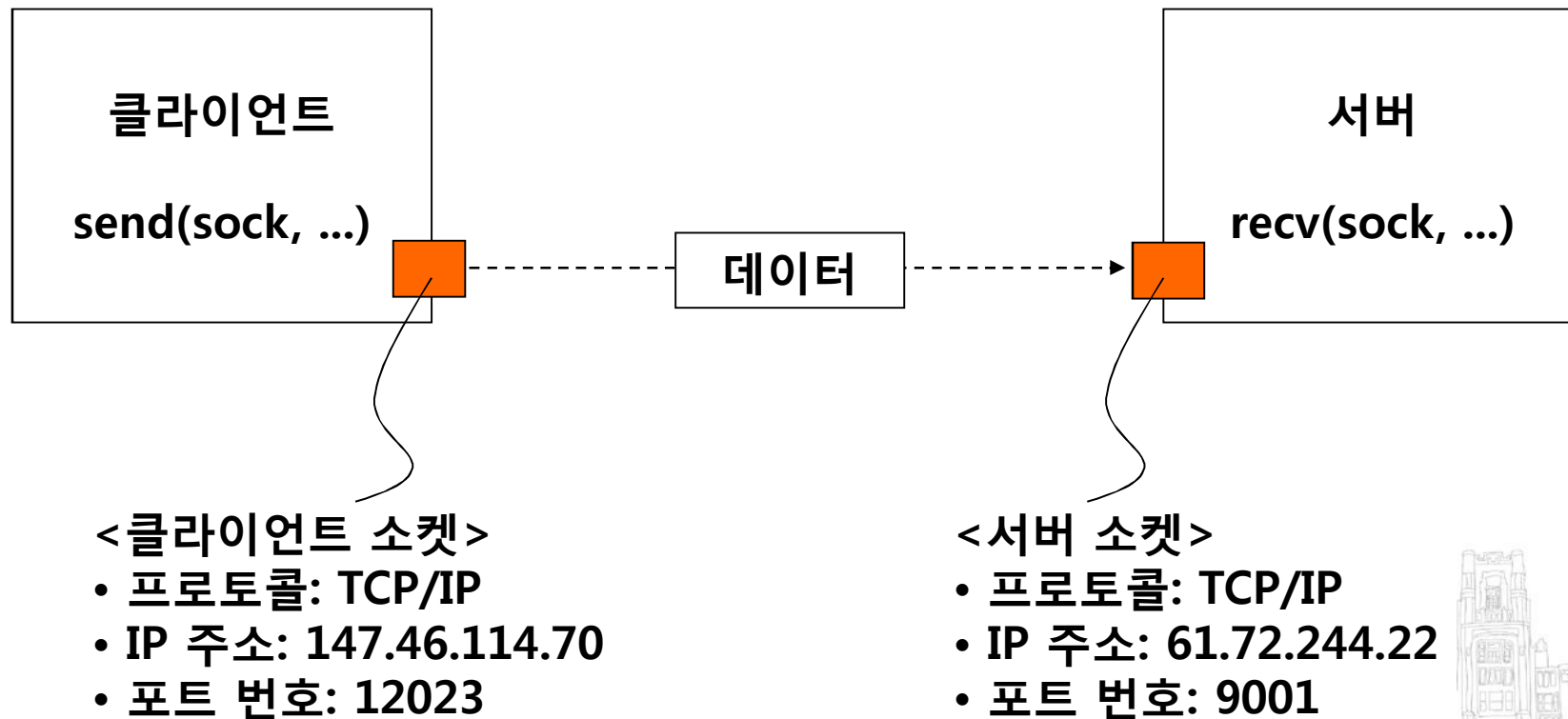
```
// 소켓 생성  
SOCKET sock = socket(...);  
...  
recv(sock, ...) // 데이터 받기  
send(sock, ...) // 데이터 보내기
```



소켓의 개념 (4)

❖ 통신 종단점

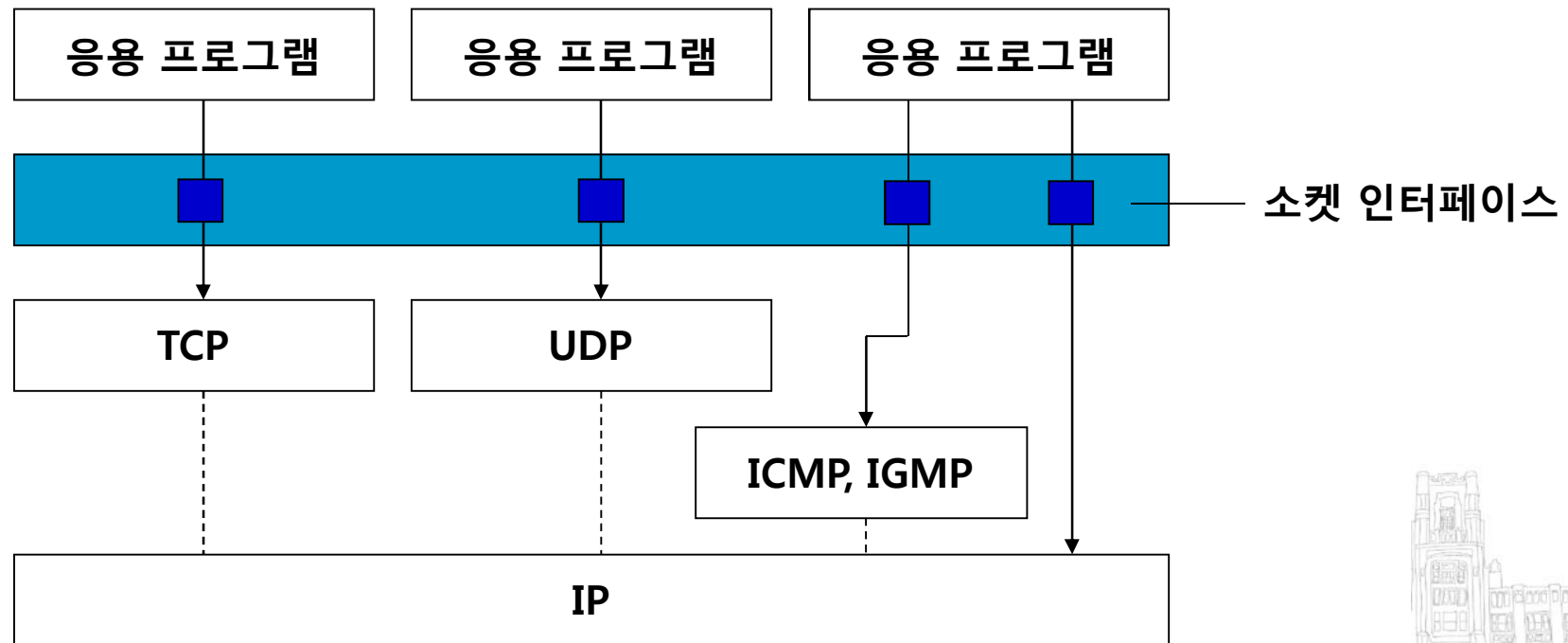
- 응용 프로그램은 자신의 소켓이 상대방의 소켓과 연결된 것으로 생각하고 데이터를 주고받음



소켓의 개념 (5)

❖ 네트워크 프로그래밍 인터페이스

- 통신 양단이 모두 소켓을 사용할 필요는 없음
- TCP/IP 프로토콜에서 (일반적으로) 응용 계층과 전송 계층 사이에 위치하는 것으로 간주



❖ 윈도우 소켓(원속)

- 버클리 유닉스에서 개발한 네트워크 프로그래밍 인터페이스를 윈도우 환경에서 사용할 수 있게 만든 것
- 윈도우 95 버전부터 API에 정식으로 포함하여 제공



❖ 윈도우 소켓과 유닉스 소켓의 차이점

- 윈도우 소켓은 DLL을 통해 대부분의 기능이 제공되므로 DLL 초기화와 종료 작업을 위한 함수가 필요
- 윈도우 프로그램은 대개 GUI를 갖추고 메시지 구동 방식으로 동작하므로 이를 위한 확장 함수가 존재
- 윈도우는 운영체제 차원에서 멀티스레드를 지원하므로 멀티스레드 환경에서 안정적으로 동작하는 구조와 이를 위한 함수가 필요



윈도우 소켓 (3)

❖ 윈도우 운영체제의 윈속 지원

운영체제	윈속 버전
윈도우 95	1.1 (2.2)
윈도우 98/Me, 윈도우 NT/2000/XP/2003서버, 윈도우 비스타/2008 서버/7/8/2010서버	2.2
윈도우 CE	1.1 (2.2)

❖ 윈속에서 지원하는 통신 프로토콜

- TCP/IP(윈도우 95 이상, 윈도우 CE 2.1 이상)
- IPv6(윈도우 XP SP1 이상, 윈도우 CE .NET 4.1 이상)
- IrDA(윈도우 98 이상, 모든 윈도우 CE 버전)
- Bluetooth(윈도우 XP SP2 이상, 윈도우 CE .NET 4.0 이상)



❖ 윈속의 장점

- 유닉스 소켓과 소스 코드 수준에서 호환성이 높으므로 기존 코드를 이식하여 활용하기 쉬움
- 가장 널리 사용하는 네트워크 프로그래밍 인터페이스이므로 한번 배우면 여러 운영체제 (윈도우, 리눅스 등)에서 사용 가능
- TCP/IP 외의 프로토콜도 지원하므로 최소 코드 수정으로 응용 프로그램이 사용할 프로토콜 변경 가능
- 비교적 저수준 프로그래밍 인터페이스이므로 세부 제어가 가능하며 고성능 네트워크 프로그램 개발 가능



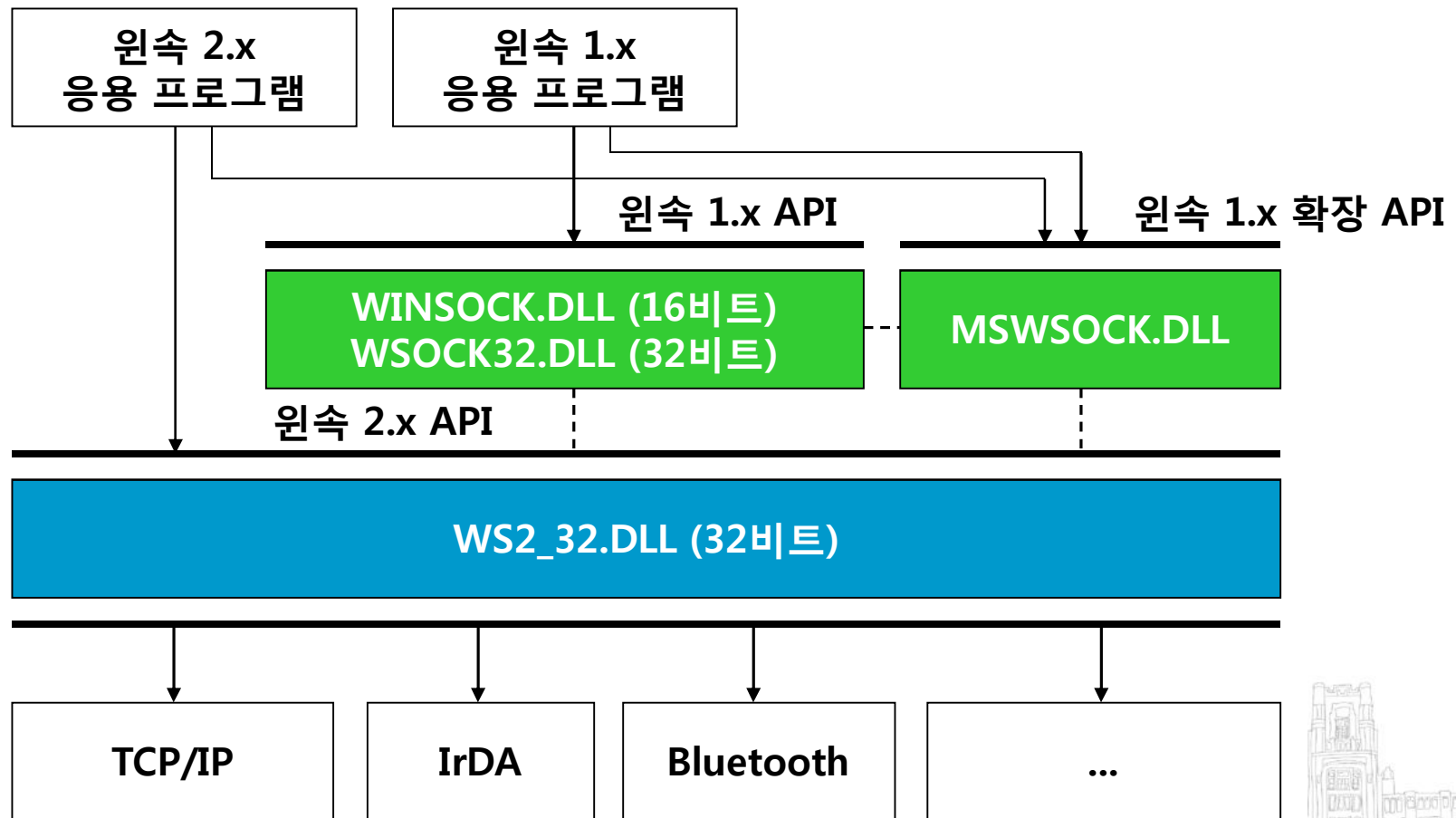
❖ 윈속의 단점

- 응용 프로그램 수준의 프로토콜을 프로그래머가 직접 설계해야 함
 - 주고받는 데이터 형식이나 전송 절차 등을 고려해 프로그래밍해야 하며, 설계 변경 시에는 코드 수정이 불가피함
- 서로 다른 바이트 정렬 방식을 사용하거나 데이터 처리 단위가 서로 다른 호스트끼리 통신할 경우, 응용 프로그램 수준에서 데이터 변환을 처리해야 함



윈도우 소켓 (6)

❖ 윈속 구조

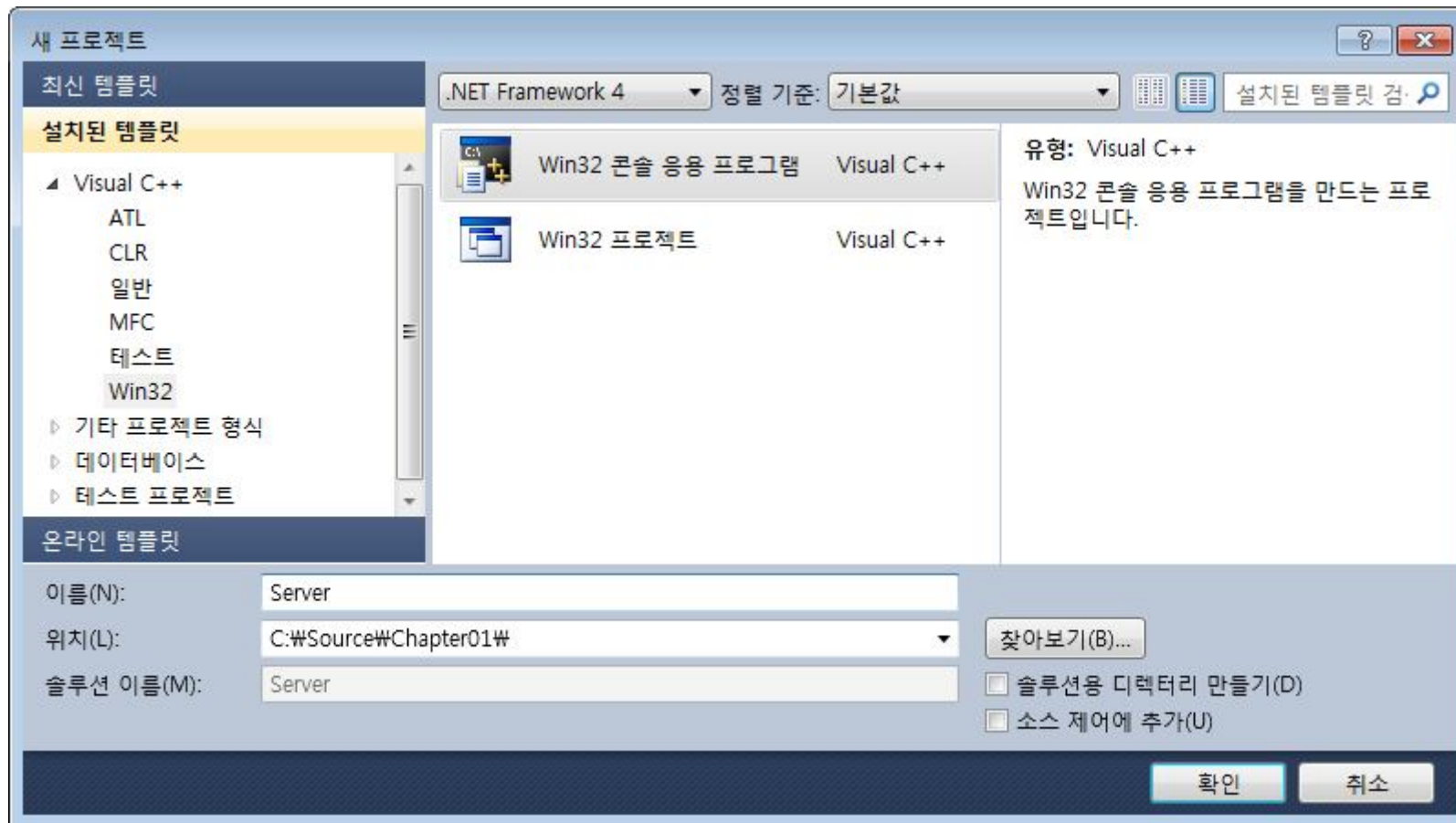


실습 1-1 p39



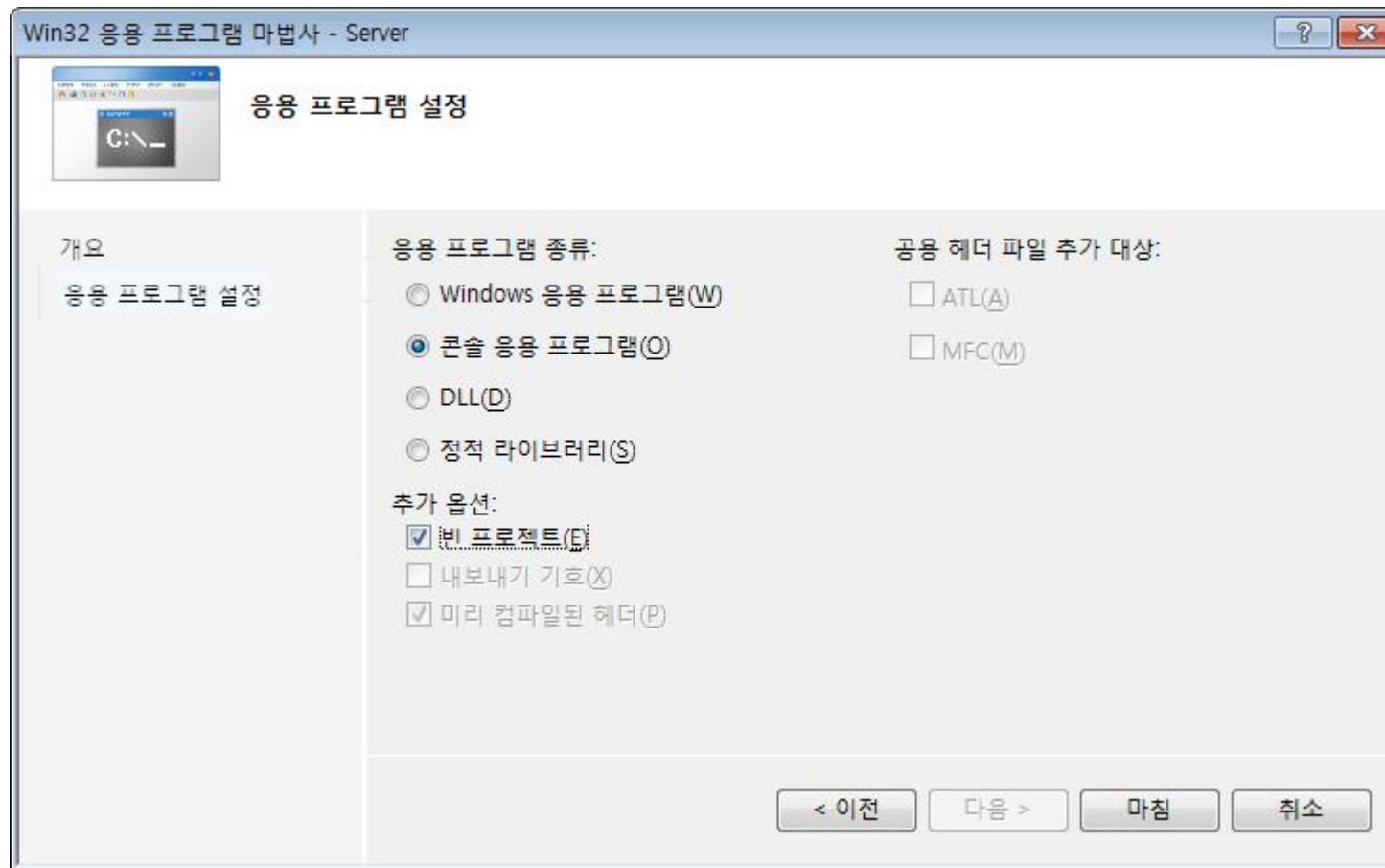
윈도우 소켓 프로그램 맛보기 (1)

❖ 프로젝트 생성



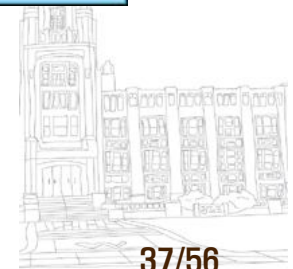
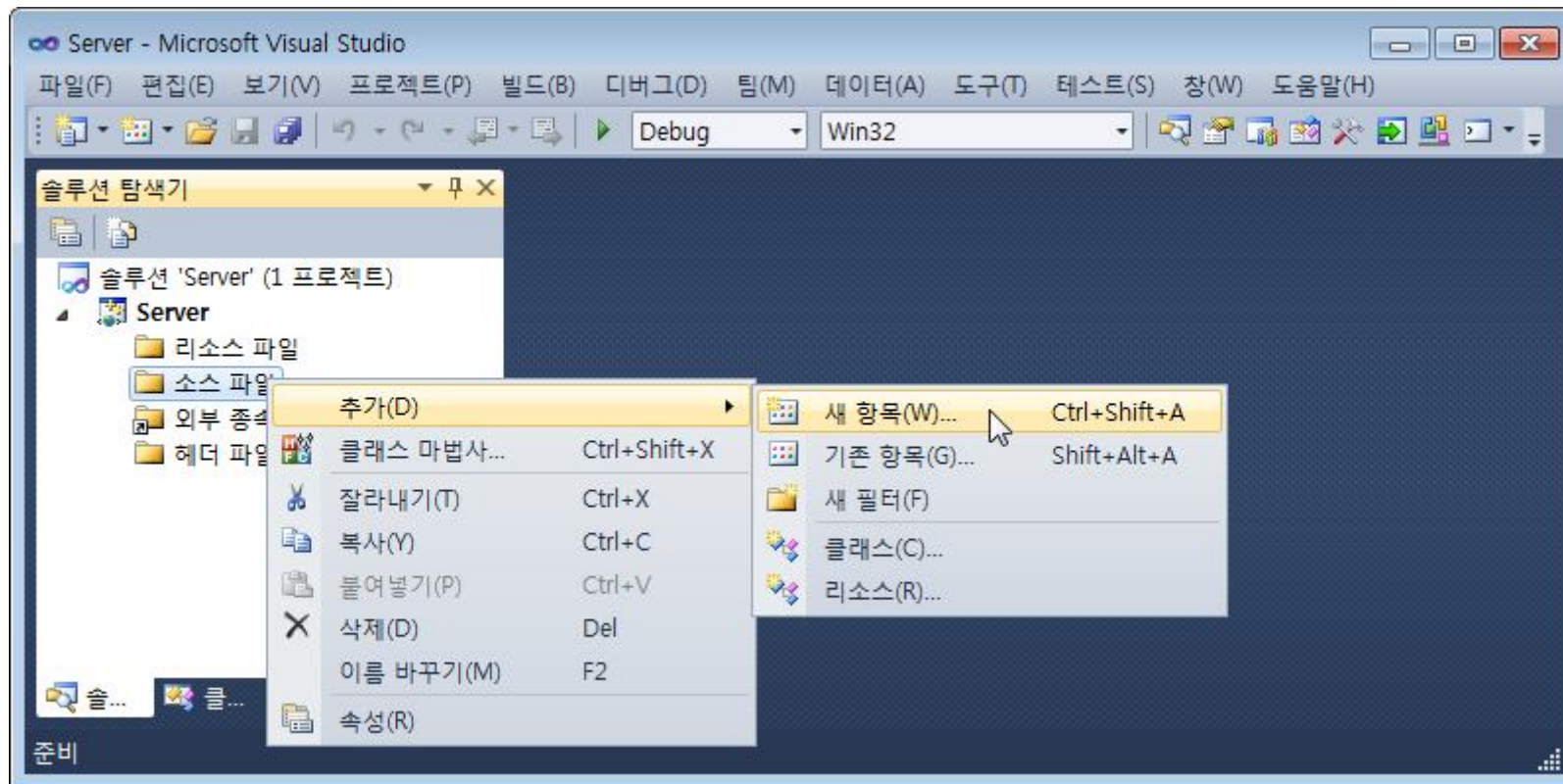
윈도우 소켓 프로그램 맛보기 (2)

❖ 설정 변경



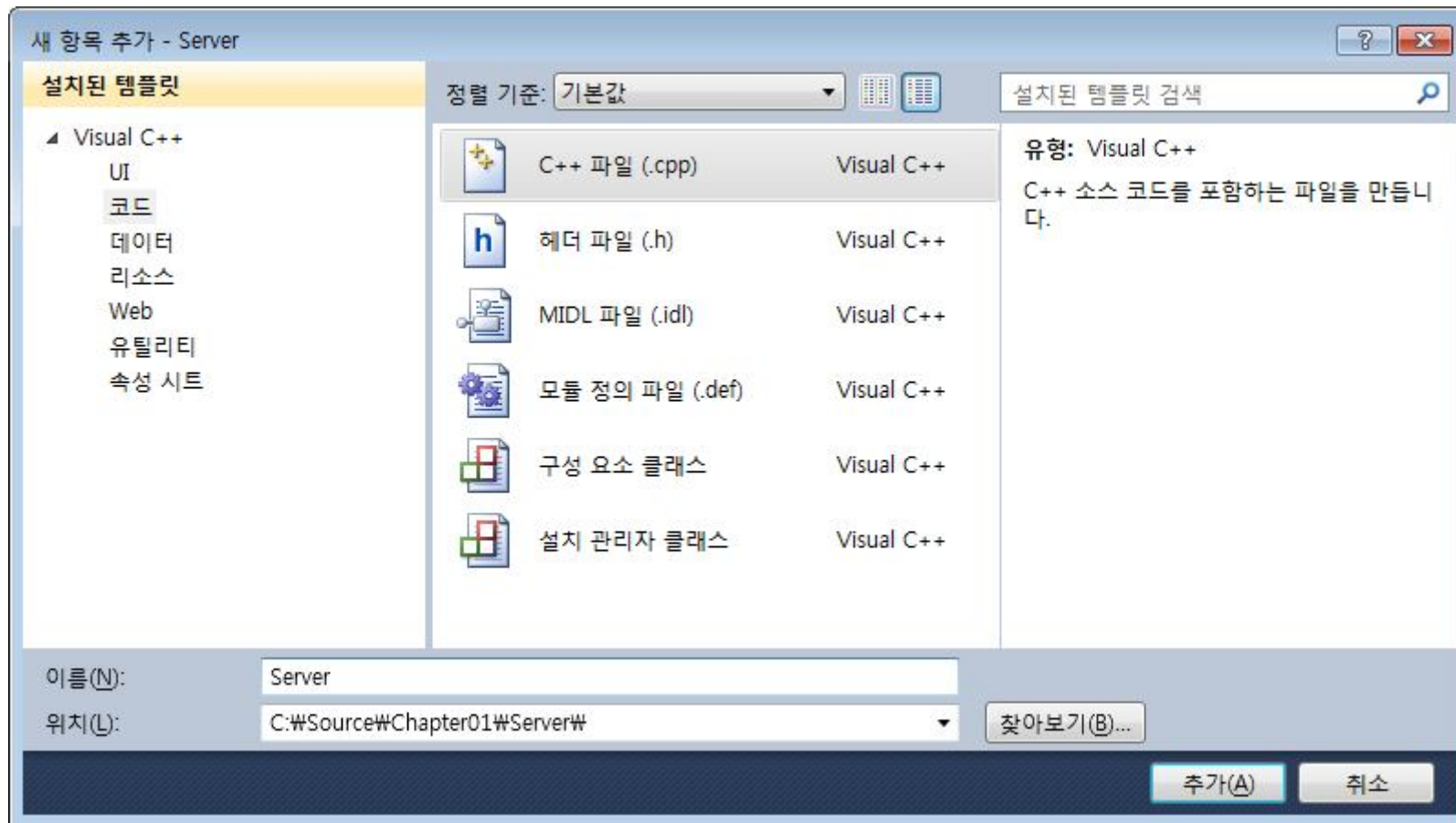
윈도우 소켓 프로그램 맛보기 (3)

❖ 소스 파일 추가 (1/2)



윈도우 소켓 프로그램 맛보기 (4)

❖ 소스 파일 추가 (2/2)



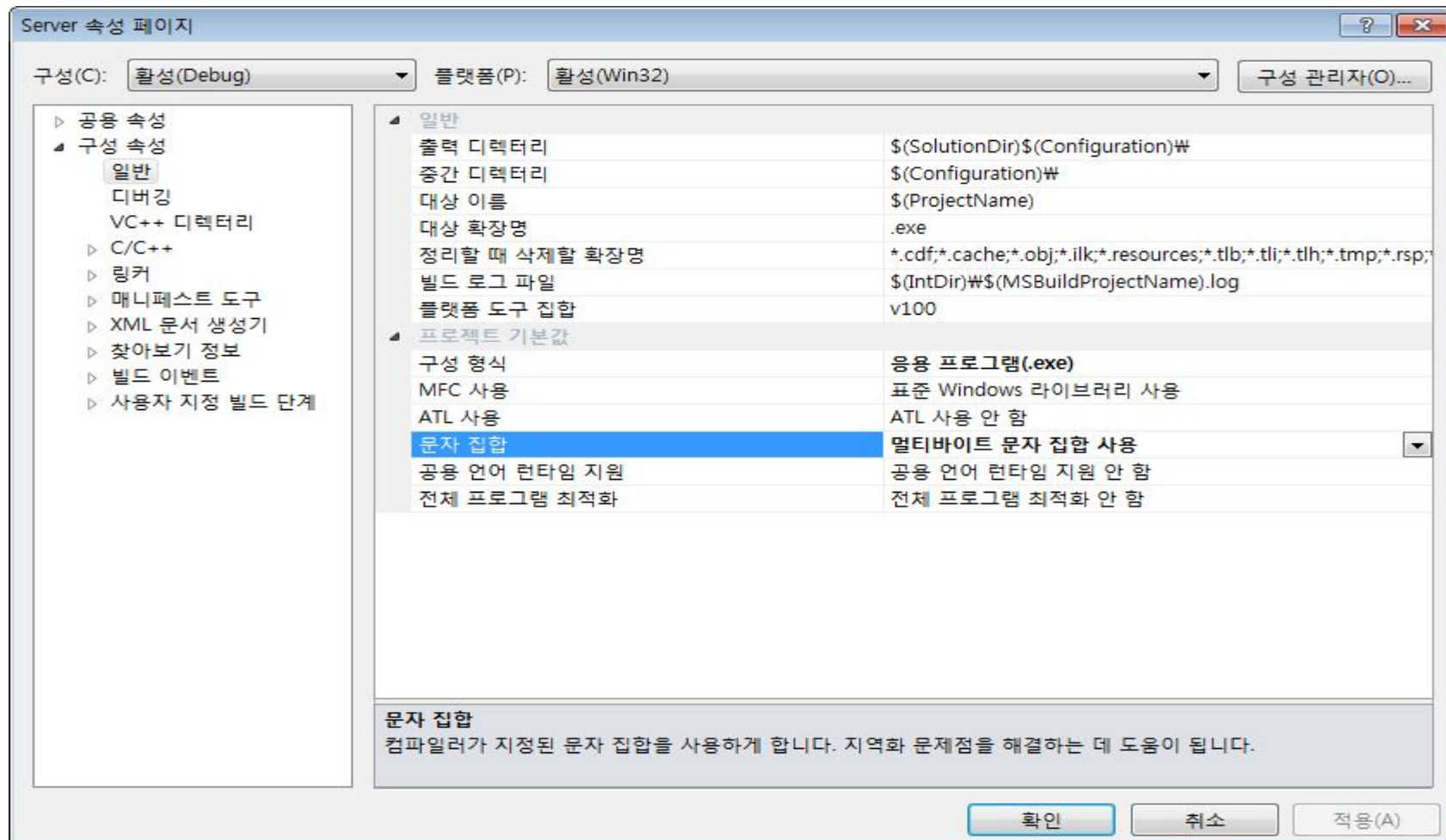
윈도우 소켓 프로그램 맛보기 (5)

❖ 문자 집합 변경과 윈속 라이브러리 추가 (1/4)



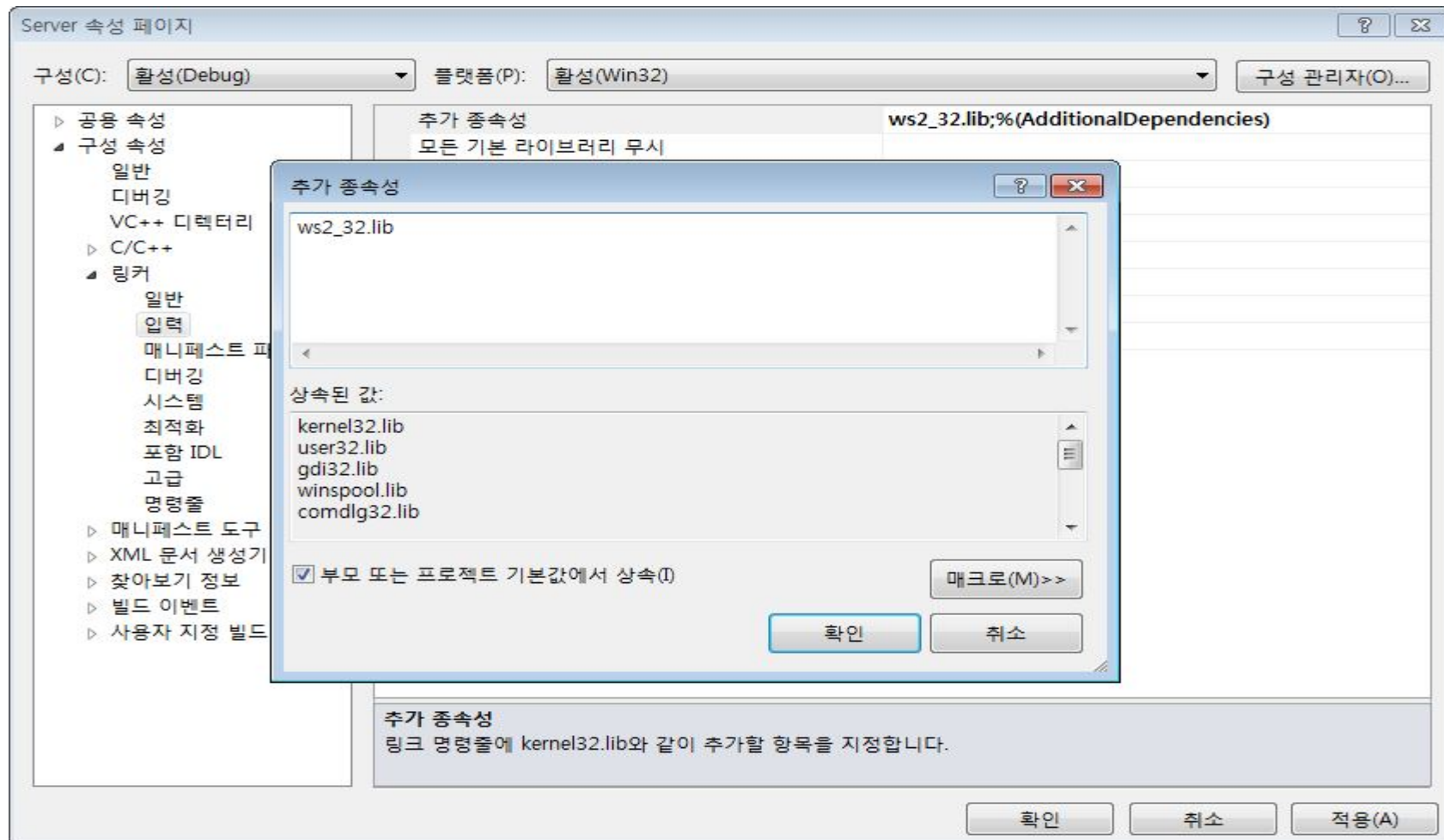
윈도우 소켓 프로그램 맛보기 (6)

❖ 문자 집합 변경과 윈속 라이브러리 추가 (2/4)



윈도우 소켓 프로그램 맛보기 (7)

❖ 문자 집합 변경과 윈속 라이브러리 추가 (3/4)



❖ 문자 집합 변경과 원속 라이브러리 추가 (4/4)

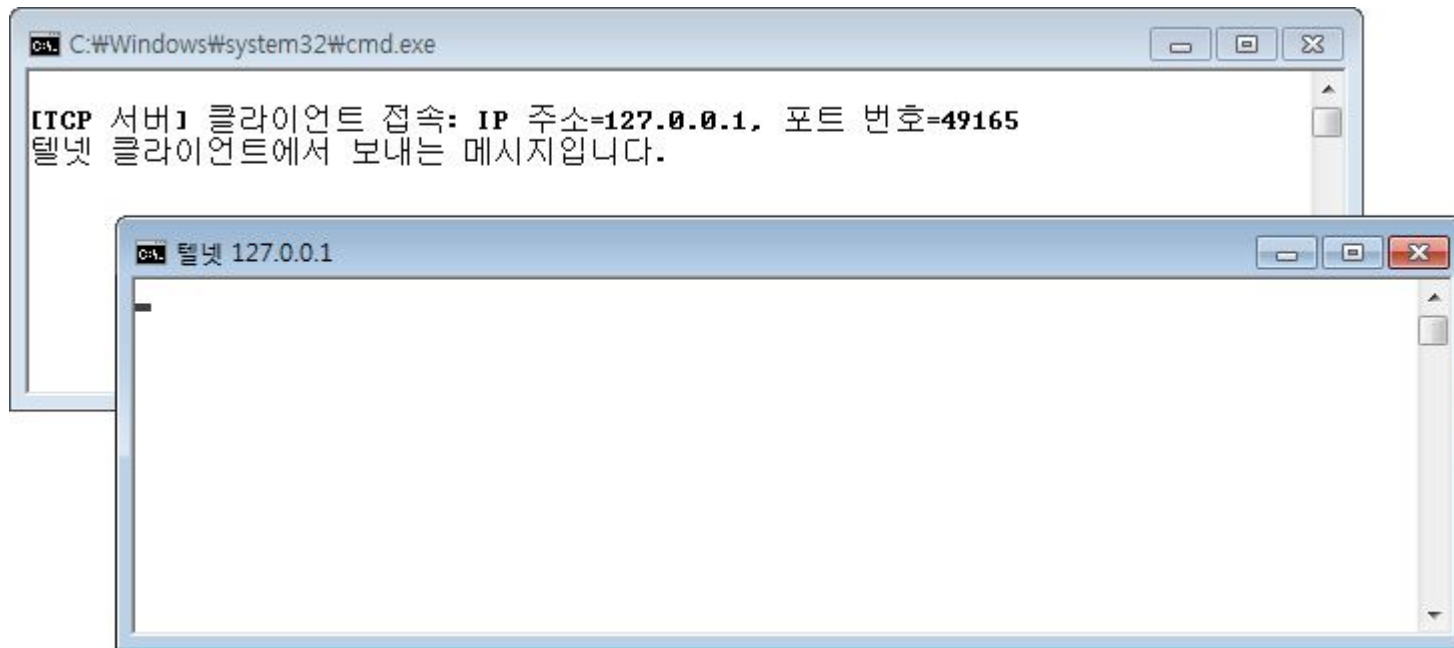
- 비주얼 C++ 6.0을 사용하는 경우에는
[Project]→[Settings]→[Link]→[Object/library modules] 부분에 “ws2_32.lib” 를 입력한다.
- 비주얼 C++ 버전에 따라 원속 라이브러리를 추가하는 방식이 달라서 번거롭다면 소스 코드의 임의 위치에 #pragma comment(lib, “ws2_32”) 한 줄을 넣으면 된다. 2장 이후의 모든 코드는 이 방식을 사용한다. => DLL 사용의 암시적 로딩(Implicit Loading)



윈도우 소켓 프로그램 맛보기 (9)

❖ 실행 화면

- F7 -> Ctrl+F5





Thank You !

oasis01@gmail.com / rhqudtn75@nate.com