Data Communication (CE14773)

Chungnam National University
Dept. of Computer Science and Engineering
Computer Communication Laboratory

Sangdae Kim - 00반 / Cheonyong Kim- 01반

Contents

- Data link layer communication
  - Transmission
  - Ethernet protocol
  - Network API
    - WinPcap
    - WinPcap example
- Chatting & File Transfer Overview
  - Requirement
- Appendix
  - Packet driver function
  - Winpcap function

Data link layer
Communication

Transmission

- Unicast
  - Sending of messages to a single network destination identified by *a unique address*
  - Point-to-point
- Multicast
  - The delivery of a message or information to *a group of destination* computers *simultaneously* in a single transmission from the source creating *copies automatically* in other network elements
  - One-to-many
- Broadcast
  - The distribution of audio and video content to a dispersed audience via radio, television, or other
  - FF:FF:FF:FF:FF:FF – Ethernet layer broadcast address

Ethernet Protocol

- 송신
  - 상위 계층으로부터 데이터를 전달받으면 그 데이터를 프레임의 데이터에 저장
  - 수신될 Ethernet 주소와 자신의 Ethernet 주소를 헤더에 저장
  - 상위 계층의 종류에 따라서 헤더에 상위 프로토콜 형태 저장 후 물리적 계층으로 Ethernet frame 전달
- 수신
  - 하위 계층(physical layer)로부터 프레임을 받으면 상위로 보내야 하는지, 혹은 폐기해야 하는지 결정
  - 상위 계층으로 보내는 기준
    - 목적지 Ethernet 주소가 브로드캐스트 주소(ff-ff-ff-ff-ff-ff)일 경우
    - 목적지 Ethernet 주소가 자신의 Ethernet 주소일 경우
  - Ethernet 프레임 헤더 중에 16 비트 프로토콜 타입 필드를 보고 판단하여 상위 계층으로 전달
    - 0x0800: Internet Protocol (IP)
    - 0x0806: Address Resolution Protocol (ARP)
    - 0x0835: Reverse Address Resolution Protocol (RARP)
- Ethernet 프레임에 담을 수 있는 최대 전송 단위 (MTU) : 1500bytes (데이터만)

Network API

◈ 요구사항

- ❖ 가능한 낮은 데이터 링크 계층까지 접근 가능
- ❖ Promiscuous Filter 가능
- ❖ Asynchronous Mode 로 동작 가능
- ❖ 사용자 정의 프로토콜을 사용할 수 있어야 함
- ❖ 사용자 계층에서 쉽게 접근 가능
- ❖ 네트워크 인터페이스의 정보 열람 가능
- ❖ 로컬 컴퓨터의 네트워크 설정 정보를 열람 가능
- ❖ 장치 드라이버 계층의 정보를 질의/설정 가능

Network API (cont.)

- Winsock
  - 가장 대표적인 Internet Program 개발 API
  - Network Layer까지 접근 가능
    - Raw Mode – Internet layer(IP)까지 지원
  - Support Multicasting
  - Overlapped Operation 지원
    - WSA_FLAG_OVERLAPPED flag
  - Buffering management(Winsock 2.0)
  - 다양한 프로토콜 지원
    - TCP, UDP, IP, DNS, ICMP, IGMP, IPX/SPX, …etc
  - 다양한 API 지원
    - Address conversion, byte ordering
  - UNIX, Window 모두 지원

Network API (cont.)

- Packet Driver
  - Data Link Layer까지 접근 가능
    - Read/Write의 기본 계층이 Ethernet Layer.
  - Promiscuous operation 지원
    - Shared media의 모든 데이터 Read 가능
  - Asynchronous operation 지원
  - Kernel Driver 수준의 정보 Query/Setting 가능
    - H/W 주소 Query, Link Speed
  - Protocol을 직접 구현 해야 함
  - Buffering을 직접 구현해야 함
  - Socket 보다 사용하기 어려움
  - http://www.htsns.com/LoginTest/ddk/main.html
    - NDIS DDK 도움말 참고

Network API (cont.)

◈ Winsock VS Packet Driver

	Winsock	Packet Driver
접근가능 계층	일반적으로 전송 계층 (제한적 접근) **SOCK_RAW**모드 시 네트워크 계층까지 접근 (송신만 가능)	데이터 링크계층까지 모두 접근 가능
OVERLAPPED 방식	**WSA_FLAG_OVERLAPPED flag**를 통해 지원가능	지원가능
비동기 방식 송수신	지원가능	지원가능
드라이버 정보에 대한 질의 및 설정 가능	**API**를 통해 지원가능	지원가능
지원 프로토콜	전송계층까지 지원	직접 구현 해야 함
버퍼링	지원함	직접 구현 해야 함

Network API (cont.)

- WinPcap
  - WinPcap is *an open source library for packet capture and network analysis* for the Win32 platforms.
  - It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll, based on libpcap version 0.6.2).
  - The packet filter is a device driver that adds to OS the ability to capture and send raw data from a network card, with the possibility to filter and store in a buffer the captured packets.
  - Packet.dll is an API that can be used to directly access the functions of the packet driver, offering a programming interface independent from the Microsoft OS.
  - Wpcap.dll exports a set of high level capture primitives that are compatible with libpcap, the well known Unix capture library. These functions allow to capture packets in a way independent from the underlying network hardware and operating system.

Network API (cont.)

- Installation of WinPcap
  - WinPcap Download
    - http://www.winpcap.org
    - Get WinPcap에서 winpcap을 받아 설치 (winpcap 4.1.3)
    - Development에서 Developer's Pack 다운로드 (winpcap 4.1.2)
  - Wiresharck Download
    - https://www.wireshark.org/download.html
    - 실제 패킷 전송 상태를 볼 수 있는 패킷 캡쳐 프로그램

Network API (cont.)

◈ Installation of WinPcap

- ❖ Visual studio에서 winpacp을 사용하기 위해서 별도의 설정이 필요함
- ❖ 다운로드 받은 Developer's pack 압축해제 (WpdPack 생성)
- ❖ Winpcap을 사용할 프로젝트를 생성 및 open

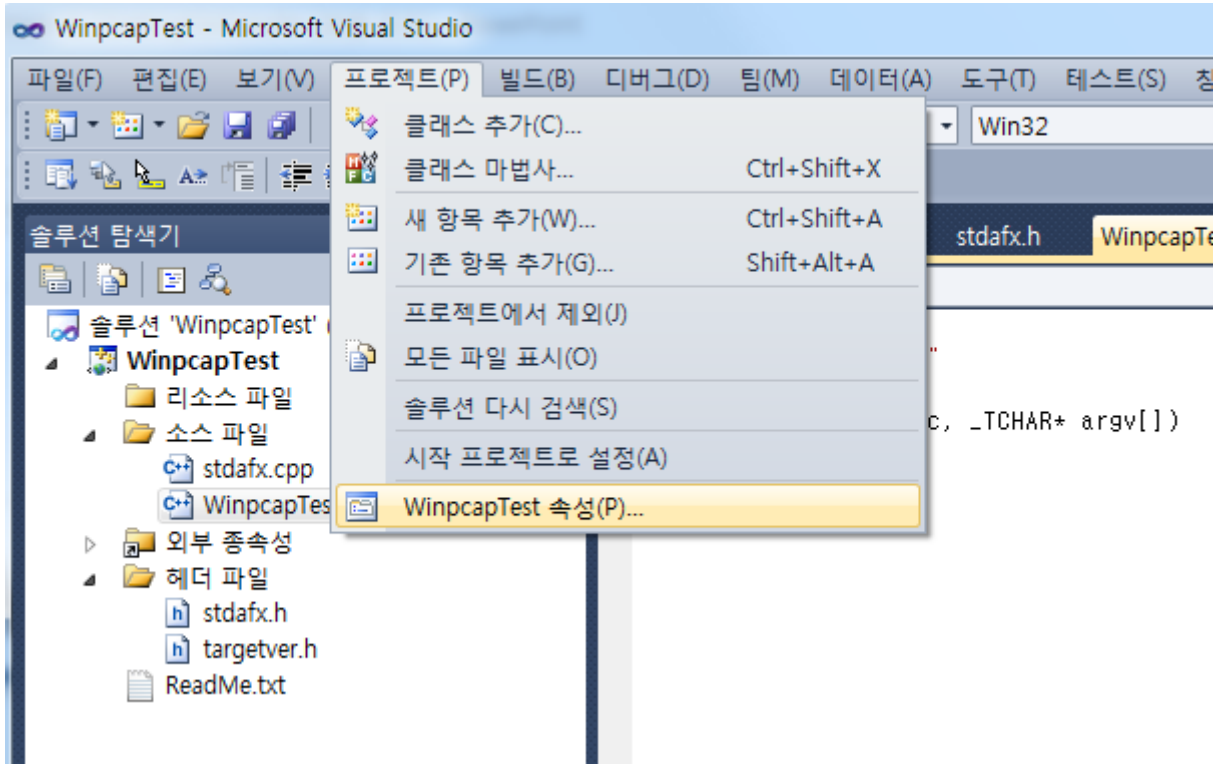
◈ 설정환경

- ❖ Visual studio 2010
- ❖ 윈도우 7 64bit
- ❖ Developer's pack 경로 : C:\WpdPack
- ❖ 적용 프로젝트 : Winpcap을 이용하기 위한 새 프로젝트

Network API (cont.)

◈ Installation of WinPcap

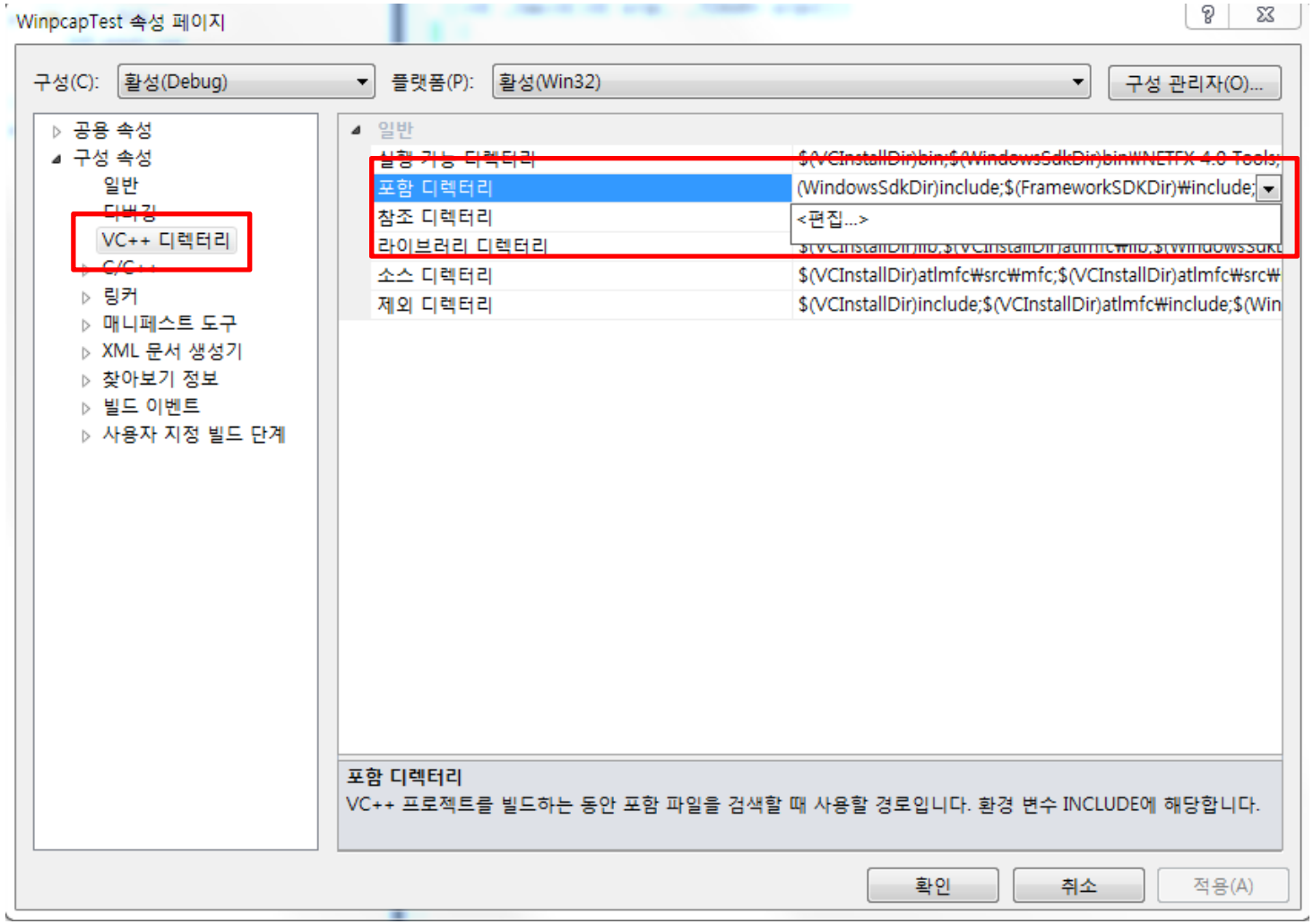
❖ 프로젝트 -> (프로젝트명)속성 클릭



Network API (cont.)

◈ Installation of WinPcap

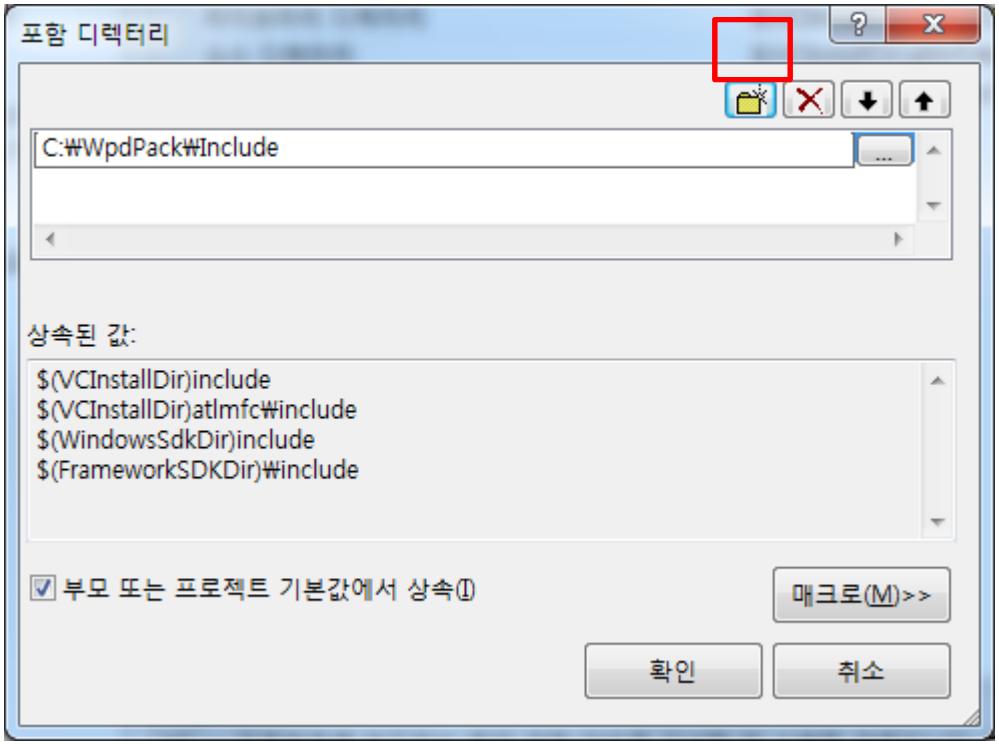
❖ 포함 디렉터리 -> 편집 클릭



Network API (cont.)

◈ Installation of WinPcap

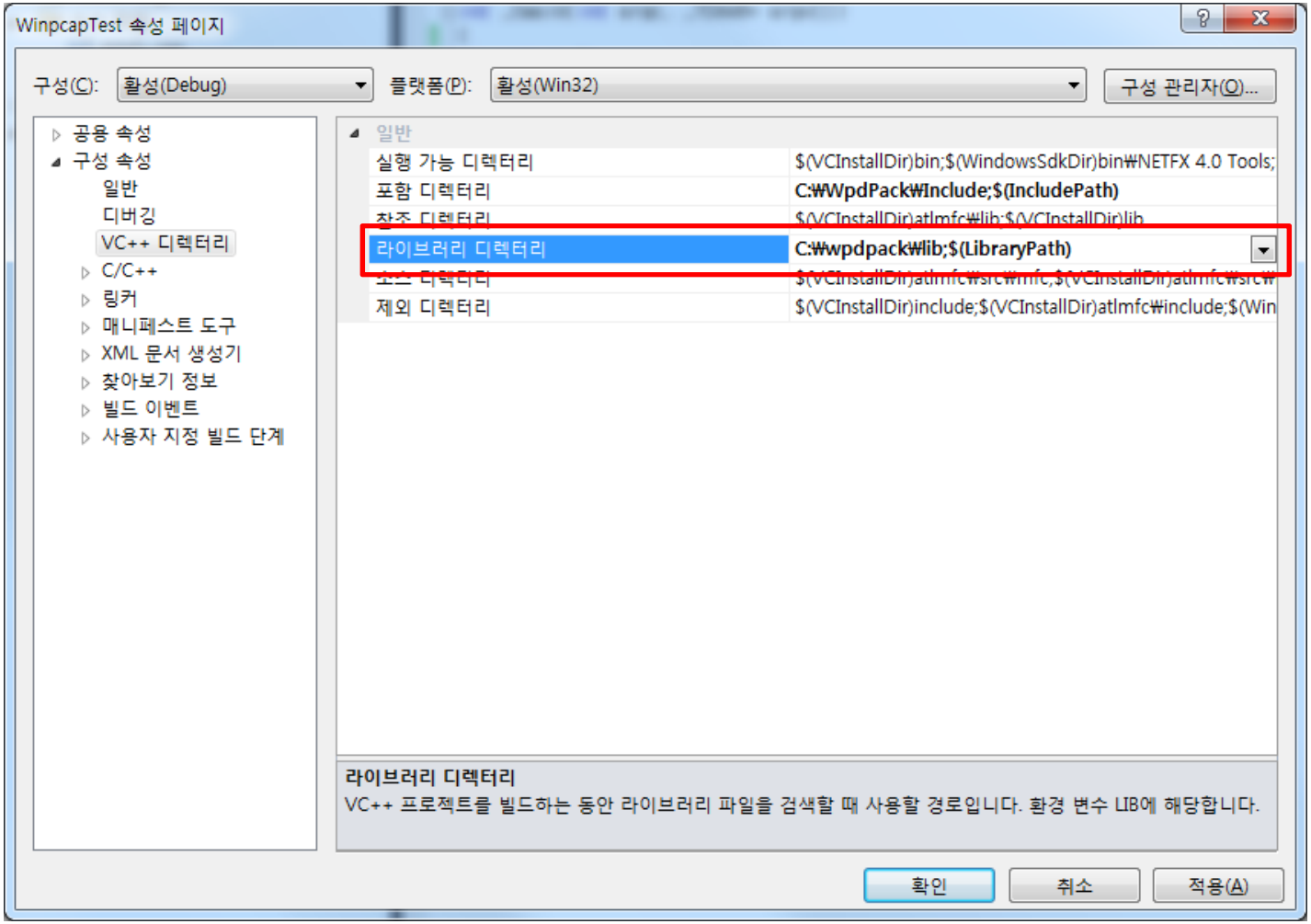
- ❖ Developer's pack 압축 해제경로를 추가.
- ❖ 경로를 입력한 후 확인



Network API (cont.)

◈ Installation of WinPcap

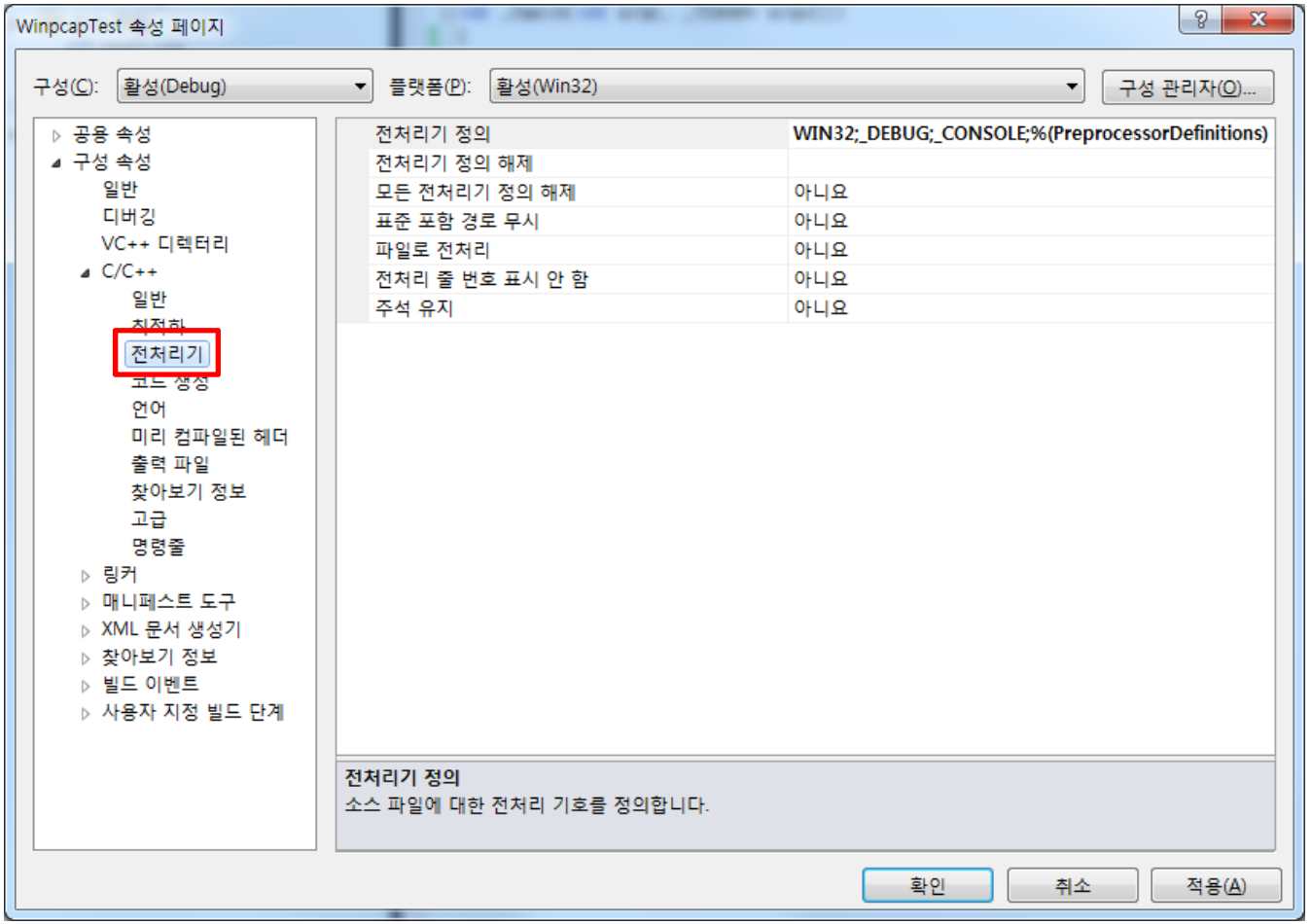
❖ 마찬가지로 라이브러리 디렉터리에도 WpdPack의 Lib를 추가



Network API (cont.)

◈ Installation of WinPcap

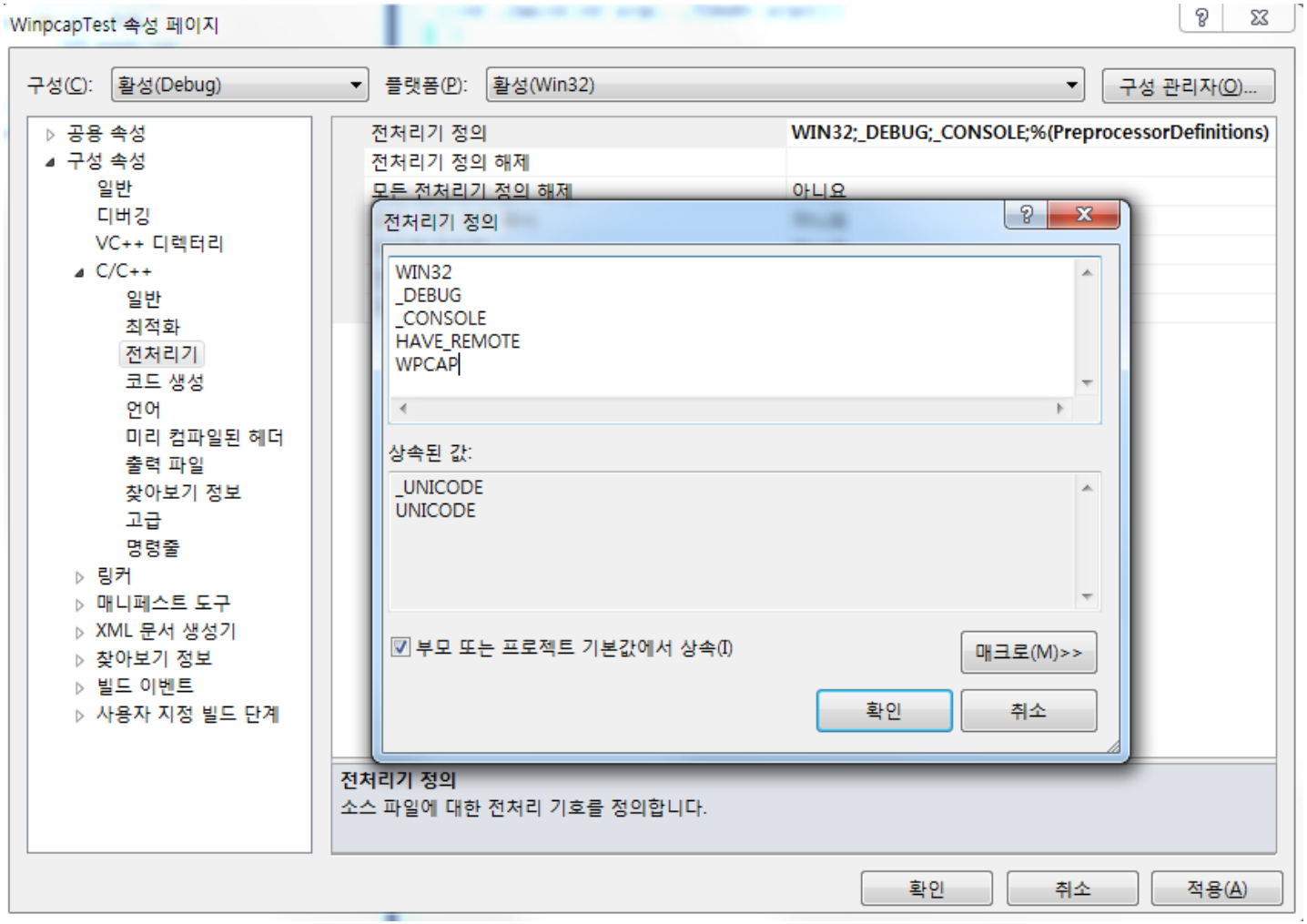
❖ 상기의 설정이 끝나면 C/C++ ➔ 전처리기로 이동



Network API (cont.)

◈ Installation of WinPcap

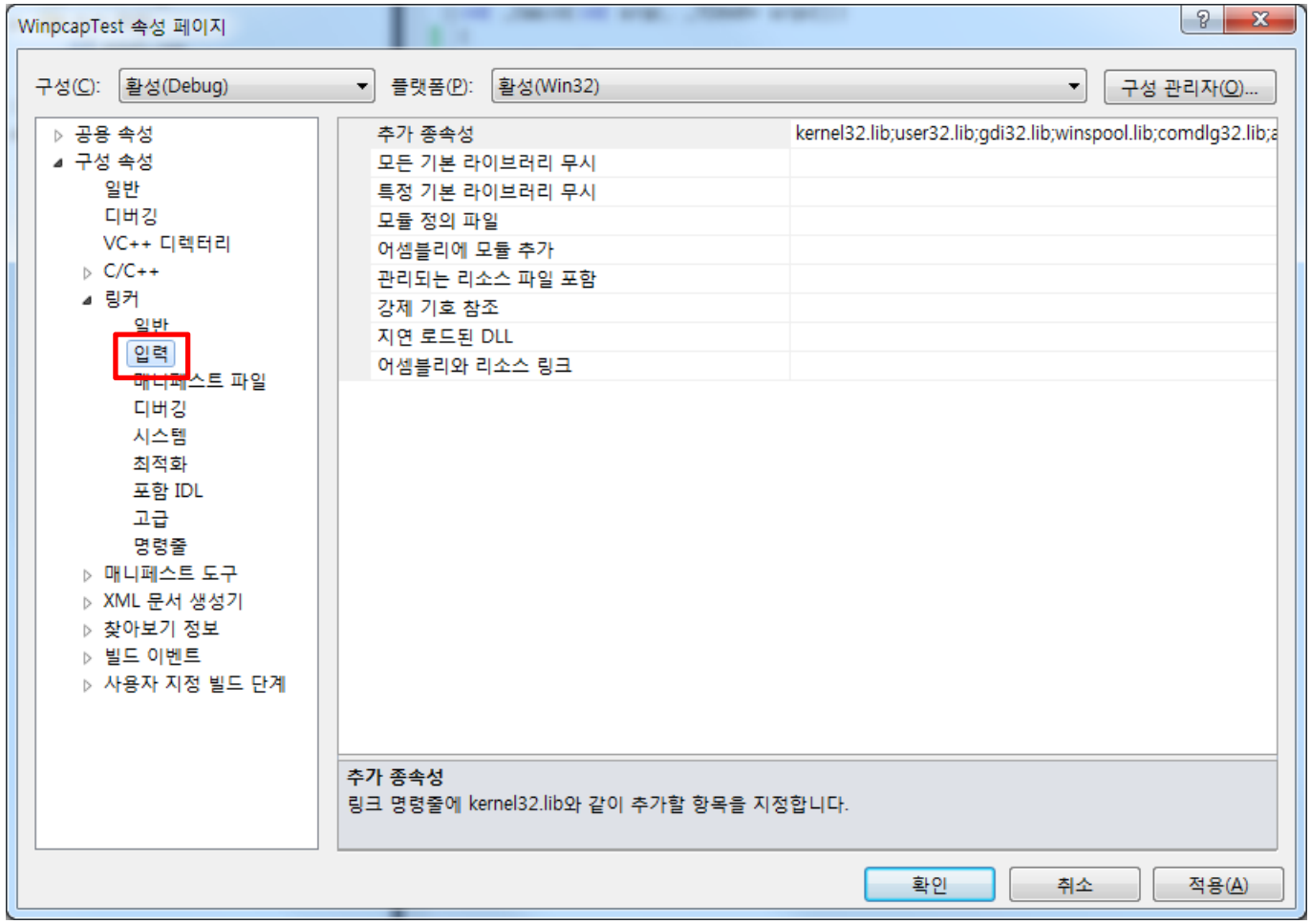
❖ 마찬가지로 편집 탭을 통해 HAVE_REMOTE, WPCAP을 추가



Network API (cont.)

◈ Installation of WinPcap

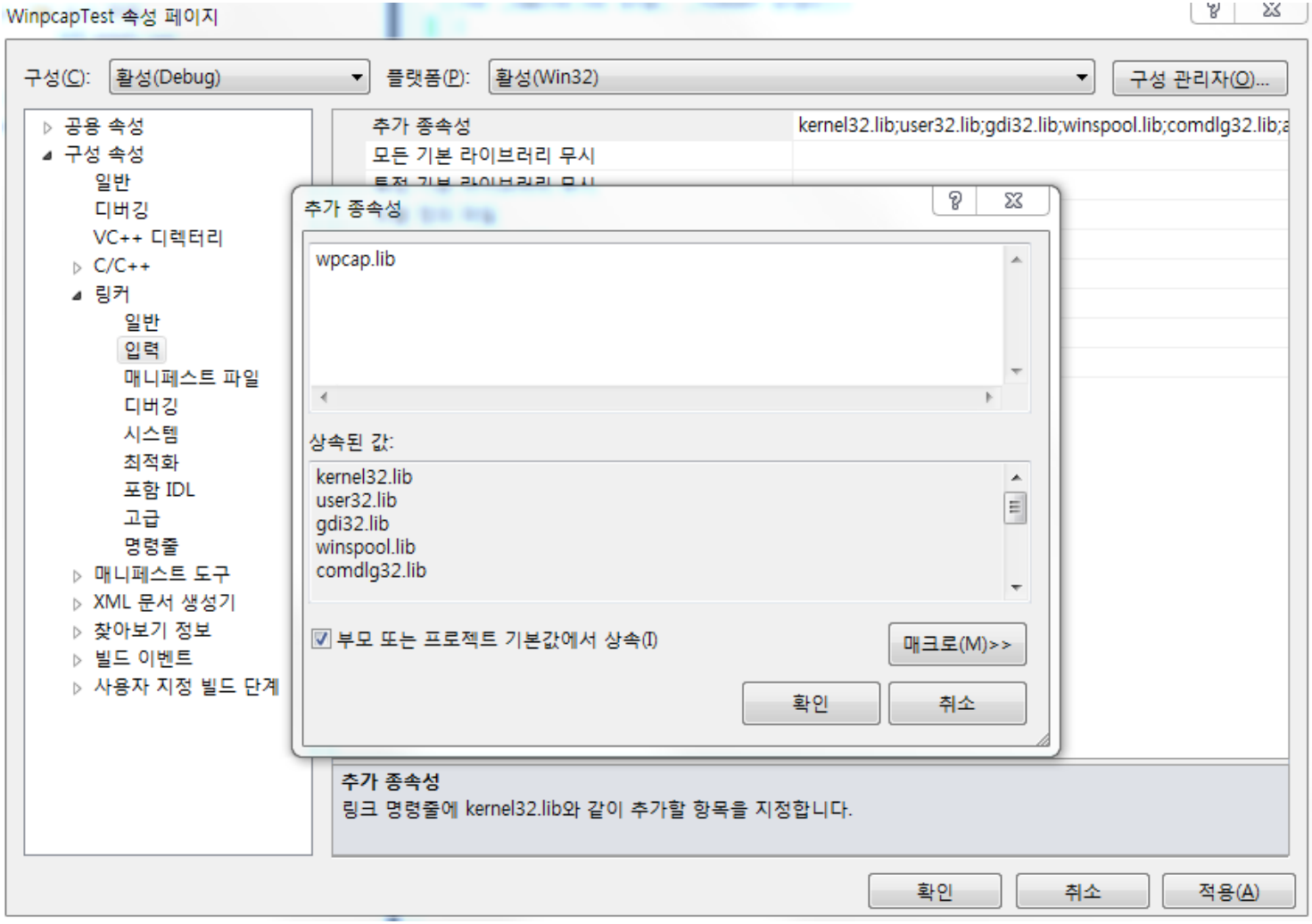
❖ 상기의 설정이 끝나면 링커 ➔ 입력으로 이동



Network API (cont.)

◈ Installation of WinPcap

❖ 편집 탭을 통해 wpcap.lib를 추가



Network API (cont.)

◈ Installation of WinPcap

- ❖ 상기의 설정을 완료하면 Visual studio 2010에서 winpcap의 함수가 정의된 pcap.h를 사용할 수 있음
- ❖ 이외에 제공되는 Network API (winsock2.h 또는 packet32.h)를 사용하기 위해서 앞서 wpcap.lib과 마찬가지로, ws_32.lib, packet.lib를 추가하여 사용하거나 다음과 같이 선언하여 사용할 수 있다.

```
#include "stdafx.h"
#include <pcap.h>
#include <WinSock2.h>
#pragma comment(lib, "ws2_32.lib")
#include <Packet32.h>
#pragma comment(lib, "packet.lib")
```

Network API (cont.)

- Example of Network API
  - Network API를 활용하여 네트워크 장치에 접근 및 정보를 얻어오는 방법을 소개
  - 예제에서 사용되는 Network API
    - pcap_findalldevs
    - pcap_lookupnet
    - pcap_open_live
    - pcap_sendpacket
    - PacketOpenAdapter
    - PacketRequest
  - Host1에서 Broadcasting으로 패킷을 전달하는 예제

Network API (cont.)

◈ Example of Network API

❖ 예제 실습을 위한 헤더 및 자료구조 선언

```
#include "stdafx.h"
#include <pcap.h>
#include <WinSock2.h>
#pragma comment(lib, "ws2_32.lib")
#include <Packet32.h>
#pragma comment(lib, "packet.lib")

/* Ethernet Header Structure */
#define  ETHER_ADDR_LEN 6
typedef struct ether_header {
    unsigned char   ether_dhost[ETHER_ADDR_LEN];
    unsigned char   ether_shost[ETHER_ADDR_LEN];
    unsigned short  ether_type;
} ETHER_HDR;
```

Network API (cont.)

◈ Example of Network API

❖ 예제 실습을 위한 변수 선언 및 초기화

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i=0, length = 0;
    char errbuf[PCAP_ERRBUF_SIZE];
    unsigned char packet[1500];

    pcap_if_t *alldevs, *d;
    pcap_t *fp;

    bpf_u_int32 net, mask;
    struct in_addr net_addr, mask_addr;

    LPADAPTER adapter = NULL;
    PPACKET_OID_DATA OidData;

    ETHER_HDR eth;

    OidData = (PPACKET_OID_DATA)malloc(sizeof(PACKET_OID_DATA));
    OidData->Oid = 0x01010101;
    OidData->Length = 6;
```

Network API (cont.)

◈ Example of Network API

❖ 현재 Host에서 접근할 수 있는 네트워크 장치 검색 및 화면 출력 ①

```
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{
    fprintf(stderr,"Error in pcap_findalldevs: %s₩n", errbuf);
    exit(1);
}

for(d=alldevs; d; d=d->next)
{
    printf("%d. %s", ++i, d->name);
    if (d->description)
        printf(" (%s)₩n", d->description);
    else
        printf(" (No description available)₩n");
}
```

Network API (cont.)

- Example of Network API
  - 현재 Host에 있는 장치의 정보를 가져온다

```
for(d=alldevs, i=0; i< 1-1 ;d=d->next,i++);

if(pcap_lookupnet(d->name, &net, &mask, errbuf) < 0){
    return 0;
    printf("네트워크 디바이스 정보를 가져올 수 없습니다.₩n");
}
```

  - 현재 예제를 진행하는 Host에는 네트워크장치가 1개
  - 2개 이상의 네트워크 장치가 존재하고, 그 중 하나를 선택해야 하는 경우, 박스안의 내용을 수정해야됨

Network API (cont.)

◈ Example of Network API

❖ 불러들여온 네트워크 장치의 값을 읽어올 수 있게 open 및 read.

❖ read한 네트워크 장치의 정보를 출력 ②

```
adapter = PacketOpenAdapter(d->name);
PacketRequest( adapter, FALSE, OidData);

net_addr.s_addr = net;
mask_addr.s_addr = mask;

printf("Device Name : %s ₩n", d->name);
printf("Network Address : %s ₩n", inet_ntoa(net_addr));
printf("Netmask Info : %s ₩n", inet_ntoa(mask_addr));
printf("Mac Address : %02x-%02x-%02x-%02x-%02x-%02x₩n",
    OidData->Data[0], OidData->Data[1], OidData->Data[2],
    OidData->Data[3], OidData->Data[4], OidData->Data[5] );
```

Network API (cont.)

◈ Example of Network API

❖ 네트워크 장치에 packet을 송/수신이 가능하게 open(파일포인터와 유사)

❖ 선언되었던 Ethernet header의 양식에 맞추어 값을 저장.

```
if ( (fp= pcap_open_live(d->name, 65536, 0, 1000, errbuf) ) == NULL)
{
    fprintf(stderr,"₩nUnable to open the adapter. %s is not supported by WinPcap₩n", argv[1]);
    return -1;
}

memset(packet, 0, sizeof(packet));

eth.ether_dhost[0] = 0xff;  eth.ether_dhost[1] = 0xff;
eth.ether_dhost[2] = 0xff;  eth.ether_dhost[3] = 0xff;
eth.ether_dhost[4] = 0xff;  eth.ether_dhost[5] = 0xff;

eth.ether_shost[0] = OidData->Data[0];  eth.ether_shost[1] = OidData->Data[1];
eth.ether_shost[2] = OidData->Data[2];  eth.ether_shost[3] = OidData->Data[3];
eth.ether_shost[4] = OidData->Data[4];  eth.ether_shost[5] = OidData->Data[5];

eth.ether_type = 0x1234;

memcpy(packet, &eth, sizeof(eth));
length += sizeof(eth);
```

❖ 시작점 - 자신의 Mac주소, 목적지 – broadcast(0xffffffff), type – 0x1234

Network API (cont.)

◈ Example of Network API

❖ open한 네트워크 장치에 생성한 패킷을 전달함.

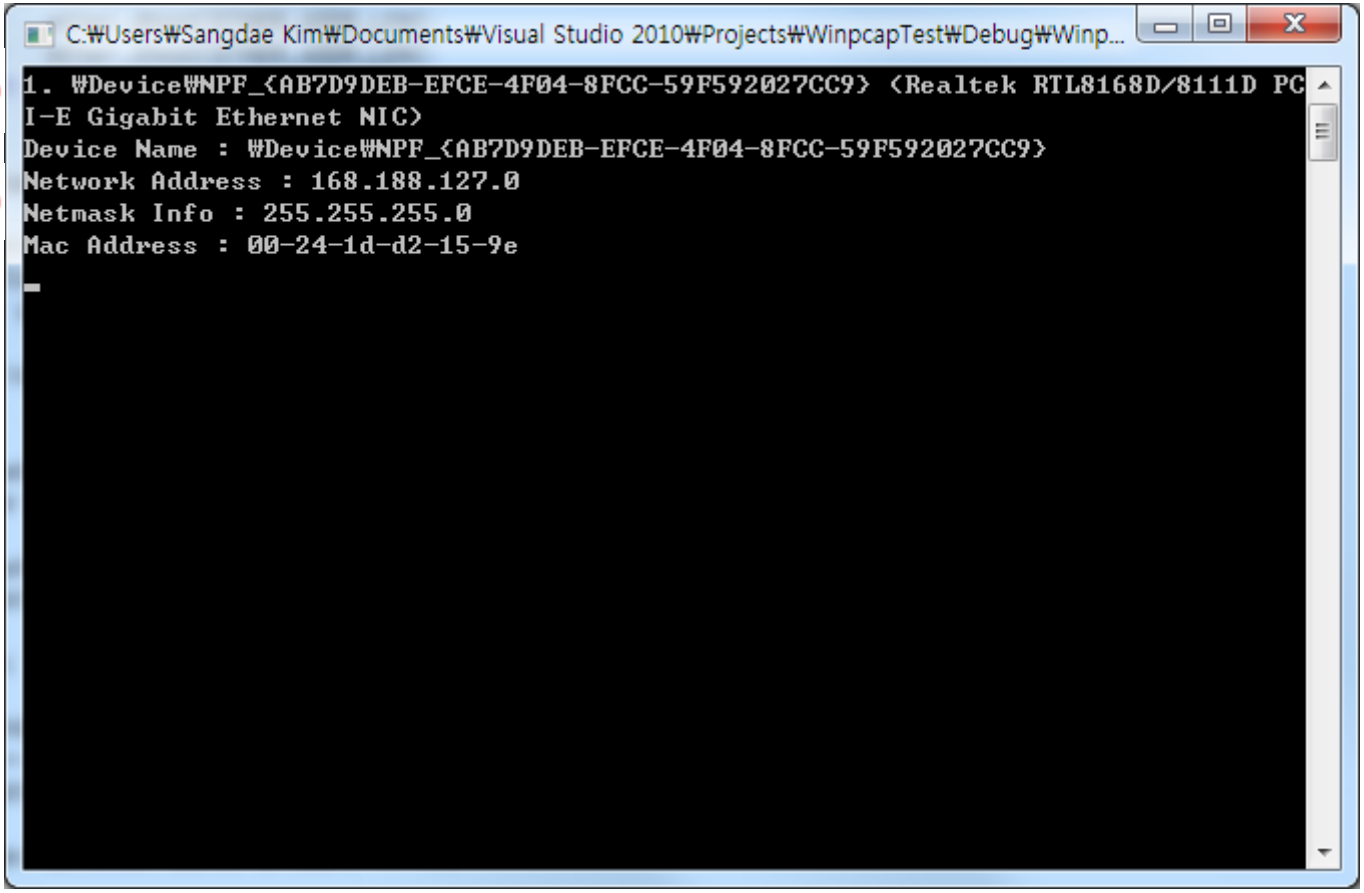
```
        if (pcap_sendpacket(fp, packet, length ) != 0)
        {
            fprintf(stderr,"₩nError sending the packet: ₩n", pcap_geterr(fp));
            return -1;
        }

        return 0;
    }
```

❖ 본 예시는 아무런 내용이 없이 단순히 Ethernet header만을 작성함.

Network API (cont.)

◈ Example of Network API

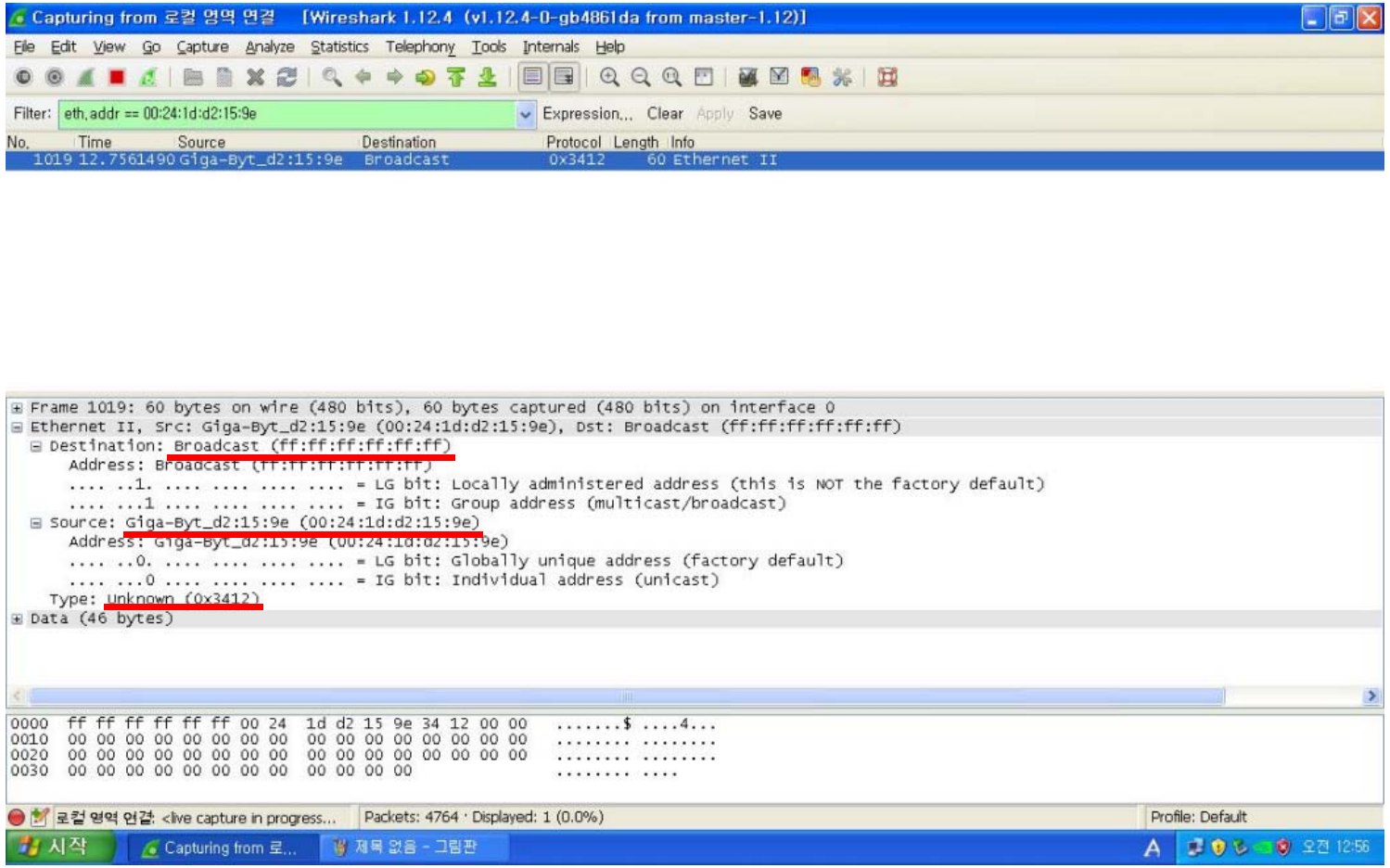
❖ 실행결과 (Host PC)



Network API (cont.)

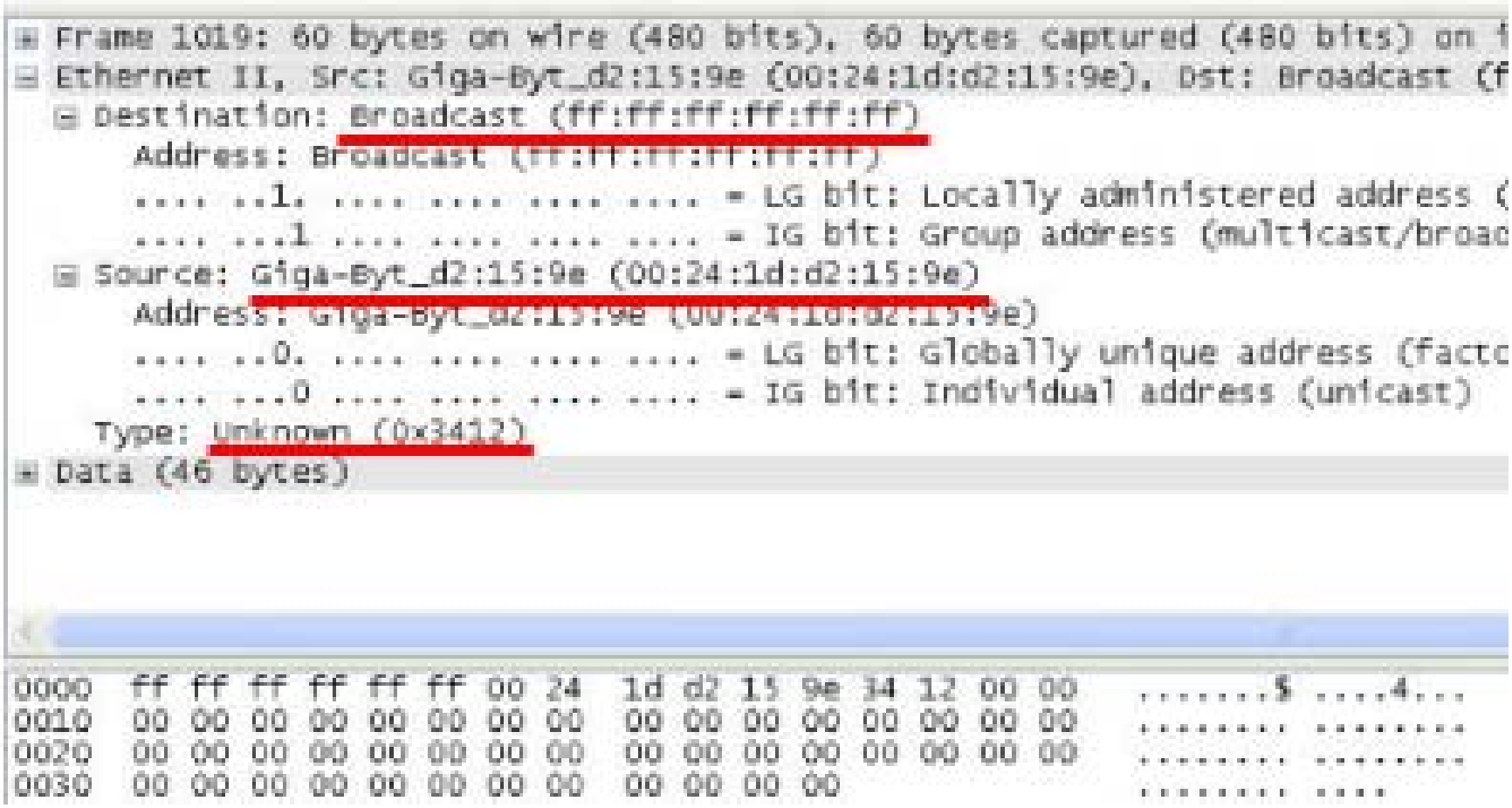
◈ Example of Network API

❖ 실행결과 (Broadcast 패킷을 전달받은 임의의 PC)



Network API (cont.)

- Example of Network API



Chatting & File Transfer Overview

Introduction

Homework #3

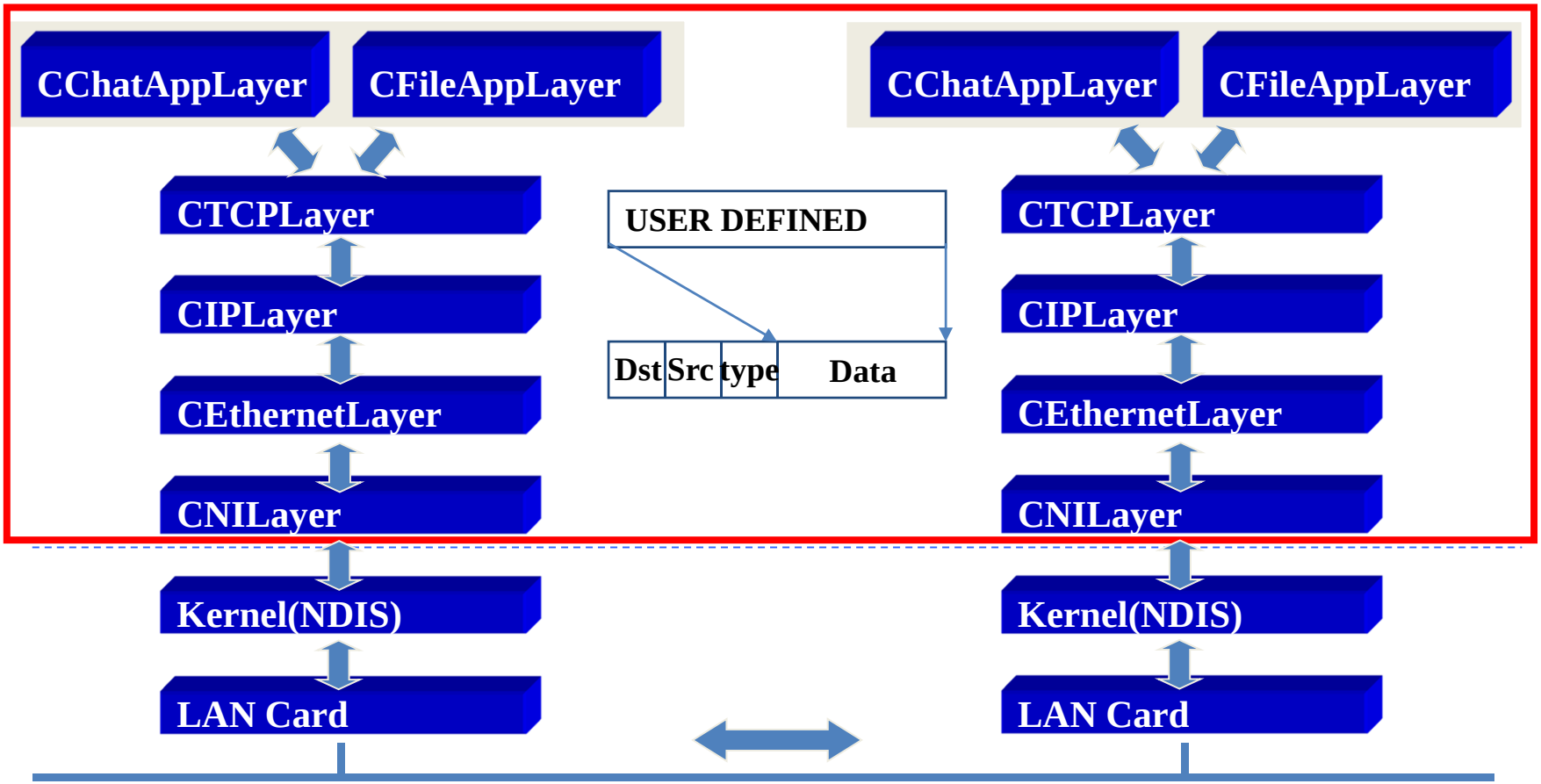
- To implement the chatting and File transfer client

Description

- Homework #2(IPC)에서는 동일 시스템 안에서 네트워크의 기본적인 구조와 개념을 이용하여 Process들 간의 메시지 통신을 하는 것이었으나 Homework #3에서는 이를 확장하여 다른 시스템(machine)의 process간의 통신을 할 수 있도록 하는 것이다.
- 과제 산출물은 실제 Network (Ethernet Protocol) 을 이용해서 Chatting과 File 전송을 할 수 있는 Application을 구현하는 것

Introduction (cont.)

- Layer Model



35

Data Structures

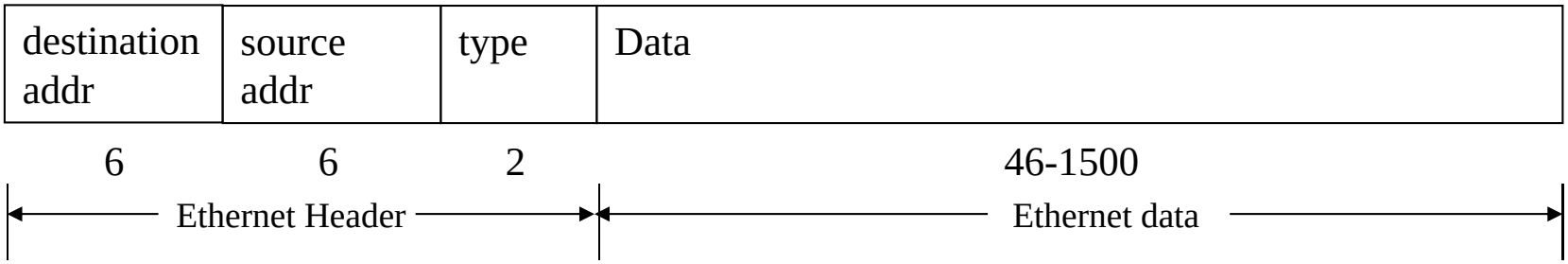
- Ethernet Address

```
typedef struct _ETHERNET_ADDR
{
        union {
                struct { e0, e1, e2, e3, e4, e5; } s_un_byte ;
                unsigned char s_ether_addr[6] ;
        } S_un ;

#define addr0       S_un.s_un_byte.e0
#define addr1       S_un.s_un_byte.e1
#define addr2       S_un.s_un_byte.e2
#define addr3       S_un.s_un_byte.e3
#define addr4       S_un.s_un_byte.e4
#define addr5       S_un.s_un_byte.e5

#define addr        S_un.s_ehter_addr

} ETHERNET_ADDR, *LPETHERNET_ADDR ;
```

Data Structures

- Ethernet Protocol

```
typedef struct _ETHERNET
{
        ETHERNET_ADDR   enet_desaddr ;      // 상대방 주소
        ETHERNET_ADDR   enet_srcaddr ;      // 자기 주소
        unsigned short  enet_type ;         // frame data type
        unsigned char   enet_data[ MAX_ETHERNET_DATA ] ;
} ETHERNET, *LPETHERNET ;
```

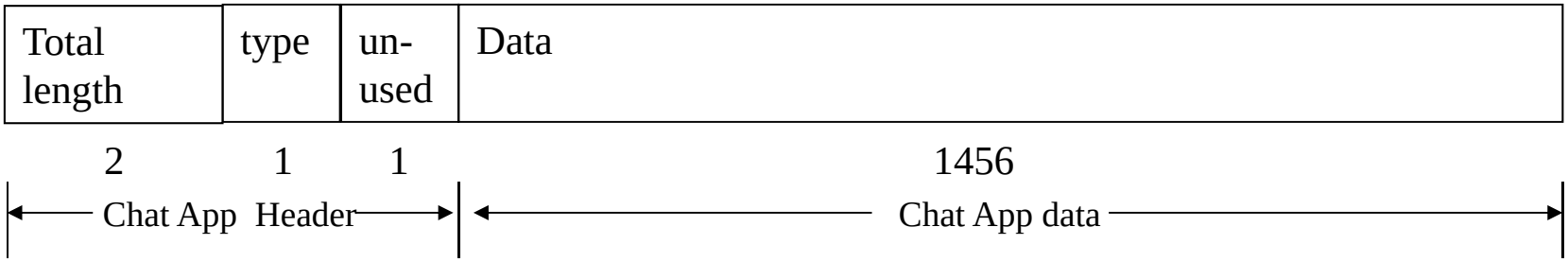


37

Data Structures

- Chat Application Protocol

```
typedef struct _CHAT_APP
{
        unsigned short      capp_totlen ;       // 메시지 총 길이
        unsigned char       capp_type ;         // 메시지 타입
        unsigned char       capp_unused ;       // 우선 사용 안함
        unsigned char       capp_data[ MAX_APP_DATA ] ;
} CHAT_APP, *LPCHAT_APP ;
```

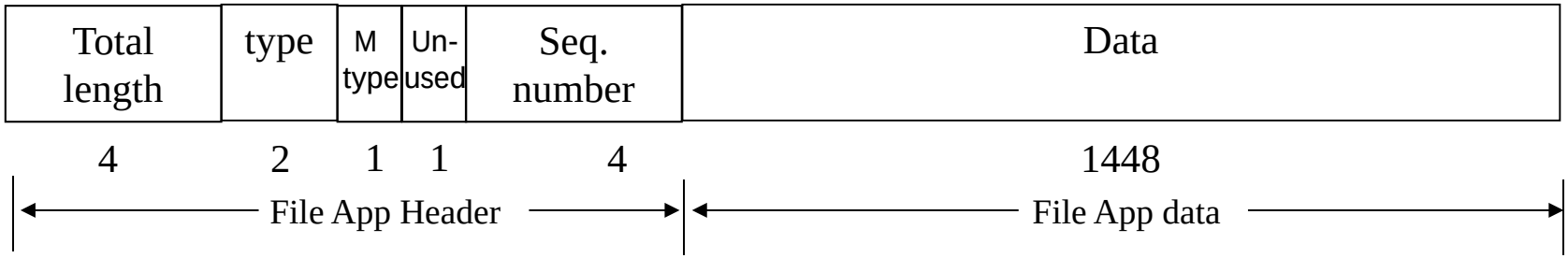


38

Data Structures

File Transfer Application Protocol

```
typedef struct _FILE_APP
{
        unsigned long       fapp_totlen ;       // 총 길이
        unsigned short      fapp_type ;         // 데이터 타입
        unsigned char       faa_msg_type        // 메시지 종류
        unsigned char       unused ;            // 사용 안함
        unsigned long       fapp_seq_num ;      // fragmentation 순서
        unsigned char       fapp_data[ MAX_APP_DATA ] ;
} FILE_APP, *LPFILE_APP ;
```



39

Requirements Analysis

◈ Basic Design

- ❖ One to one 방식
  - ◆ 시스템에는 하나의 프로세스만 동작하며 통신할 상대는 다른 시스템의 프로세스로써 1:1통신을 한다.
- ❖ Chatting과 File 전송을 동시에 가능
  - ◆ File Transfer Thread 구현
- ❖ Chatting Message와 File Size는 제한 없이 전송 가능
  - ◆ Fragmentation 가능
- ❖ 실제 Network Protocol (Ethernet Protocol)을 이용하여 Ethernet Frame을 송수신
  - ◆ Packet driver 이용하여 정보를 얻어 winPcap을 이용하여 송수신
- ❖ Ethernet Frame을 수신
  - ◆ 수신만을 하기 위한 Thread 구현
- ❖ class CBaseLayer는 homework #2(IPC)의 소스를 그대로 사용할 것

Requirements Analysis (cont.)

◈ Optional Design

❖ Reliable Communication

- ◆ 송수신 간의 데이터 전송 시 신뢰성을 보장하기 위한 메시지 구현
- ◆ 데이터 전송 후 모든 데이터 전송이 성공되면 수신 측에서 잘 받았다는 메시지를 송신 측에게 보내고 이를 받으면 송신 측에서는 다음 전송을 대기한다.
- ◆ 만약, 송신 측에서는 보내고 수신 측에서는 수신 준비가 안 되었을 경우 송신 측에서 수신 대기 하지 않는 것을 감지하여 (Timer를 작동 후에 2초간 대기하여 응답이 없을 경우) “time-out” 메시지를 화면에 출력할지 아닐지를 결정
- ◆ Hint. : chatApp와 FileApp의 data type을 추가하여 그 자료구조를 이용
- ◆ 구현 시 가산 점수 부여

Requirements Analysis (cont.)

- Chatting Application Layer (class CChatAppLayer)
  - Field Port Number of TCP Layer : 2080 (user defined)
  - Chatting Message에 대해서 Fragmentation을 함으로써 길이에 제한 없이 전송하도록 구현
  - Fragmentation
    - 전송 시
      - 데이터를 일정 크기로 잘라서 하나씩 전송한다.
      - capp_type에 메시지 데이터가 첫 부분(0x00)인지 중간 부분 (0x01) 인지, 마지막 부분 (0x02) 인지 구분하여 저장 후 전송
    - 수신 시
      - 첫 부분 일 경우 그 크기만큼 버퍼 할당하고 중간부분이면 계속 버퍼에 쌓는다.
      - 마지막 부분일 경우 받은 메시지를 모두 모아서 Dialog 객체에게 넘겨준다.

Requirements Analysis (cont.)

- File Transfer Application Layer (class CFileAppLayer)
  - Field Port Number of TCP Layer : 2090 (user defined)
  - File에 대해서 Fragmentation을 함으로써 size에 제한 없이 전송하도록 구현
  - Fragmentation
    - 전송 시
      - 파일명을 얻고, 파일을 열어서 버퍼에 저장
      - 첫 부분 : 파일에 대한 정보를 전송 (파일 명), fapptype = 0x00
      - 중간부분 : 파일의 데이터가 전송 (Fragmentation적용), fapptype = 0x01
      - 마지막 부분 : 모든 데이터가 다 전송되고 마지막이라는 메시지, fapptype = 0x02
      - 파일을 닫는다
    - 수신 시
      - 첫 부분(fapptype = 0x00) 일 경우 그 파일명을 받아서 파일을 생성
      - 중간부분 (fapptype = 0x01) 일 경우 파일에 데이터를 써서 덧붙인다
      - 마지막 부분 (fapptype = 0x02) 일 경우 파일의 크기 및 순서를 고려 잘 전송 받았는지 확인하고 파일을 닫는다.

Requirements Analysis (cont.)

◈ TCP Layer(class CTCPLayer)

- ❖ 수신 받은 데이터로부터 Port 번호를 읽어서 Chatting Application Layer 나 File Transfer Application Layer로 구분해서 보낸다.
- ❖ 그 밖의 구현에 필요한 member method는 추가해도 무방
- ❖ Dest_Port가 2080이면 ChatappLayer에게, 2090이면 FileAppLayer에게 데이터 넘김

◈ IP Layer (class CIPLayer)

- ❖ 미리 저장한 IP주소를 저장해서 송신한다.
- ❖ 수신 받은 데이터로부터 IP주소를 읽어서 자신의 것이면 TCP Layer로 보내고, 아니면 Discard한다.

Requirements Analysis (cont.)

◈ Ethernet Layer(class CEthernetLayer)

- ❖ 수신받은 프레임 중에서 destination address가 자기 것이 아니면 discard한다.
- ❖ 그 밖의 구현에 필요한 member method는 추가해도 무방
- ❖ 헤더를 제외한 데이터 부분은 IPLayer로 전송

◈ Network Interface Layer (class CNILayer)

- ❖ winPcap을 이용하여 기본적인 packet 송수신 operation 구현한 class
- ❖ Adapter와 상위 Layer간의 데이터 송수신의 중간자 역할을 담당
- ❖ CBaseLayer를 상속
- ❖ Packet32.h를 이용하여 Mac 정보를 얻어온다.

Requirements Analysis (cont.)

◈ Thread 구현

❖ UINT ReadingThread(LPDWORD lpdwParam)

◆수신된 Ethernet Frame을 받아서 상위 레이어로 올리는 역할을 담당

◆Synchronous Mode로 동작

```
BOOL PacketReceivePacket( LPADAPTER AdapterObject,
LPPACKET lpPacket, BOOLEAN Sync, PULONG BytesReceived )
// sync = true ;
```

❖ UINT FileTransferThread(LPDWORD lpdwParam)

◆Chatting과 File Transfer를 동시에 가능하게 하기 위해 Thread를 구현

User Interface (UI)

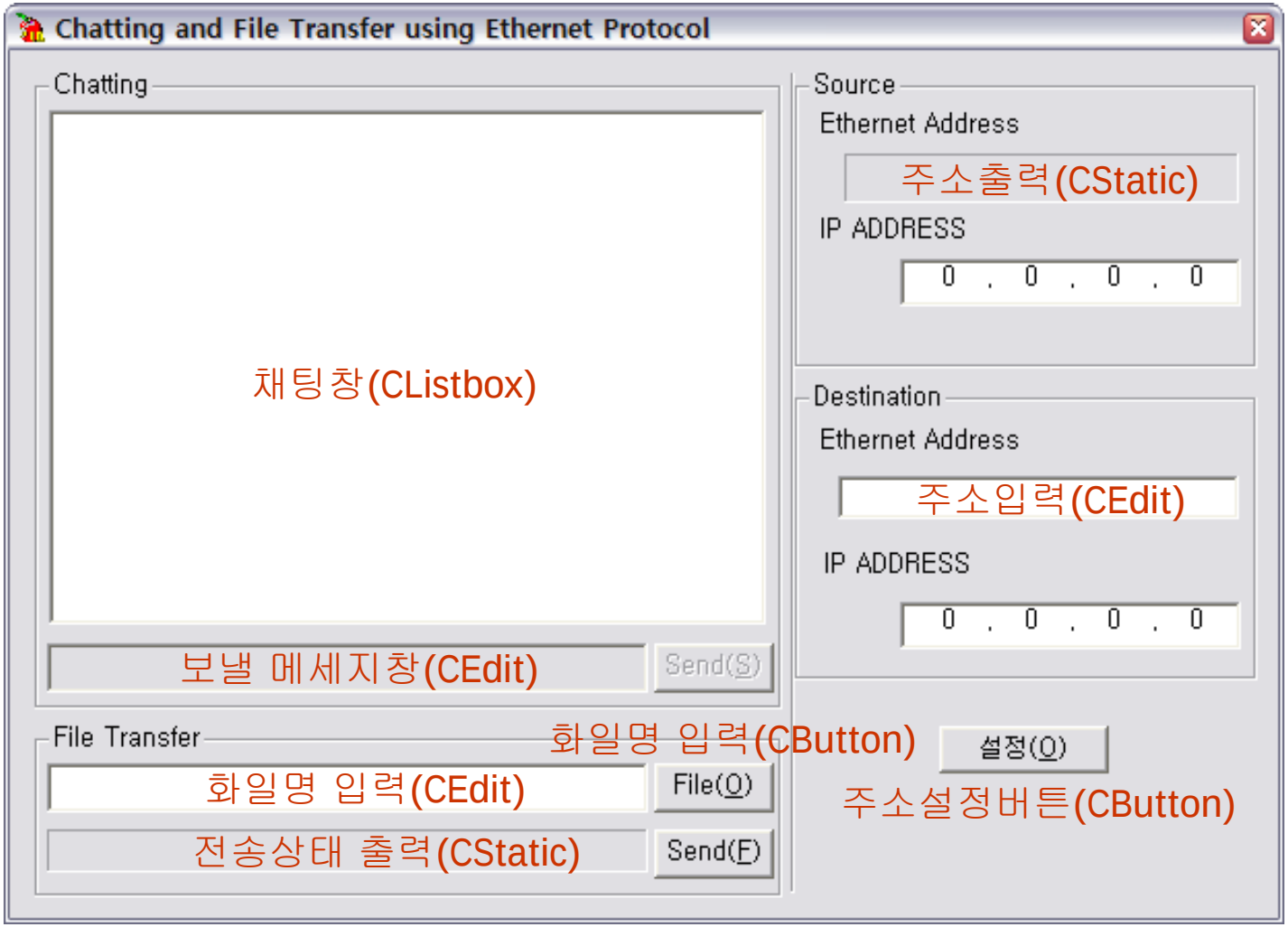
- Chatting
  - Chatting Message가 출력될 Window
  - 보내질 Chat message가 입력될 Window
  - Send Button

- File Transfer
  - File Name을 불러올 Button
  - File Name을 화면에 출력할 Window
  - 전송 상태를 보여줄 Window
  - Send Button

- Address
  - Source Ethernet Address 보여줄 Window
  - Destination Ethernet Address 입력할 Window
  - 설정 Button

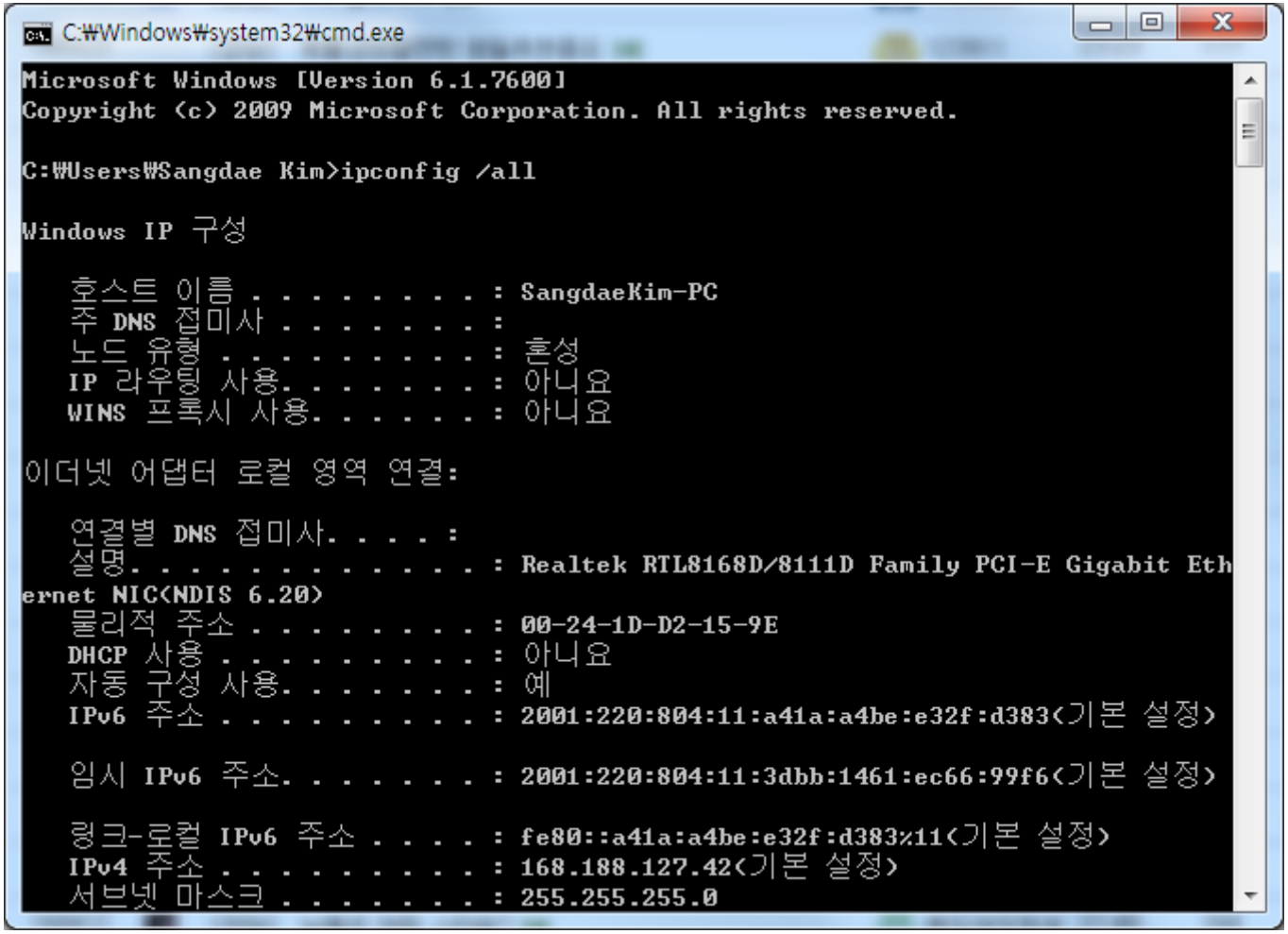
User Interface (cont.)

◈ Example



Tips

- ipconfig /all
  - Can get the IP address and Ethernet Address of the Network Interface Card.



Tips (cont.)

- homework #2에 그대로 Layer만 삽입, 새로 구현 하지 말고 Workspace를 새로 작성하여 직접 UI도 그리면서 프로젝트 수행

- 평소에 과제를 수행할 때는 전용선을 사용하여 네트워크에 영향을 주지 않도록 주의
  - 전용선은 수업시간에 조당 1개씩 배당되며 데모 후 반납

- 다큐먼트 작성에 소홀하지 않도록 주의
- 과제 제출일
  - 1차 제출: 3월 31일/4월 1일 24:00 시까지
  - 최종 제출: 4월 14/15일 24:00 시까지
  - 데모: 4월 중

Tips (cont.)

◈ 제출 양식

❖ 파일명 예시

◆ [분반]조_hw과제번호_(코드/문서)

파일명 예시

- 00반 1조 1번째 과제 제출 :
  - 코 드 : [00]1_hw01_code.zip
  - 문 서 : [00]1_hw01_doc.doc (or .docx, .hwp)
  - 압축파일 : [00]1_hw01.zip (상기의 두 파일을 압축)
- 01반 3조 2번째 과제 제출 :
  - 코드 : [01]3_hw02_code.zip
  - 문서 : [01]3_hw02_doc.doc (or .docx, .hwp)
  - 압축파일 : [01]3_hw02.zip (상기의 두 파일을 압축)

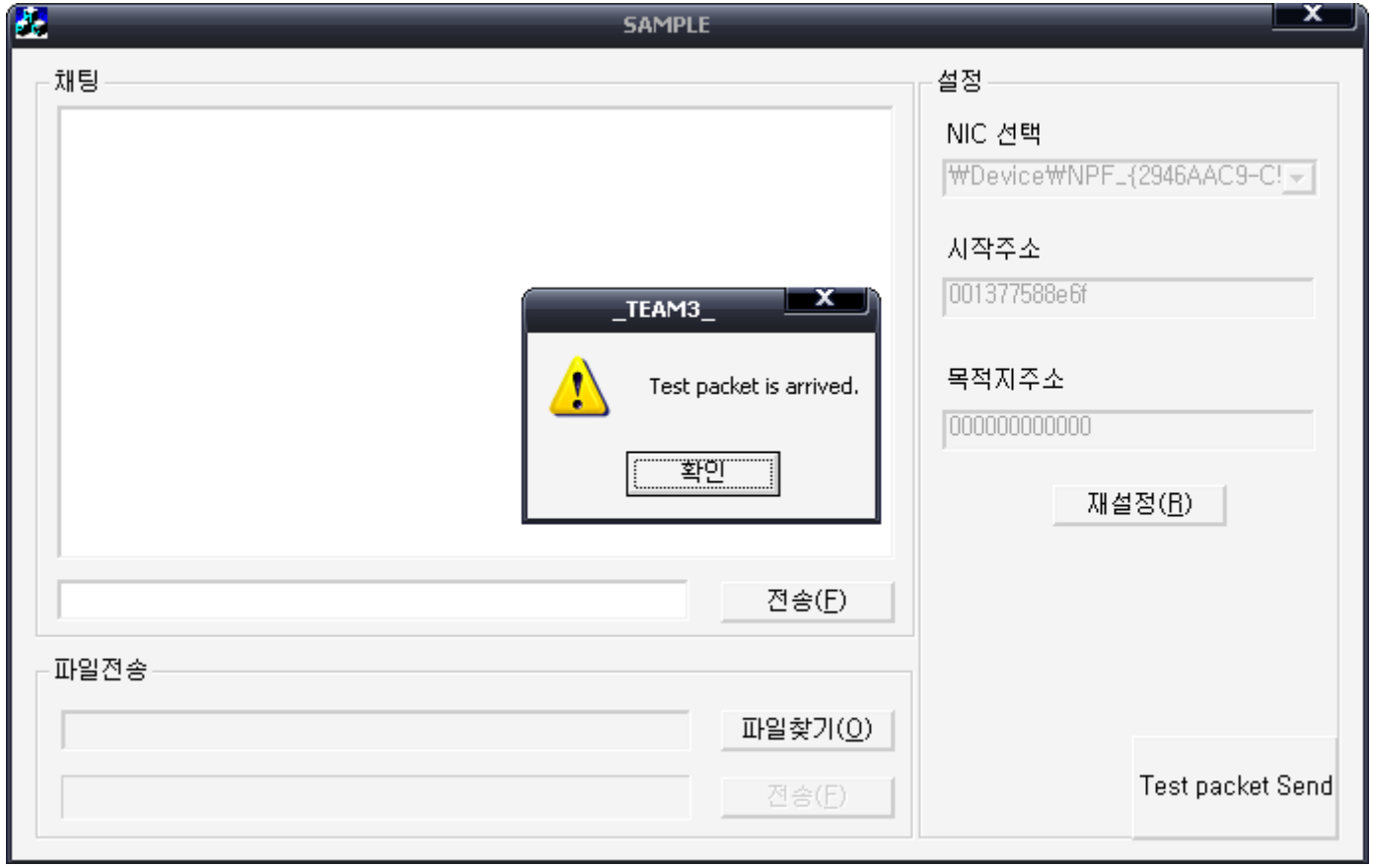
❖ 메일 제목 예시:

◆ [분반]3_hw02_조장

- 00반 1조 1번째 과제 제출 : [00]1_hw01_조장명
- 01반 3조 2번째 과제 제출 : [01]3_hw02_조장명

❖ Debug 제외한 나머지 코드 압축

1차 과제 (example.)



1차 과제

- Test packet 을 Ethernet 에 broadcast 하는 버튼을 구성한다.
- 패킷을 구분하기 위해 0x1234 type 을 받았을 때에 간단한 메시지를 확인하도록 구현. (Screen shot)
- 패킷 캡쳐 프로그램(wireshark등)을 사용하여 해당 패킷을 간단히 분석. (Screen shot)

- 기한
  - 00반 : 4월 1일 23시 59분
  - 01반 : 3월 31일 23시 59분

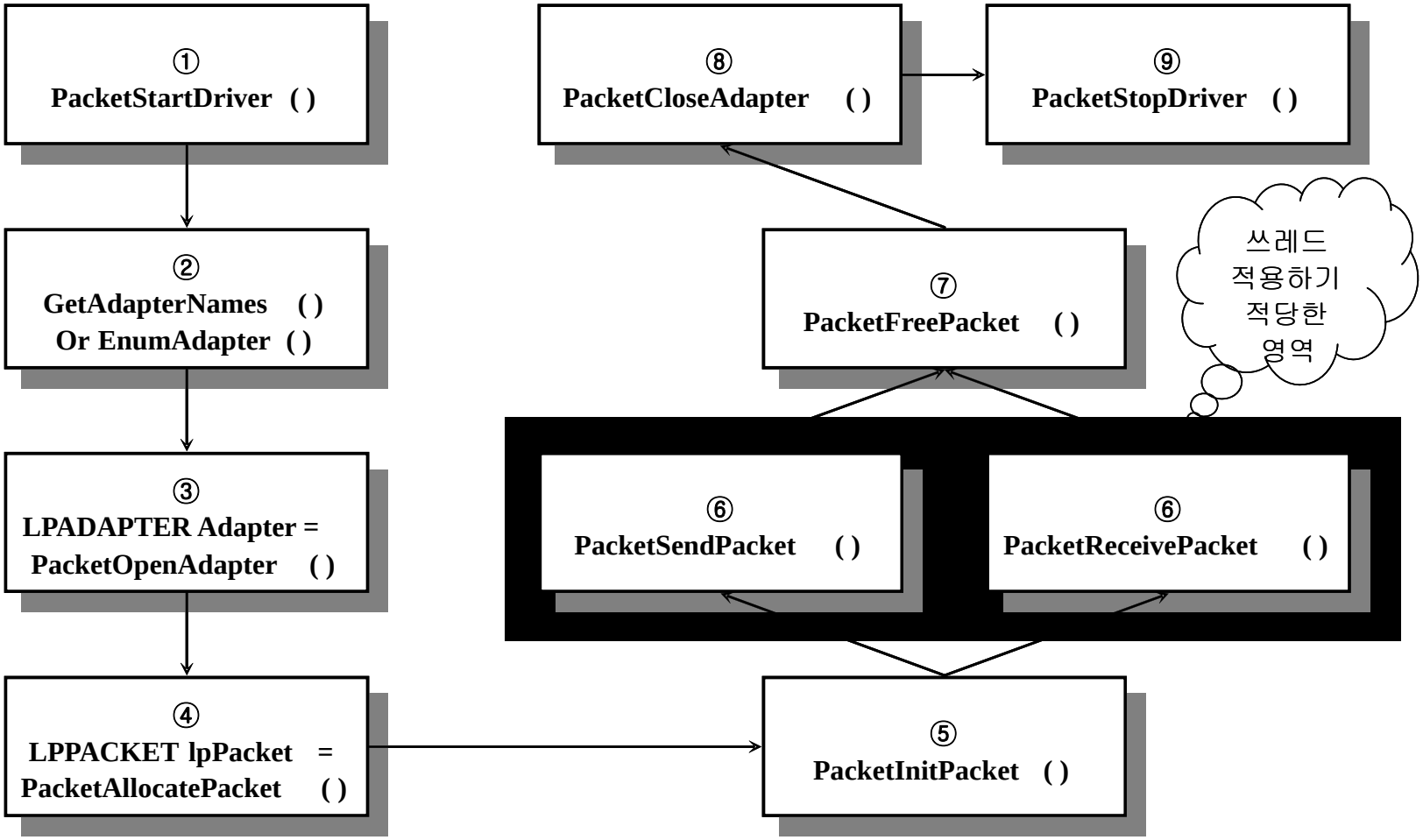
- 각 조별로 e-mail 송신할 것

- 스크린샷을 첨부하여 패킷 분석과 조별 진행 상황을 작성하여 보낼 것

- Filename example
  - [01]3_hw02_screenshot.doc (01반 3조일 경우)

Appendix

Appendix


① PacketStartDriver ( )
② GetAdapterNames ( ) Or EnumAdapter ( )
③ LPADAPTER Adapter = PacketOpenAdapter ( )
④ LPPACKET lpPacket = PacketAllocatePacket ( )
⑤ PacketInitPacket ( )
⑥ PacketSendPacket ( )
⑥ PacketReceivePacket ( )
⑦ PacketFreePacket ( )
⑧ PacketCloseAdapter ( )
⑨ PacketStopDriver ( )
쓰레드 적용하기 적당한 영역

Appendix (cont.)

◈ Packet Driver API

- ❖ LPADAPTER PacketOpenAdapter(LPTSTR AdapterName)
  - ◆ NIC의 이름을 사용하여 초기화 한다.
  - ◆ Packet Driver에 사용에 필요한 ADAPTER의 핸들을 얻는다.
- ❖ VOID PacketCloseAdapter(LPADAPTER lpAdapter)
  - ◆ PacketOpenAdapter에서 얻은 핸들을 닫는다.
- ❖ PVOID PacketAllocatePacket()
  - ◆ Packet 송수신을 위해 Heap에 Packet 구조체를 메모리에 할당하고 Event를 설정한다.
- ❖ VOID PacketFreePacket(LPPACKET lpPacket)
  - ◆ PacketAllocatePacket에서 할당한 메모리와 Event를 해제한다.
- ❖ VOID PacketInitPacket(LPPACKET lpPacket, PVOID Buffer, UINT Length)
  - ◆ Packet 구조체의 버퍼 포인터 부분이 버퍼를 포인트 하도록 한다.
- ❖ BOOL PacketSendPacket(LPADAPTER AdapterObject, LPPACKET lpPacket, BOOLEAN Sync)
  - ◆ 설정한 ADAPTER로 Packet을 전송한다.

Appendix (cont.)

◈ Packet Driver API(cont.)

- ❖ BOOL PacketReceivePacket(LPADAPTER AdapterObject, LPPACKET lpPacket, BOOLEAN Sync, PULONG BytesReceived)
  - ◆ 설정한 ADAPTER로부터 Packet을 받는다.
- ❖ BOOL PacketWaitPacket(LPADAPTER AdapterObject, LPPACKET lpPacket, PULONG BytesReceived)
  - ◆ Async모드 수행에 대한 결과를 얻는다.
- ❖ BOOL PacketResetAdapter(LPADAPTER AdapterObject)
  - ◆ 설정한 ADAPTER를 Reset 한다.
- ❖ BOOL PacketRequest(LPADAPTER AdapterObject, BOOLEAN Set, PPACKET_OID_DATA OidData)
  - ◆ 커널계층의 정보를 Query/Setting 한다.
- ❖ BOOL PacketSetFilter(LPADAPTER AdapterObject, ULONG Filter)
  - ◆ 설정된 ADAPTER의 Filter를 변경한다.
- ❖ BOOLEAN PacketGetAddress(LPADAPTER lpAdapter, PUCHAR AddressBuffer)
  - ◆ 설정된 ADAPTER의 주소를 얻는다.

Appendix (cont.)

◈ WinPcap Driver API (디바이스&네트웍 정보)

- ❖ int pcap_lookupnet(char *device, bpf_u_int32 *netp, bpf_u_int32 *maskp, char *errbuf)
  - ◆ Network Device에 대한 Network 및 mask 번호를 되돌려준다.
  - ◆ Error가 발생할경우 -1 이 리턴되며, error 내용이 errbuf 에 저장된다.
- ❖ char *pcap_lookupdev(char *)
  - ◆ pcap_open_live() 와 pcap_lookupnet() 에서 사용하기 위한 Network Device에 대한 포인터를 되돌려준다.
  - ◆ 성공할 경우 "eth0", "eth1" 과 같은 이름을 되돌려주며 실패할 경우 0을 되돌려준다.
- ❖ int pcap_datalink(pcap_t *)
  - ◆ link layer 타입을 되돌려준다. (DLT_EN10MB 과 같은)

Appendix (cont.)

◈ WinPcap Driver API (패킷 캡쳐 초기화)

파일관련 작업을 할 때 file descriptor(파일지정자)를 이용해서 작업하는 것과 마찬가지로, 패킷 캡쳐 관련 작업을 할 때에도 packet capture descriptor 를 가지고 작업을 한다. packet capture descriptor 는 pcap_t * 형으로 선언되어 있다.

- ❖ pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms, char *ebuf)
  - ◆ 첫번째 인자로 주어지는 network device에 대한 packet capture descriptor을 만들기 위한 함수이다.
  - ◆ snaplen은 받아들일수 있는 패킷의 최대 크기(byte)이다.
  - ◆ promisc 는 network device를 promiscuous mode 로 할것인지를 결정하기 위해서 사용한다.
  - ◆ to_ms 는 읽기 시간초과(time out) 지정을 위해서 사용되며 millisecond 단위이다.
- ❖ pcap_t *pcap_open_offline(char *fname, char *ebuf)
  - ◆ fname 를 가지는 파일로부터 패킷을 읽어들인다. 만약 fname 이 "-" 일 경우 stdin으로부터 읽어들인다.
  - ◆ ebuf 는 에러메시지를 저장하기 위해서 사용된다.

Appendix (cont.)

◈ WinPcap Driver API (패킷 캡쳐(**Read**) 관련)

- ❖ u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)
  - ◆ NEXT 패킷에 대한 포인터를 리턴한다.
  - ◆ 우리는 이 패킷을 읽음으로써 패킷의 정보를 얻어올수 있다. 실지로 이 함수를 이용해서 패킷캡쳐와 관련된 모든 일을 할수 있다.
- ❖ int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
  - ◆ p 는 PCD 이며, cnt 는 패킷 캡쳐를 몇번에 걸쳐서 할것인지를 결정하기 위해서 사용한다.
  - ◆ 만약 0이 지정되면 계속 패킷을 받아들이게 된다.
- ❖ int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
  - ◆ pcap_loop 와 거의 비슷하다.

Appendix (cont.)

◈ WinPcap Driver API (패킷 필터링 관련)

- ❖ int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)
  - ◆들어오는 패킷을 필터링 해서 받아들이기 위해서 사용한다
- ❖ int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
  - ◆pcap_compile 을 통해서 지정된 필터를 적용시키기 위해서 사용되며, 앞으로 들어오는 패킷에 대해서는 이 필터룰에 의해서 필터링 된다.

Appendix (cont.)

◈ WinPcap Driver API (기타 필요한 함수)

- ❖ int pcap_findalldevs (pcap_if_t **alldevsp, char *errbuf)
  - ◆ 시스템에서 사용가능한 interface list를 return.
- ❖ int pcap_sendpacket (pcap_t *p, u_char *buf, int size)
- ❖ int pcap_setbuff (pcap_t *p, int dim)
  - ◆ 커널버퍼의 크기를 설정.