Data Communication (CE14773)

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

Sangdae Kim - 00반 / Cheonyong Kim- 01반

Overview

- Network Overview
  - Network Architecture (protocol stack)
  - Encapsulation
  - Demultiplexing
  - Protocol Format
    - Protocol Stack Implement
- Inter-Process Communication Overview
  - IPC (inter-Process Communication)

Network Overview

Network Architecture (1/2)

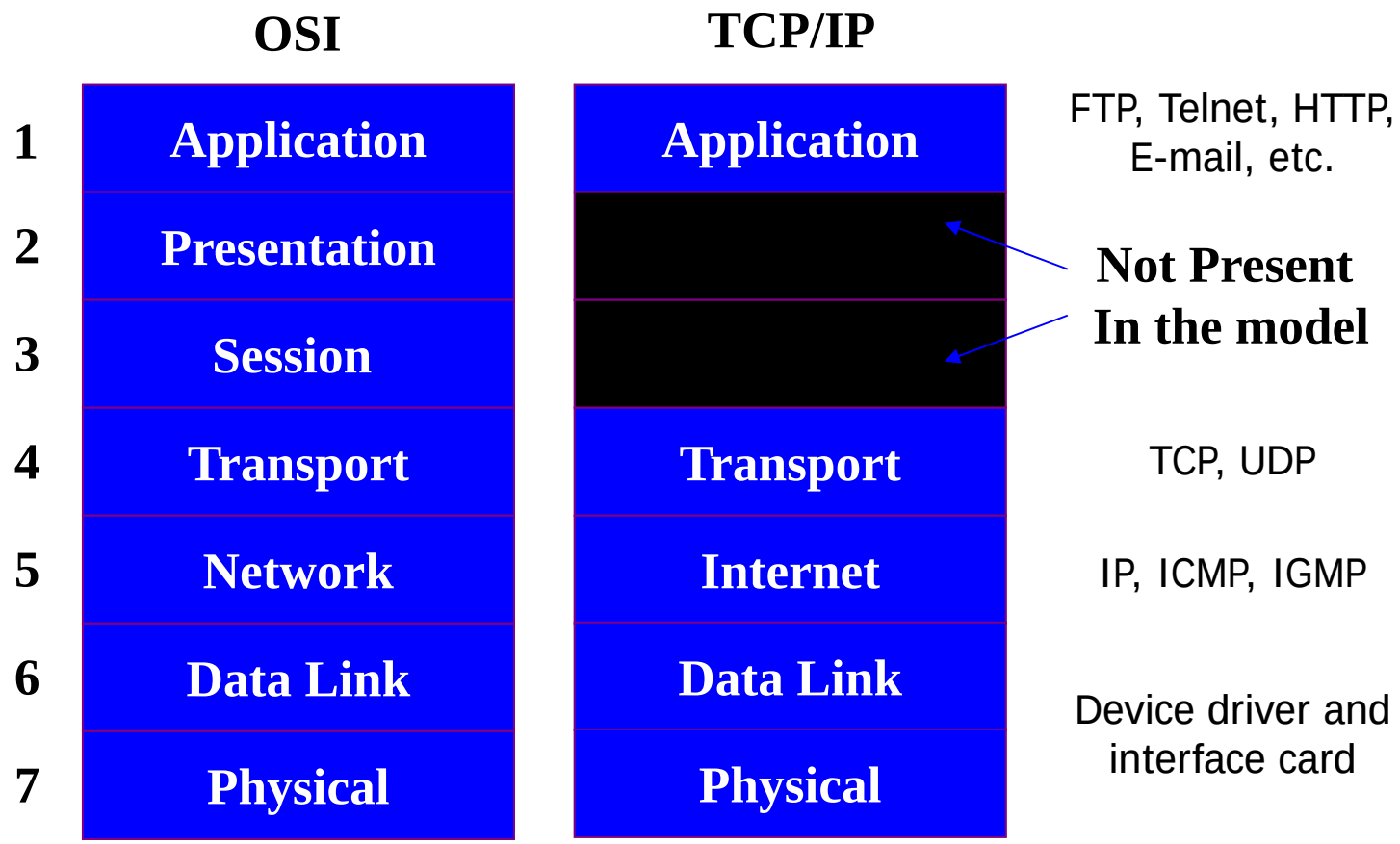


Figure 1_1. The OSI Reference Model & The TCP/IP Reference Model

Network Architecture (2/2)

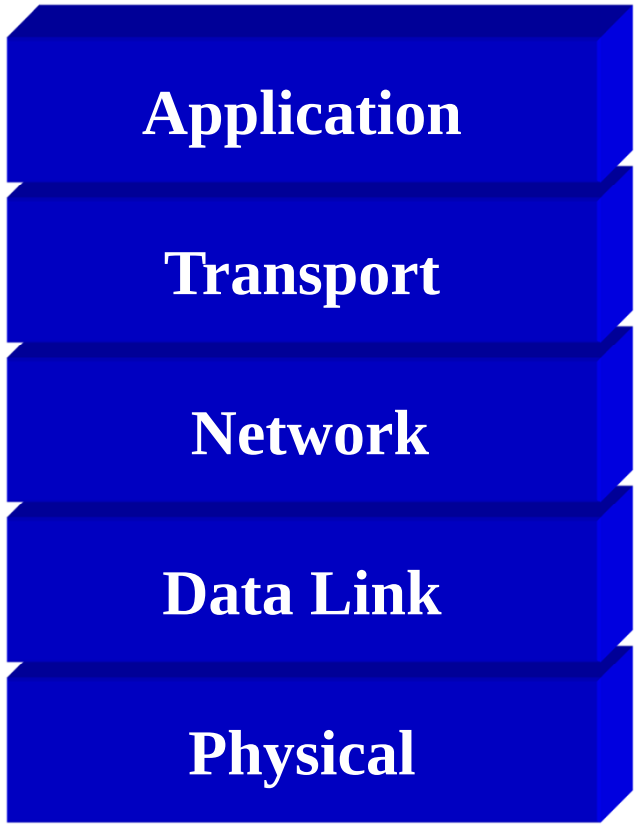


Figure 1_2. The Five Layers of the TCP/IP protocol suite

Encapsulation

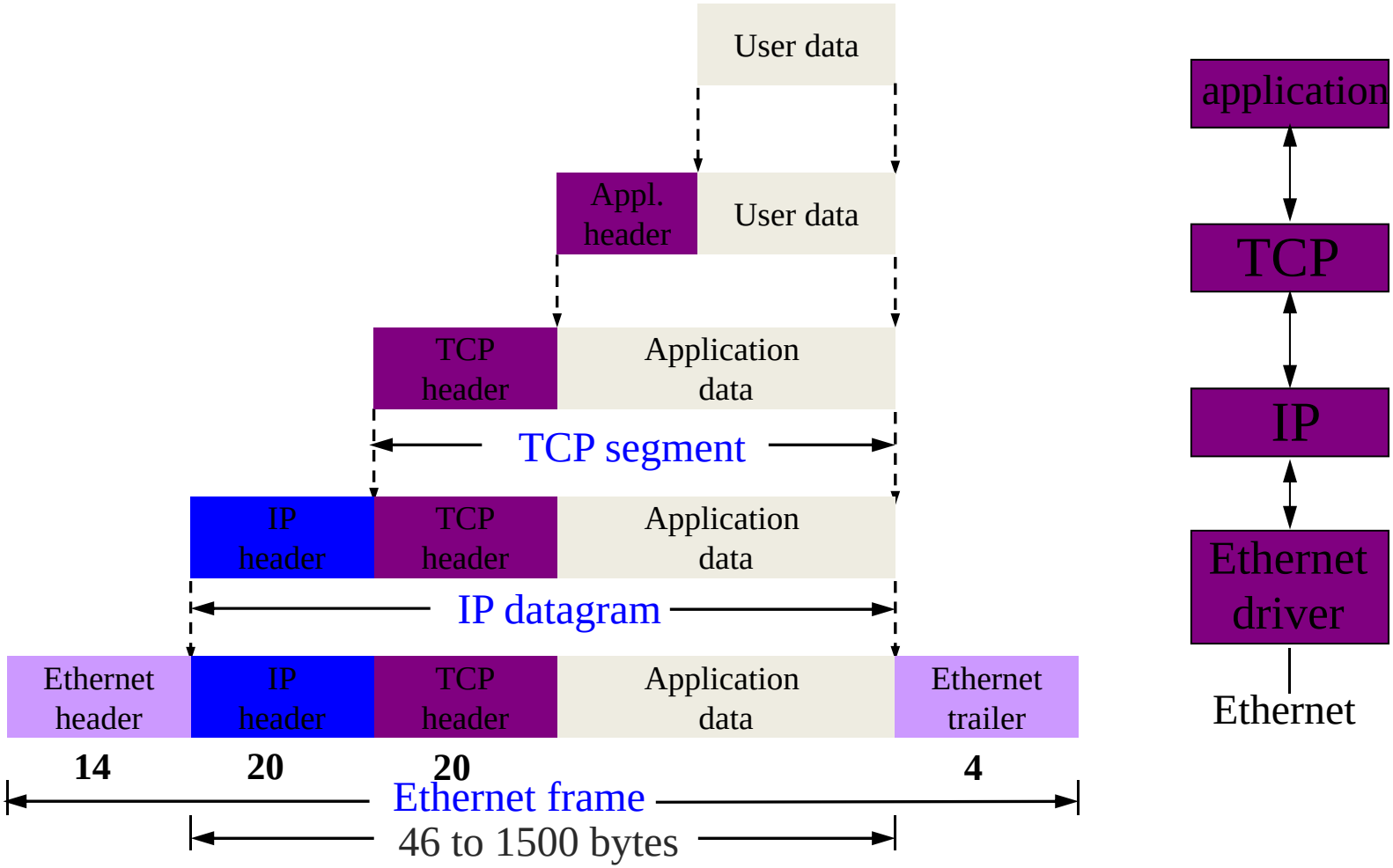


Figure 2. Encapsulation of data as it goes down the protocol stack

Demultiplexing

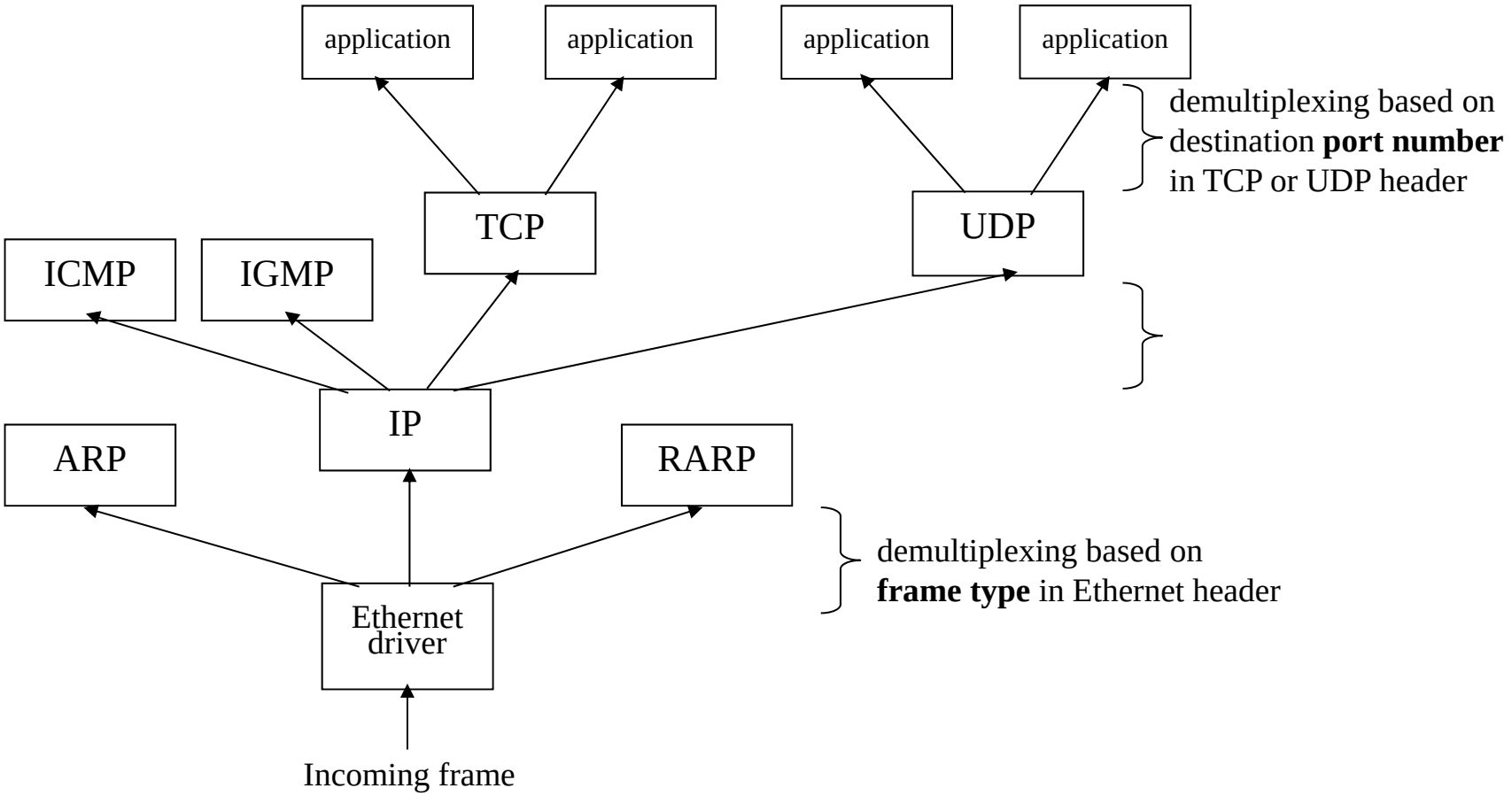


Figure 3. The demultiplexing of a received Ethernet frame

Protocol Format – IP

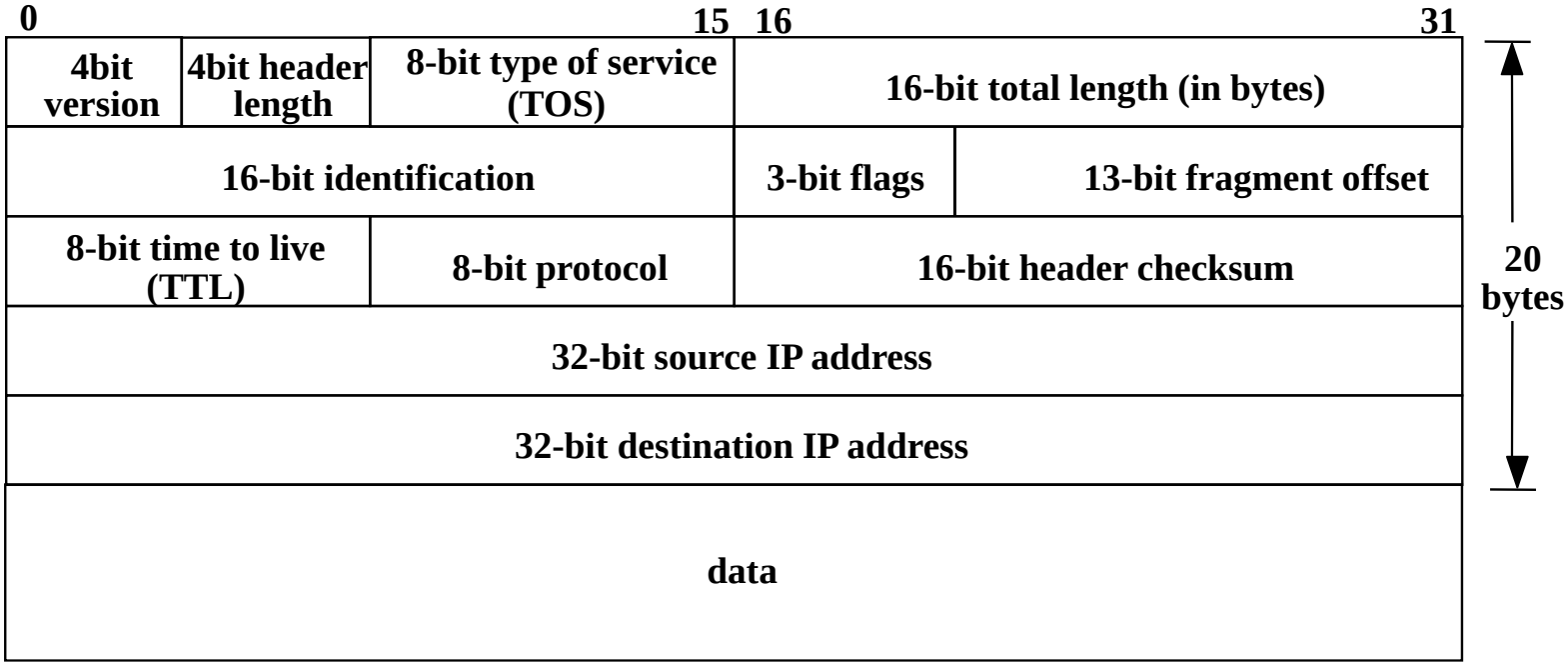


Figure 5. IP datagram, showing the fields in the IP header

Protocol Format – Ethernet (1/2)

Figure 4. Ethernet encapsulation (RFC 894)

- Implementation of protocol stack using C++ with Ethernet

```
struct _ETHERNET_HEADER {
        unsigned char       enet_dstaddr[ 6 ] ;
        unsigned char       enet_srcaddr[ 6 ] ;
        unsigned short      enet_type ;
        unsigned char       enet_data[ 1500 ] ;
}
```

Protocol Format – Ethernet (2/2)

```
#define ETHER_MAX_DATA_SIZE 1500
class CEthernetLayer : public CBaseLayer
{
public:
        BOOL            Receive( unsigned char* ppayload ) ;
        BOOL            Send( unsigned char* ppayload, int nlength ) ;
        void            SetDestinAddress( unsigned char* pAddress ) ;
        void            SetSourceAddress( unsigned char* pAddress ) ;
        unsigned char*  GetDestinAddress( ) ;
        unsigned char*  GetSourceAddress( ) ;

        CEthernetLayer( char* pName ) ;
        virtual ~CEthernetLayer() ;

        typedef struct _ETHERNET_HEADER {
            unsigned char   enet_dstaddr[6] ;
            unsigned char   enet_srcaddr[6] ;
            unsigned short  enet_type ;
            unsigned char   enet_data[ ETHER_MAX_DATA_SIZE ] ;
        } ETHERNET_HEADER, *PETHERNET_HEADER ;
protected:
        ETHERNET_HEADER         m_sHeader ;
};
```

Protocol Format – ARP

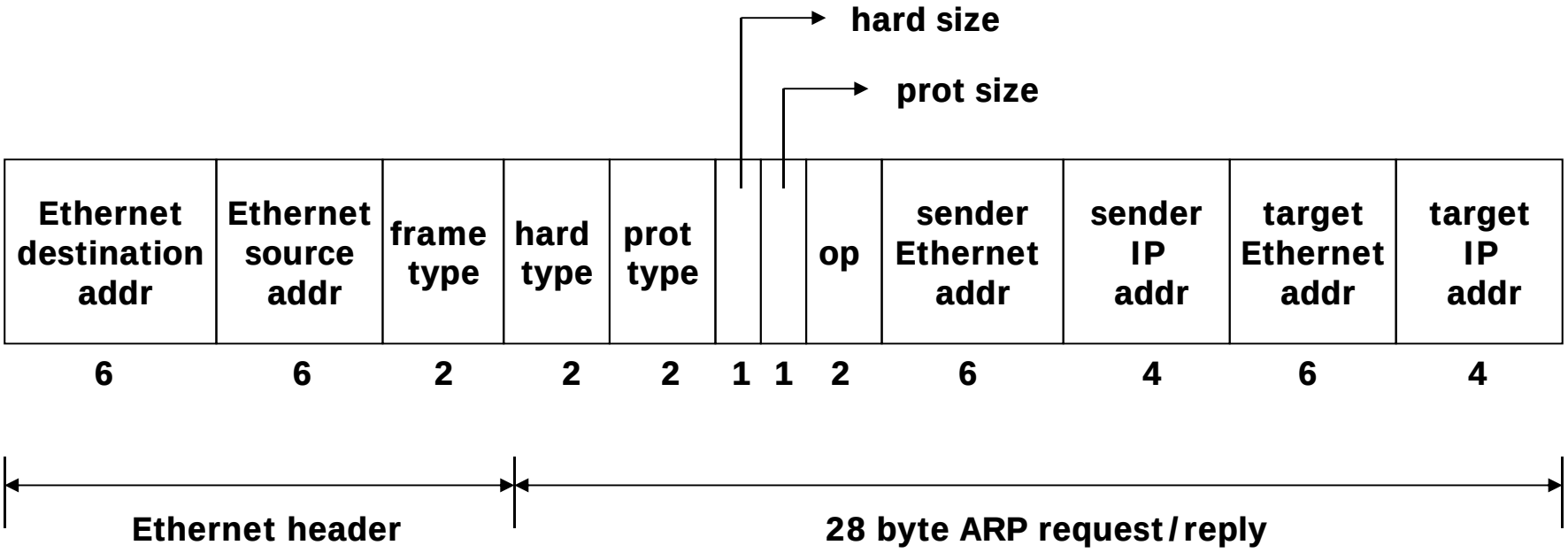


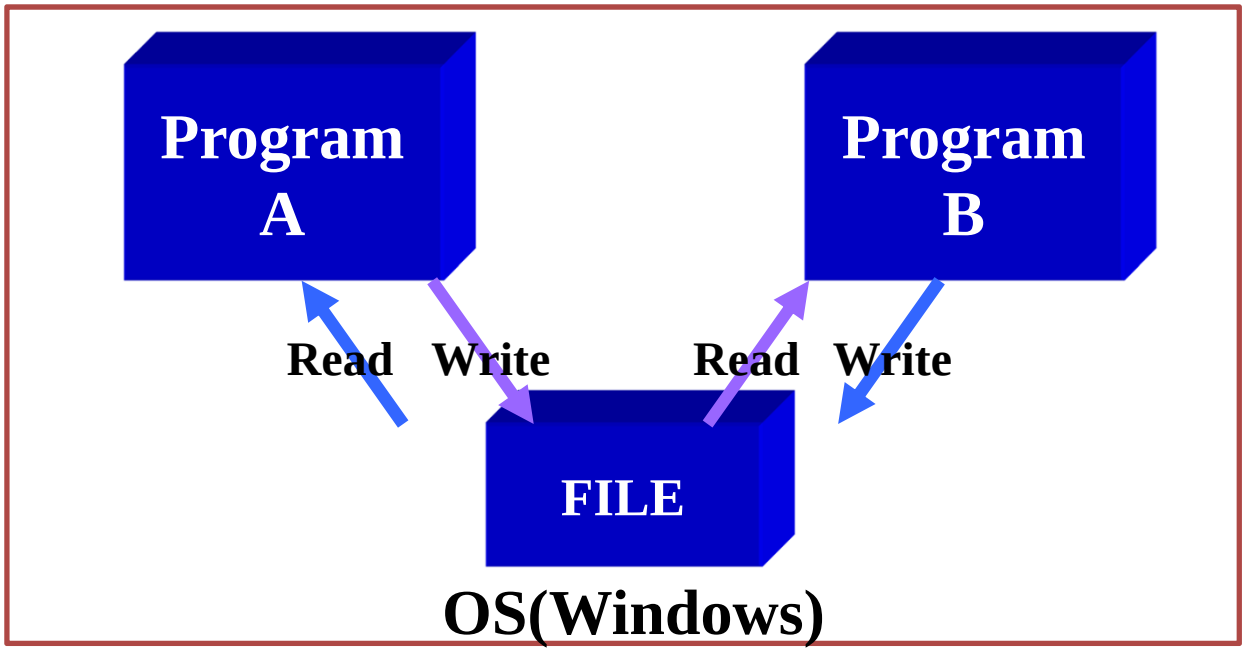
Figure 6. Format of ARP request or reply packet when used on an Ethernet

Inter-Process Communication Overview

Data Propagation by Using IPC

◈ Window상의 두 개의 프로그램(process)이 하나의 파일을 공유하여 통신을 하고 있다.

- ❖ Program B가 write동작을 완료하면, Program A가 read한다.
- ❖ Program A가 write동작을 완료하면, Program B가 read한다.



Data Propagation by Using IPC

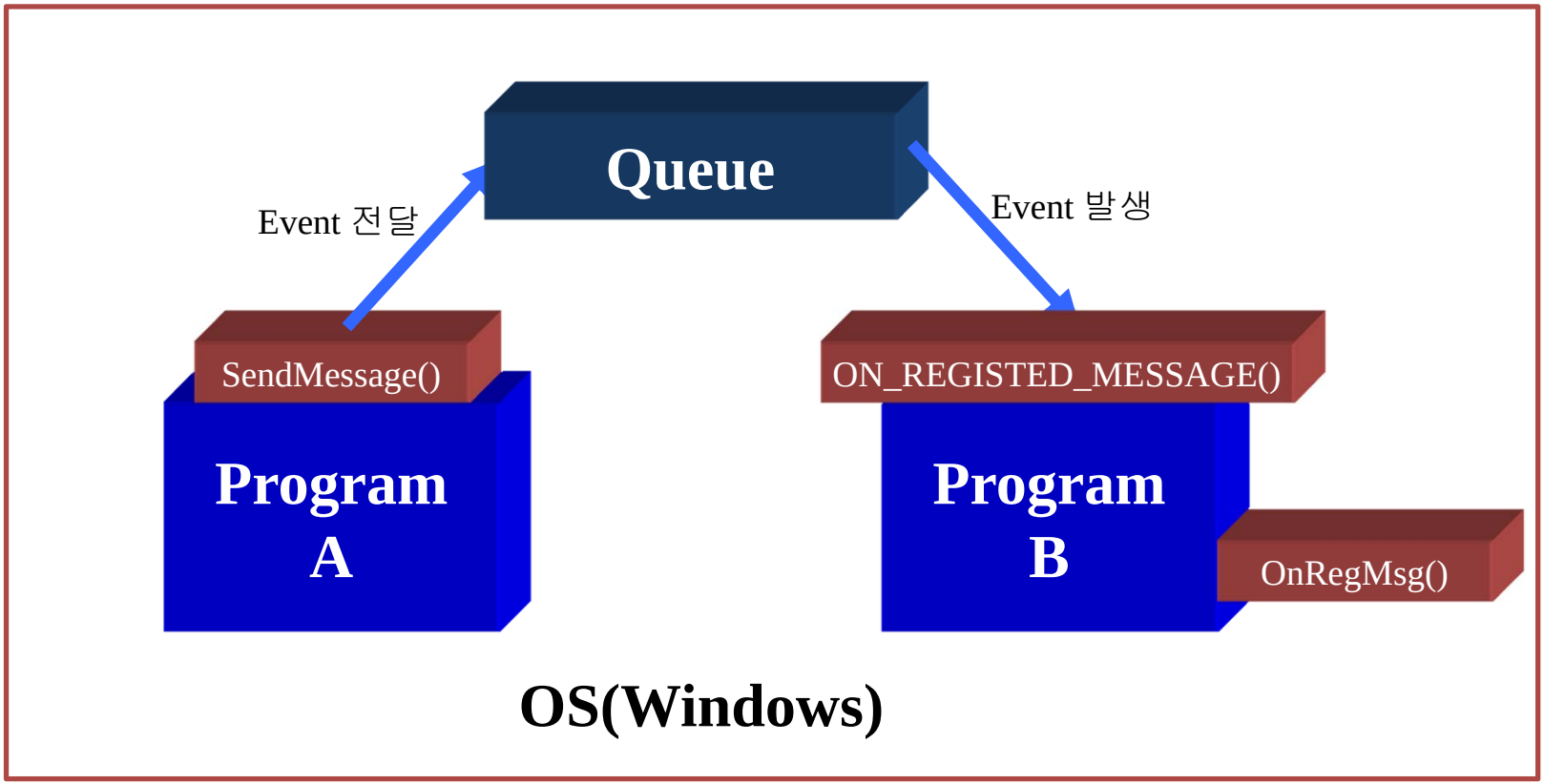
◈ 동기화 문제(Synchronization)

- ❖ Write가 끝나는 시점을 reading하는 쪽에 알려주어, write가 끝나면 read한다.
- ❖ We have a problem.
  - ◆ 각 process의 할당 메모리의 공유가 불가능하다.

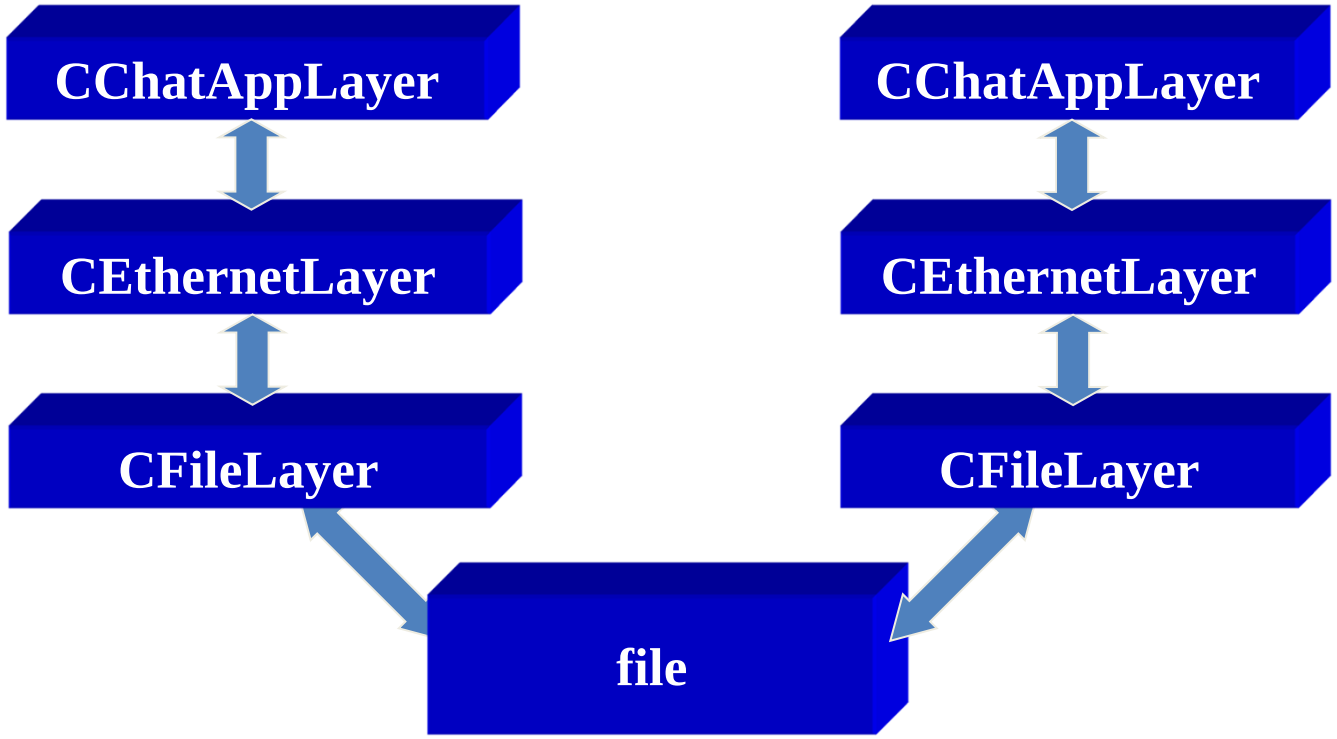
◈ Solution

- ❖ IPC (InterProcess Communication) 사용
  - ◆ Shared Memory
  - ◆ Message queue
  - ◆ PIPE
- ❖ 실제로 과제에서는 Message queue방식의 Register를 사용한다.

IPC – Message Queue



Hierarchical Architecture



Homework #2

◈ 과제 개요

process A의 사용자로부터 입력 받은 메시지를 process B의 메시지 창에 출력한다.

◈ 과제 세부 사항

❖ 내 Application과 상대Application 주소 설정

◆ 각 프로그램은 고유한 프로그램 ID를 가지고 있다.

❖ Process ID의 세부 사항

◆ process ID는 32bits의 크기(unsigned interger)를 가지고 있다.

❖ 메시지 전송을 레이어 상의 메시지형식은 다음과 같다.

◆ source ID(4bytes), destination ID(4bytes), message length(2bytes), message type(1byte), message data(? bytes)

❖ 메시지 수신

◆ 메시지는 상대방이 자기주소로 보낸 메시지와 broadcast message 만을 수신하며 나머지는 discard한다.

◈ 제출일

❖ 00 - 3/25 (수) 24:00, 01 - 3/24 (화) 24:00

❖ 00 - **CClabDC00@gmail.com**

❖ 01 - **CClabDC01@gmail.com**

Homework #2(cont.)

- Protocol Stack을 사용한 메시지 송신
  - Protocol stack

    CChatAppLayer –> CEthernetLayer –> CFileLayer

    - CChatAppLayer :메시지를 UI로 부터 얻은 정보를 바탕으로 Ethernet Layer에게 App 자료 구조 전체를 전달한다.
    - CEthernetLayer : 메시지 자료구조를 Ethernet의 데이터 부분에 저장하여 Ethernet자료구조 전체를 CFileLayer로 전송한다.
    - CFileLayer : CEthernetLayer로 전송 받은 데이터를 파일로 기록한다.

Homework #2(cont.)

- Protocol Stack을 사용한 메시지 수신
  - Protocol stack

    CChatAppLayer <– CEthernetLayer <– CFileLayer

    - CChatAppLayer : CEthernetLayer로부터 받은 App 자료구조를 바탕으로 UI로 전달할 데이터를 구분한다.
    - CEthernetLayer : File Layer 로부터 받은 자료구조를 중에서 App 자료구조에 해당하는 부분을 CChatAppLayer로 전달한다.
    - CFileLayer : File로 부터 동기화를 통해 전달되 메시지를 받아 파일의 데이터를 읽어서 Ethernet으로 데이터를 전달한다.

Homework #2(cont.)

- 동기화(synchronization)
  - 데이터 전송을 위해 하나의 파일을 두 개의 프로그램이 공유하고 있다. 신뢰성 있는 동작을 위해 동기화 메커니즘을 사용한다.
  - 송신측
    - 송신측은 파일에 기록을 완료하고 기록한 것을 수신측에게 알려준다.
    - 수신측으로부터 응답 메시지가 올 때까지 2초간 응답이 없으면 응답이 없음을 알리고 메시지를 출력하고 다시 입력을 받는다. 응답 받는 동안은 재입력을 받지 않는다.
    - 수신측으로부터 결과가 돌아오면 화면에 표시한다.
  - 수신측
    - 송신측으로부터 연락을 받고 CFileLayer를 통하여 파일을 읽는다.
    - 정상적으로 파일을 읽어 화면에 출력하였으면 송신측에게 정상적으로 동작하였다고 송신측에 알려준다.

Homework #2(cont.)

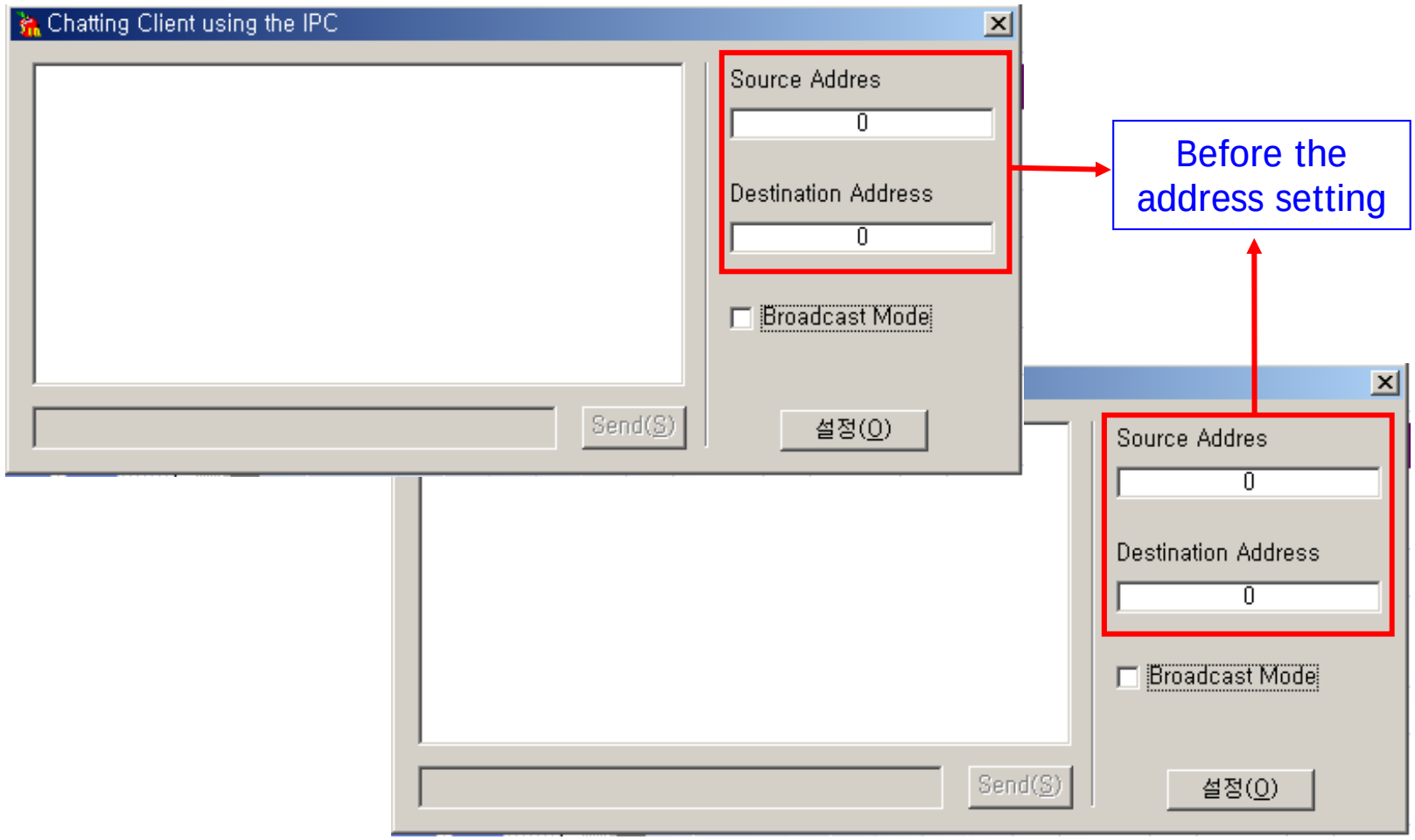
- UI
  - 프로그램 화면에 나타나야 할 정보를 정의한다.
    - Source / Destination Address, Message etc.
  - 메시지 출력 방식
    - 1번 process에서 2번 process로 정상적인 메시지 전송이 이루어진 경우 : [1:2]입력메시지

      Ex.) 안녕하세요. -> "[1:2] 안녕하세요."
    - 1번에서 broadcasting 한 메시지일 경우 :

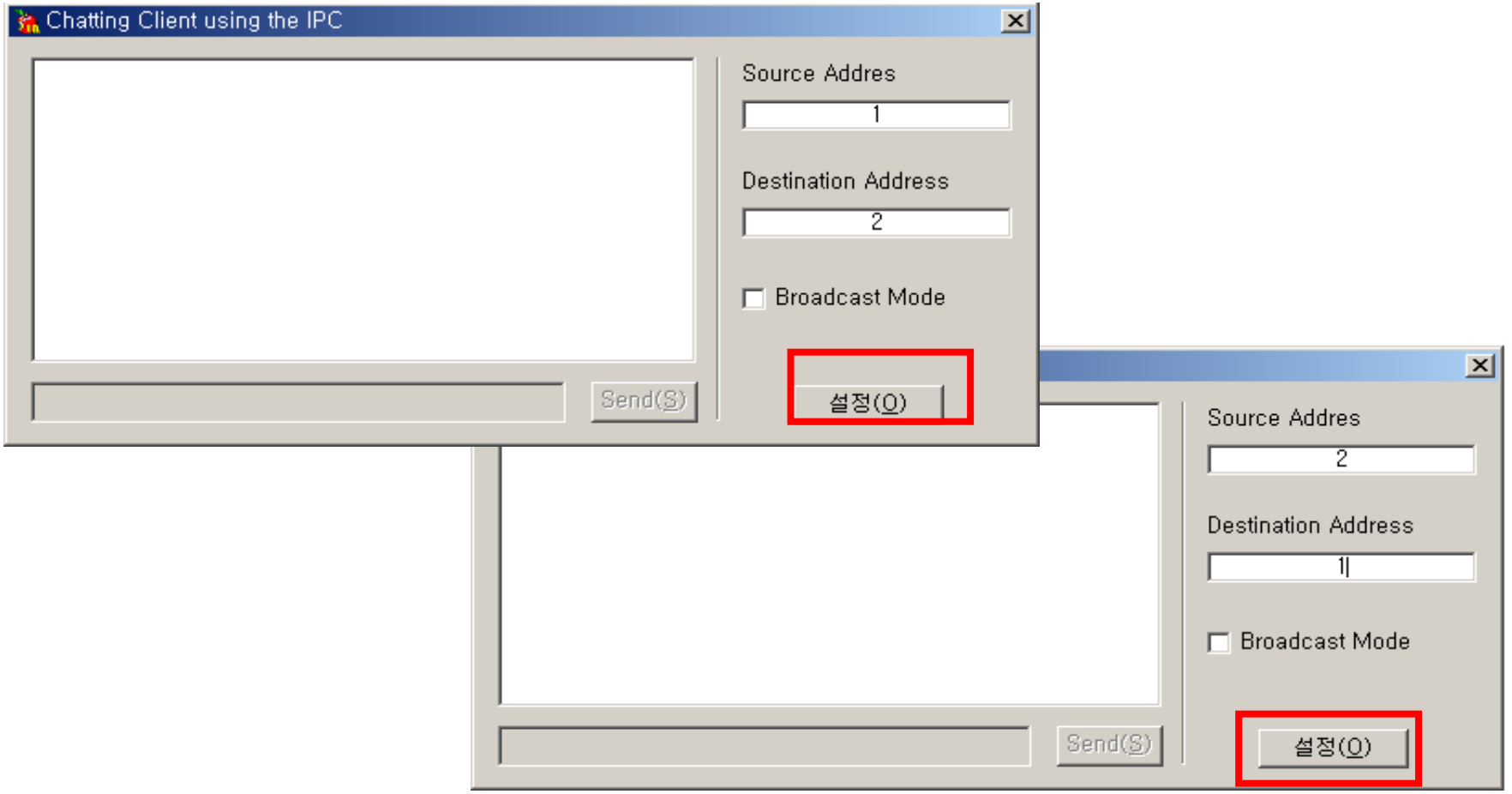
      Ex.) 안녕하세요. -> "[1:BROADCAST] 안녕하세요."
    - Broadcast Message 경우에는 응답메시지가 하나만 만 도착해도 올바른 수신이 된 것으로 인식
  - 응답이 없는 경우 : 메시지 출력 후 "time-out" message를 다시 출력한다.

    Ex.) 안녕하세요. -> "[1:2]안녕하세요." 출력 후 "time-out"을 다시 출력.

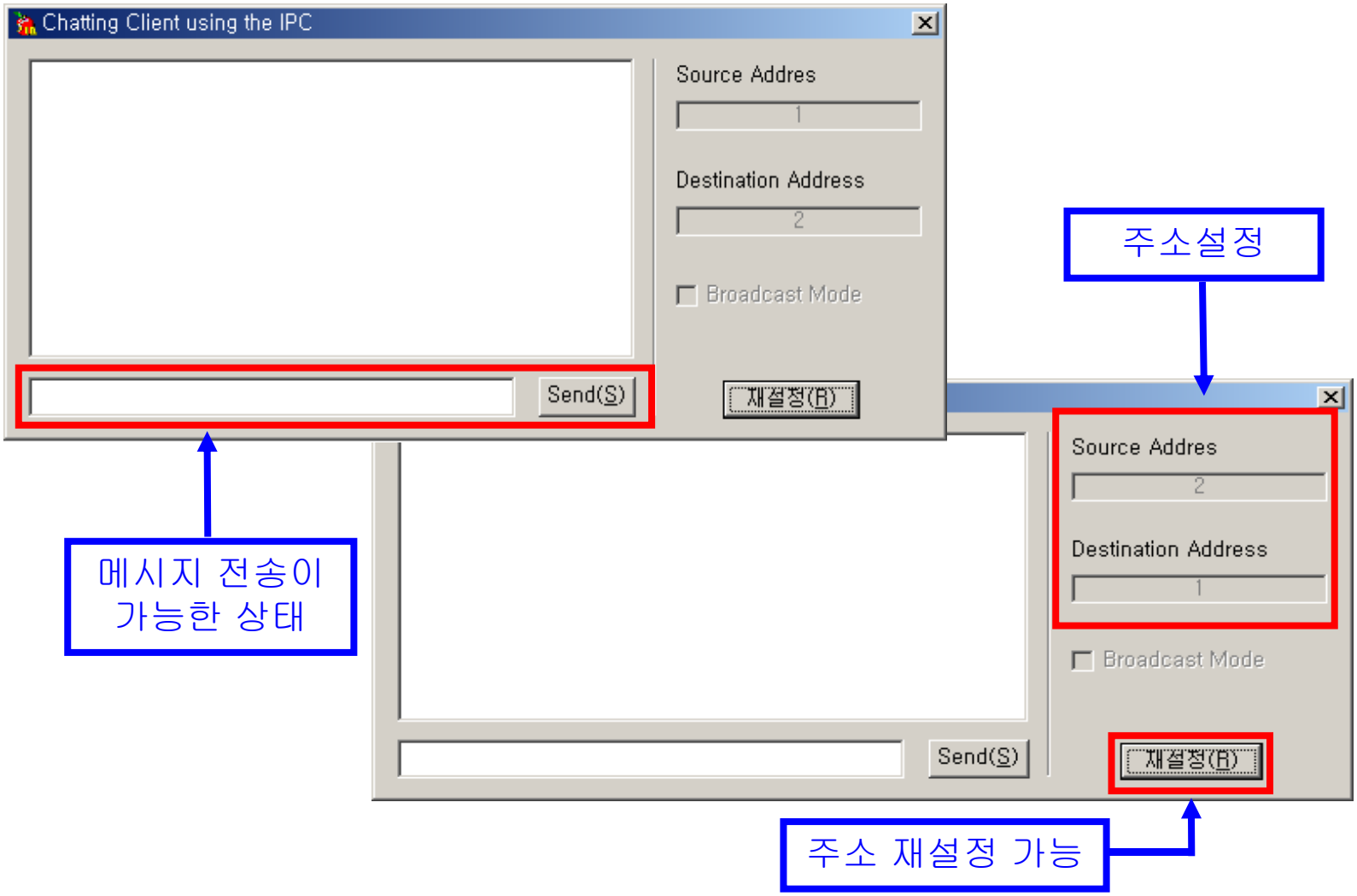
User Interface example



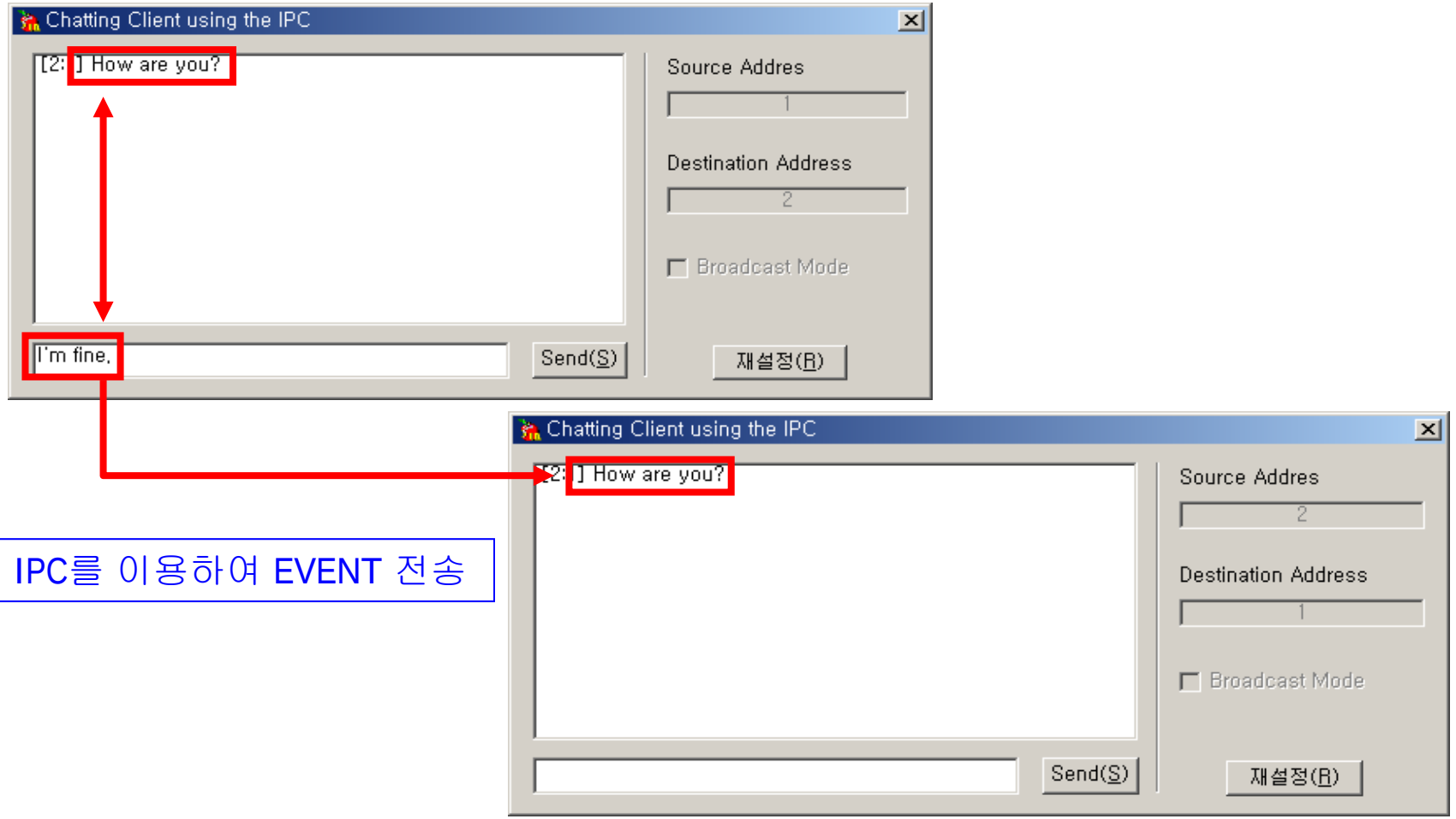
User Interface example



User Interface example



User Interface example



IPC를 이용하여 EVENT 전송

HW#2 Architecture