Data Communication (CE14773)

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

Sangdae Kim - 00반 / Cheonyong Kim- 01반

Contents

- Data Fragmentation
  - Data Fragmentation concept
  - Data Fragmentation in Chatting & File transfer
    - Data Fragmentation in CFT
    - Data Structure
    - Operation Process

Data Fragmentation

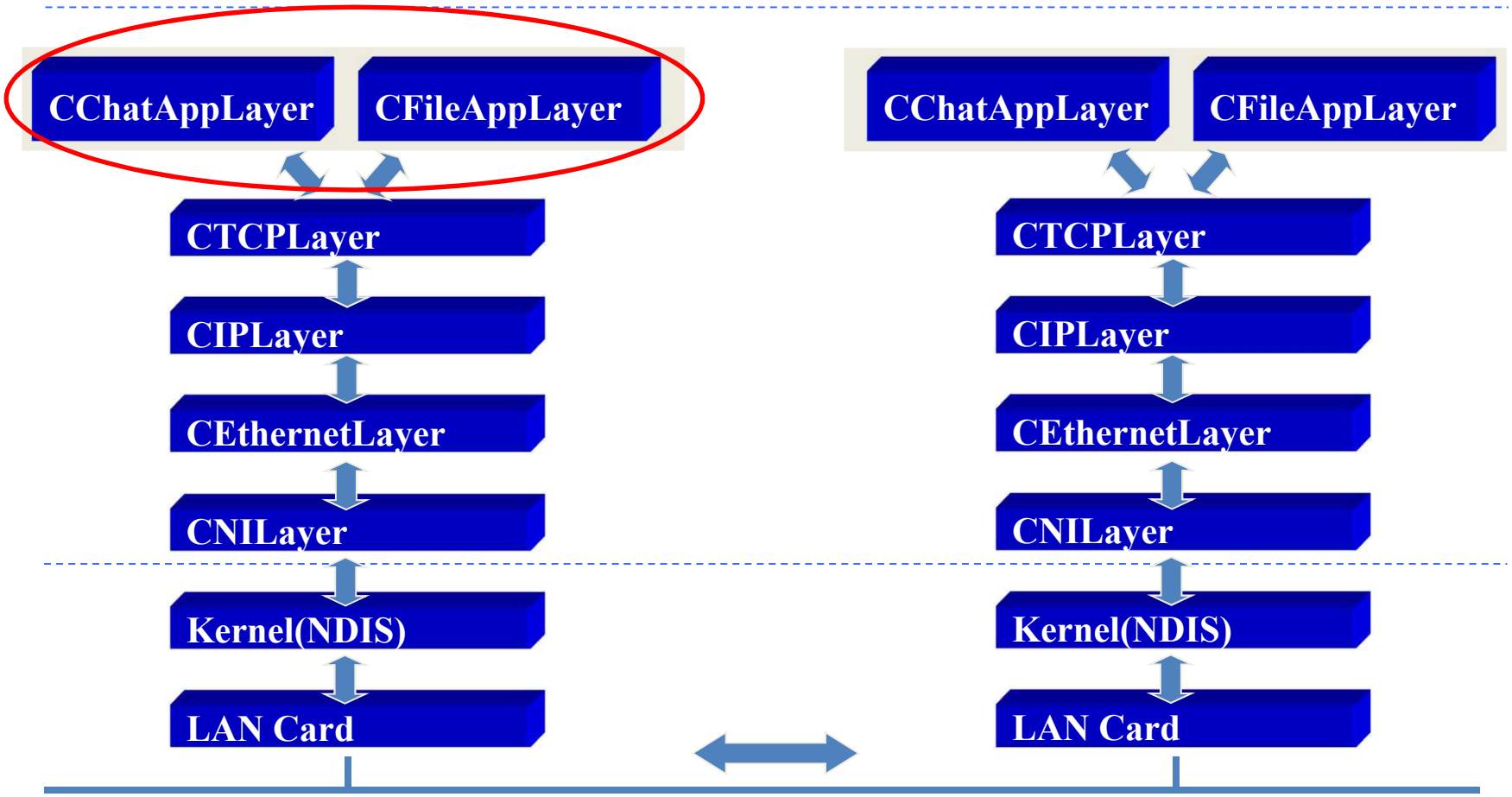
Data Fragmentation Concept

- Concept
  - 최대 전송단위 이상의 크기를 가진 데이터를 보낼 때 필요
  - 예를 들어, 이더넷 프레임(frame)의 최대 크기는 1500bytes. 3000bytes의 크기를 가진 데이터를 보내기 위해서는 1500bytes씩 두 개로 나누어 전송해야 됨
  - 이번 Chatting & File Transfer에서는 채팅, 파일 전송에 활용할 수 있음

- Consideration
  - 메시지 종류 (데이터 송신 메시지, 데이터 수신확인 메시지, 수신 에러 메시지 등)
  - 메시지 순서 (단편화된 데이터들을 순서)
  - 원본 데이터의 크기(데이터들이 모두 잘 전송되었는가?)

Data Fragmentation in CFT

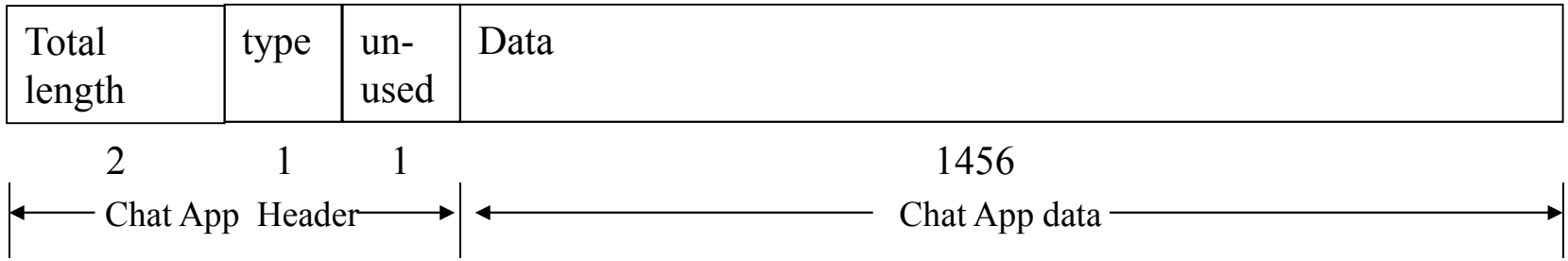
- Layer Model



Data Structures (Chat layer)

◈ 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 ;
```



◈ Chatting layer에서 사용되는 구조체

- ❖ totlen에 사용자가 입력한 문자열의 길이를 저장
- ❖ type에서 단편화에 대한 정보를 담을 수 있음
  - ◆ ex) 0x00 – 단편화 되지 않음, 0x01 – 단편화 첫부분, 0x02 – 단편화 중간, 등등..
- ❖ data에는 사용자가 입력한 문자열을 저장
  - ◆ 1456bytes까지 하나의 패킷에 넣을 수 있음

Operation Process (Chat layer)

◈ 송신

① 입력된 채팅의 길이를 측정

② 입력된 채팅의 길이가 1456bytes이하라면 단편화 없이 전송

③ 입력된 채팅의 길이가 1456bytes초과이라면 단편화

④ 첫 번째 조각에 채팅 전체에 대한 정보를 담아서 전달

- ◆ 전체 길이

⑤ 입력된 채팅을 1456bytes씩 잘라서 보냄.

- ◆ 이때 type을 통하여 처음, 중간, 끝을 구분해야 함.

Operation Process (Chat layer)

◈ 수신

① 전달받은 첫 조각을 통해 단편화 여부를 확인

◆ 전체 길이를 통해 1456byte 이상이라면 단편화가 되어있을 것임

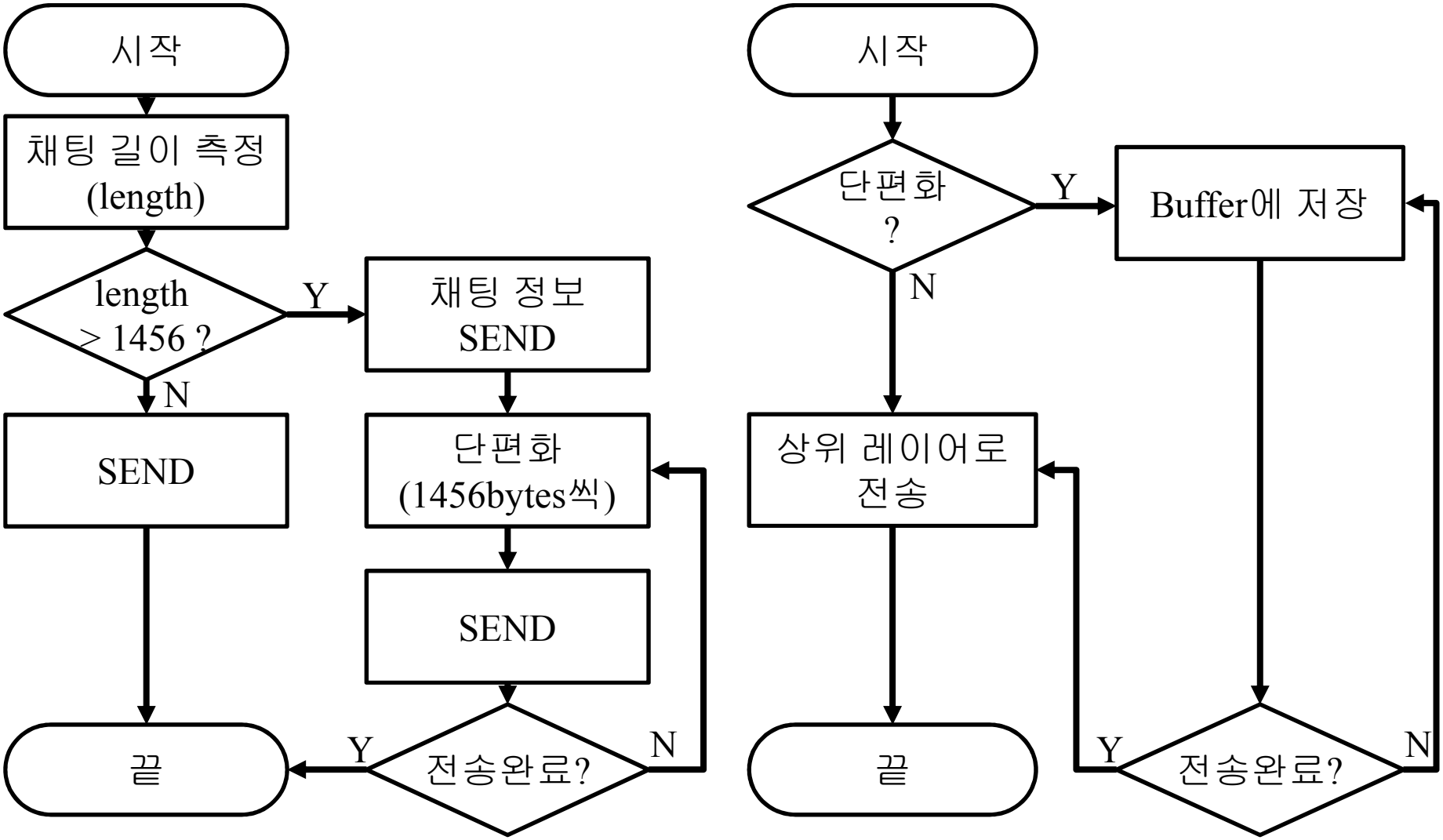
② 단편화가 되어있지 않다면, 그대로 상위 레이어로 전송

③ 단편화가 되어있다면, buffer를 활용하여 채팅이 모두 전달될 때까지 덧붙임

④ 모든 조각을 전달받았다면 buffer에 쓰여진 내용을 상위 레이어로 전송

◆ type을 통하여 모든 조각을 전달받았는지 확인할 수 있음

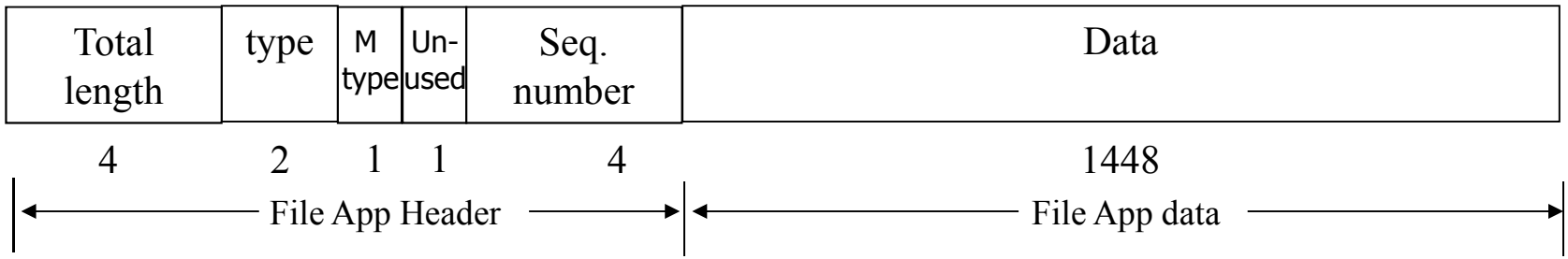
Process Flow Chart (Chat layer)



Data Structures (File layer)

◈ 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 ;
```



◈ File layer에서 사용되는 구조체

- ❖ totlen에 사용자가 선택한 파일의 길이를 저장
- ❖ type에서 단편화에 대한 정보를 담을 수 있음
  - ◆ ex) 0x00 – 단편화 되지 않음, 0x01 – 단편화 첫부분, 0x02 – 단편화 중간, 등등..
- ❖ seq_num에서 이 단편화된 조각이 몇 번째 조각인지 저장
- ❖ data에는 사용자가 선택한 파일을 저장
  - ◆ 1448bytes까지 하나의 패킷에 넣을 수 있음

Operation Process (File layer)

◈ 송신

① 선택된 파일의 길이를 측정(GetLength()함수 등)

② 선택된 파일의 길이가 1448bytes이하라면 단편화 없이 전송

③ 선택된 파일의 길이가 1448bytes초과이라면 단편화

④ 첫 번째 조각에 파일 전체에 대한 정보를 담아서 전달

- ◆ 전체 길이

⑤ 선택된 파일의 1448bytes씩 잘라서 보냄.

- ◆ 1448bytes만큼 파일에서 읽어와서 구조체의 data에 저장
- ◆ 이때 type을 통하여 처음, 중간, 끝을 구분해야 함.

⑥ 모든 단편화된 조각을 보내면서 progress bar를 수정

- ◆ (보낸 조각 개수 / 단편화된 조각 총 개수) 등을 이용

Operation Process (File layer)

◈ 수신

① 전달받은 첫 조각을 통해 단편화 여부를 확인

◆ 전체 길이를 통해 1448bytes 초과라면 단편화가 되어있을 것임

② 단편화가 되어있지 않다면, 그대로 상위 레이어로 전송

③ 단편화가 되어있다면, buffer를 활용하여 채팅이 모두 전달될 때까지 덧붙임

◆ 대부분의 파일은 1448bytes를 상당히 초과할 것임

◆ linked list 등의 자료구조를 활용하여 단편화된 조각의 순서도 고려하여 저장해야됨

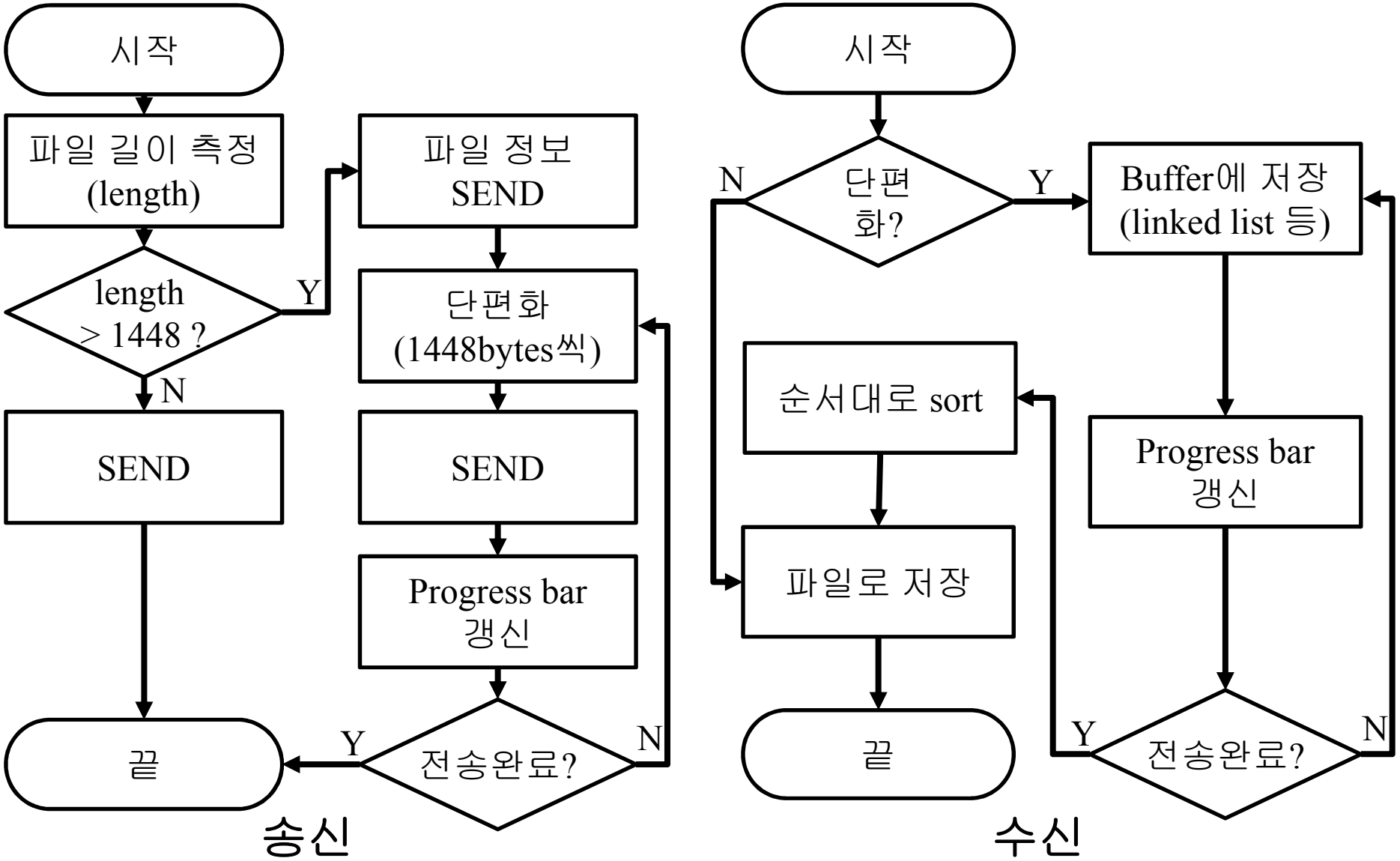
④ 모든 조각을 전달받았다면 buffer에 쓰여진 내용을 파일로 저장

◆ type을 통하여 모든 조각을 전달받았는지 확인할 수 있음

⑤ 단편화된 조각을 받으면서 progress bar를 수정

◆ (받은 조각 개수 / 단편화된 조각 총 개수) 등을 이용

Process Flow Chart (File layer)



Notice

- Chatting & File transfer demo
  - 18일 (토) 오후 1시 부터
  - 각 조별 30분씩 시연 및 검증
  - 개별 구두 테스트 (3문항)