정보통신
Stop and Wait protocol Simulation

2011270314
컴퓨터정보학과
서인석

목차

개요 (1/3)

프로젝트의 목적

- HDLC 의 Stop and Wait protocol을 구현하여 시뮬레이션환경에서, 에러(Frame, Ack)가 발생할 때 처리방법을 화면으로 보여주는 시스템

개요 (2/3)

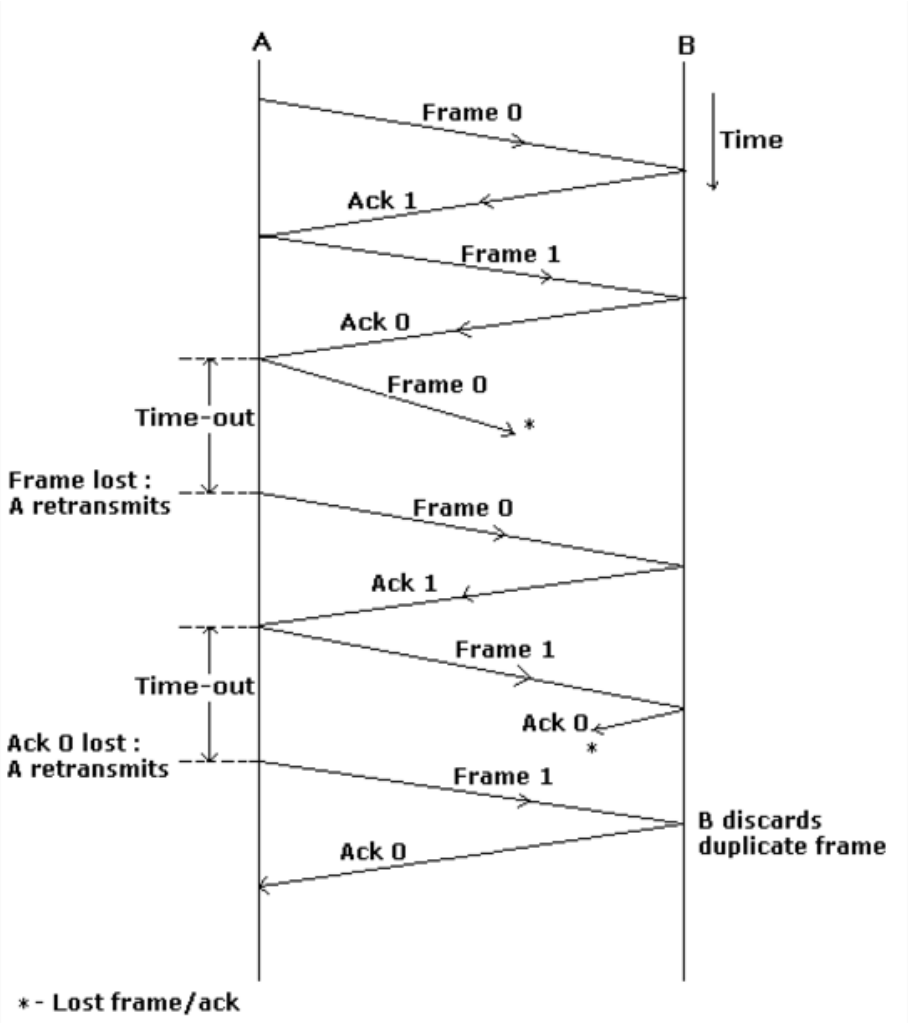
Stop and Wait 프로토콜

패킷이 손상되었거나 분실 되었을 경우 재전송하는 DLL의 오류제어 기법인 ARQ의 한 종류이다.

Stop and Wait 프로토콜의 특징

1) 타이머가 필요하다.
2) 손실 및 중복된 프레임을 구별하기 위해 순서번호가 필요하다.
3) 순서번호를 프레임과 같이 보낸다.
4) 지연이 큰 경우에 효율이 저하된다.

개요 (3/3)


A
B
Frame 0
Time
Ack 1
Frame 1
Ack 0
Frame 0
*
Time-out
Frame lost :
A retransmits
Frame 0
Ack 1
Frame 1
Time-out
Ack 0
*
Ack 0 lost :
A retransmits
Frame 1
B discards
duplicate frame
Ack 0
* - Lost frame/ack

구현 설명

C를 이용한 단일

Stop and Wait 시뮬레이션

JAVA를 이용한 소켓통신

Stop and Wait 시뮬레이션

구현 설명

C를 이용한 단일

Stop and Wait 시뮬레이션

구현 설명(C를 이용한 Stop and Wait)

함수 선언및 전역변수 선언

```c
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>

#define PACKET_NUM 5 // 프레임 길이 설정

typedef struct packet{
    int data;
}; // 패킷구조체 생성
typedef struct frame{
    int seq; // Sender에서의 순서번호(seq)
    int ack; // Receiver에서의 ack
    packet info; // 프레임내의 데이터
    int err; // 에러 발생을 보이기 위한 에러
}; // 프레임 구조체 생성

typedef enum{ARRIVAL, ERR, TIMEOUT} event_type; // 프레임도착, Time out, 에러 이벤트 정의

void FromNetworkLayer(packet *); // 네트워크층에서 데이터를 받음
void ToPhysicalLayer(frame *); // 프레임구성후, 물리층으로 전달
void FromPhysicalLayer(frame *); // 전송받은 데이터를 물리층에서 데이터링크층으로 전달
void ToNetworkLayer(packet *); // 프레임을 네트워크층으로 전달
void WaitSender(event_type *); // Sender에서의 Timer,에러,도착을 확인
void WaitReciever(event_type *); // Receiver에서의 도착및 에러 확인
void sender(); // Sender 정의
void reciever(); // Receiver 정의
int Random(int n); // 랜덤으로 오류를 발생하기위한 함수
int MakeSeq(packet *,int*); // Sender에서의 순서번호(seq)를 만들기위한 함수

int i = 0; // 전송하는 패킷의 번호
char turn = 's'; // Sender,Receiver의 순서
int Finish = 0; // 종료상태
frame DATA; // 전송되는 데이터
```

구현 설명(C를 이용한 Stop and Wait)

Sender 함수

```
void sender(){
    static int seq= 0; // 순서번호
    static frame s; // Sender에서의 프레이밍
    packet buffer; // 네트워크층에서 받은 패킷
    event_type event; // 위에서 정의한 3가지 이벤트
    static int flag = 0;

    if (flag == 0){ // 최초의 데이터 전송 한번 수행
        FromNetworkLayer(&buffer); // 네트워크층에서 패킷을 받음
        s.info = buffer; // 받은 데이터를 저장
        s.seq = seq; // 순서번호
        printf("SENDER :  Data[%d]   Seq[%d]        ", s.info, s.seq);
        turn = 'r'; // Receiver의 차례로 넘어가게 함
        ToPhysicalLayer(&s); // 물리층으로 데이터를 넘김
        flag = 1;
    }

    WaitSender(&event); // 어떤 이벤트가 발생하는지 정의

    if (turn == 's'){ // Sender의 차례
        if (event == ARRIVAL){ // 프레임이 도착. 이벤트 ARRIVAL발생
            FromNetworkLayer(&buffer); // 네트워크층에서 패킷을 받음
            MakeSeq(&buffer,&seq); // 받은 패킷을 이용하여 순서번호(seq)를 생성
            s.info = buffer; // 받은 데이터를 저장
            s.seq = seq; // 순서번호(seq) 저장
            printf("SENDER :  Data[%d]   Seq[%d]       ", s.info, s.seq);
            turn = 'r'; // Receiver의 차례로 넘어가게 함
            ToPhysicalLayer(&s); // 물리층으로 데이터를 넘김
        }

        else if (event == TIMEOUT){ // 이벤트 Timeout 발생
            printf("SENDER : Time Out! Resending Frame");
            turn = 'r'; // Receiver의 차례로 넘어가게 함
            ToPhysicalLayer(&s); // 물리층으로 다시 데이터를 넘김
        }
    }
}
```

구현 설명(C를 이용한 Stop and Wait)

Receiver 함수

```
void reciever(){
    static int ack = 1; // Receiver에서 보내는 ack
    frame r, s; // 받은 프레임, 다시 보낼 프레임 생성
    event_type event; // 이벤트

    WaitReciever(&event); // 어떤이벤트가 발생하는지 정의

    if (turn == 'r'){ // Receiver의 차례
        if (event == ARRIVAL){ // 프레임이 도착. 이벤트 ARRIVAL 발생
            FromPhysicalLayer(&r); // 물리층에서 데이터를 받음

            if (r.seq + 1 == ack || r.seq - 1 == ack){ // 받은 패킷의 순서번호가 전에 보낸 ack와 같은지 판단
                printf("   RECIEVER : Data[%d] Received Ack[%d] Send₩n", r.info, ack);
                ToNetworkLayer(&r.info); // 네트워크층으로 데이터를 전송

                if (ack < 1)
                    ack++;
                else
                    ack = 0; // ack가 0또는 1이 되도록 함.
            }

            else
                printf("   RECIEVER : Ack Resent(Discard Frame)₩n");

            turn = 's'; // Sender 차례로 바꿈
            ToPhysicalLayer(&s); // ack를 전송하기위해 물리층으로 데이터를 보냄
        }

        if (event == ERR){ // 에러가 발생했을때
            printf("   RECIEVER : Frame Loss!₩n");
            turn = 's'; // 프레임이 중간에 잃어버렸다면 차례를 Sender로 바꿔줘서 다시 보내도록 함
        }
    }
}
```

구현 설명(C를 이용한 Stop and Wait)

네트워크,물리계층 표현

```
void FromNetworkLayer(packet *buffer){ // 네트워크층에서 데이터링크층으로 패킷을 주기위해 데이터를 생성
    (*buffer).data = i; // 데이터는 1,2,3...n
    i++;
}

void ToPhysicalLayer(frame *s){ // 데이터링크층에서 물리층으로 데이터를 보냄
    s->err = Random(5); // 20%의 확률로 에러가 생성된다. 0일때 에러라고 가정함
    DATA = *s;
}

void FromPhysicalLayer(frame *buffer){ // Receiver의 물리층에서 데이터를 전송받음
    *buffer = DATA;
}

void ToNetworkLayer(packet *buffer){ // Receiver의 데이터링크층에서 네트워크층으로 데이터를 보냄
    if (i == PACKET_NUM){ // 전송받은 i가 프레임의 길이만큼이면 전송 완료이므로 종료
        Finish = 1;
        printf("₩nDISCONNECTED");
    }
}
```

구현 설명(C를 이용한 STOP AND WAIT)

이벤트정의 및 Seq생성함수

```
void WaitSender(event_type * e){ // Sender측의 이벤트 설의
    static int timer = 0; // 타이머 생성

    if (turn == 's'){ // Sender의 차례일때
        timer++; // 타이머를 증가 시킴

        if (timer == 5){ // 타이머가 증가될동안 ack가 오지않는다면 이벤트를 TIMEOUT으로 설정
            *e = TIMEOUT;
            timer = 0;
            return;
        }

        if (DATA.err == 0) // 데이터전송에 에러가 있다면 이벤트를 ERR로 설정
            *e = ERR;
        else{ // 정상적으로 프레임이 도착했다면 ARRIVAL으로 설정
            timer = 0;
            *e = ARRIVAL;
        }
    }
}

void WaitReciever(event_type * e){ // Receiver에서의 이벤트 설정
    if (turn == 'r'){ // Receiver차례일때
        if (DATA.err == 0) // 전송중 에러일때
            *e = ERR; // 이벤트를 ERR로 설정
        else // 아닐경우 정상도착이므로 ARRIVAL로 설정
            *e = ARRIVAL;
    }
}

int Random(int n){
    return rand( ) % n; // 오류를 랜덤하게 생성
}
int MakeSeq(packet *buffer, int *seq){
    *seq = buffer->data % 2; // 보내는 데이터(여기서는 데이터값 ex)1,2,3...n)를 이용해 모듈러 2연산하여 순서번호(seq)를 생성함
    return *seq;
}
```

구현 설명

JAVA를 이용한 소켓통신

Stop and Wait 시뮬레이션

구현설명(JAVA를 이용한 소켓통신 STOP AND WAIT)

Receiver

```
public class Reciever {
    ServerSocket reciever; // 서버를 담당하는 Receiver소켓 생성
    Socket connection = null;
    ObjectOutputStream out; // 객체출력스트림
    ObjectInputStream in; // 객체입력스트림
    String packet, ack, data = ""; //패킷,Ack,데이터를 저장할 변수
    int i = 0, Ack = 0; // Ack 초기화
    Random rand = new Random(); // 랜덤하게 오류를 발생시키기 위해 선언

    public void run() { // 패킷을 받고 Ack를 전송하는 함수 run 함수
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  // 버퍼입력

            System.out.println("RECEIVER");

            reciever = new ServerSocket(1, 10); // 서버 설정

            System.out.println("--------------------");
            System.out.println("Cenecting");

            connection = reciever.accept(); // Sender와 연결


            System.out.println("Cennect!");
            System.out.println("------------------");

            out = new ObjectOutputStream(connection.getOutputStream()); // 객체 출력스트림에 받아온 데이터를 읽음
            out.flush(); // 현재 버퍼에 저장되어 있는 내용을 클라이언트로 전송하고 버퍼를 비움
            in = new ObjectInputStream(connection.getInputStream()); // 객체 입력 스트림에 받은 데이터를 입력
            out.writeObject("Connected    .");

            System.out.println("\nSample > Frame : [Sequence|Data]  Ack : Ack[NUM]\n");
```

구현설명(JAVA를 이용한 소켓통신 STOP AND WAIT)

Receiver

```
do {
    try {
        //전송받은 패킷을 열고 확인하는 부분
        packet = (String) in.readObject(); // 전송받은 패킷을 읽는다
        if (Integer.valueOf(packet.substring(0, 1)) == Ack) { // 현재 Ack설정값과 Sender에서 보낸 순서번호가 같다면
            data += packet.substring(1); // 순서번호와 데이터가 합쳐진 packet에서 데이터만 추출함
            Ack = (Ack == 0) ? 1 : 0; // Ack를 1또는 0으로 설정

            System.out.println("\nReceived Frame : ["  + packet.substring(0, 1)  + "|" +  packet.substring(1, 2) + "]");
        }

        else  // 전과 같은 패킷을 받았다면 중복이므로 버림
            System.out.println("Received Frame : ["  + packet.substring(0, 1)  + "|" +  packet.substring(1, 2) + "]" + " Duplation! Discard this Frame!");


        //Ack를 전송하는 부분
        if (rand.nextInt(10) < 5) { // 20%의 확률로 오류를 발생시킴 (중간에 Ack를 잃어버리는 경우)
            out.writeObject(String.valueOf(Ack));
            System.out.println("Send Ack : Ack[" + Ack + "]");
        }

        else {
            out.writeObject(String.valueOf((Ack + 1) % 2)); // 오류가 아닌 정상적으로 Ack를 전송, +1증하고 모듈러 2연산을 하여 다음에 필요한 패킷의 순서번호를 가지는 Ack를 전송
            System.out.println("Send Ack : Ack[" + Ack + "]");
        }
    } catch (Exception e) {
    }
}while (!packet.equals("end")); // 패킷의 마지막'end'가 나올때까지 반복함

System.out.println("\nReceived Date  : " + data);
System.out.println("Compelete! Connection ended    .\n");
```

구현설명(JAVA를 이용한 소켓통신 STOP AND WAIT)

Sender

```
public class Sender {
    Socket sender; // 클라이언트를 담당하는 Sender소켓 생성
    ObjectOutputStream out; // 객체 출력 스트림
    ObjectInputStream in; // 객체 입력 스트림
    String packet, ack, str, msg;  //패킷,Ack,데이터,msg 를 저장할 변수
    int n, i = 0, seq = 0; // 순서번호 초기화
    Random rand = new Random();

    public void run() {
        try {
            BufferedReader buffer = new BufferedReader(new InputStreamReader(System.in));  // 버퍼입력

            System.out.println("SENDER");

            sender = new Socket("localhost", 1); // 소켓 ip및 포트번호 설정

            System.out.println("--------------------");
            System.out.println("Cenecting");

            out = new ObjectOutputStream(sender.getOutputStream()); // 객체 출력스트림에 받아온 데이터를 읽음
            out.flush();
            in = new ObjectInputStream(sender.getInputStream()); // 객체 입력 스트림에 받은 데이터를 입력
            str = (String) in.readObject();

            System.out.println("connect!");
            System.out.println("--------------------");
            System.out.println("Write Data : ");
            packet = buffer.readLine();  // 키보드로 입력받은 전송할 데이터를 packet에 저장
            n = packet.length();

            System.out.println("\nSample > Frame : [Sequence|Data]  Ack : Ack[NUM]");
```

구현설명(JAVA를 이용한 소켓통신 STOP AND WAIT)

Sender

```
do {
    try {
        if (i < n) {
            msg = String.valueOf(seq); // 전송할 파라미터 순서번호를 뺀 실제 데이터값
            msg = msg.concat(packet.substring( i , i + 1));
        }

        else if (i == n) { // 실제 데이터값 마지막에 끝을 알리는 'end'를 저장
            msg = "end";
            out.writeObject(msg);
            break;
        }


        System.out.println("\nSend Frame : ["  + msg.substring(0, 1)  + "|" +  msg.substring(1, 2) + "]");

        if(rand.nextInt(10) < 5) // 20%의 확률로 오류를 발생시킴 (중간에 Frame를 잃어버리는 경우)
            out.writeObject(msg);

        else{
            System.out.println("Time Out! ReSending! [Frame loss]");
            continue;
        }

        seq = (seq == 0) ? 1 : 0;
        seq = seq % 2; // 순서번호를 모듈러 2연산하여 설정
        out.flush();

        ack = (String) in.readObject(); // Receiver에게 Ack를 받음

        if (ack.equals(String.valueOf(seq))) { // 받은 Ack와 현재 Seq값이 같다면 정상적인 통신
            i++;
            System.out.println("Received Ack[" + ack + "]");
        }
```

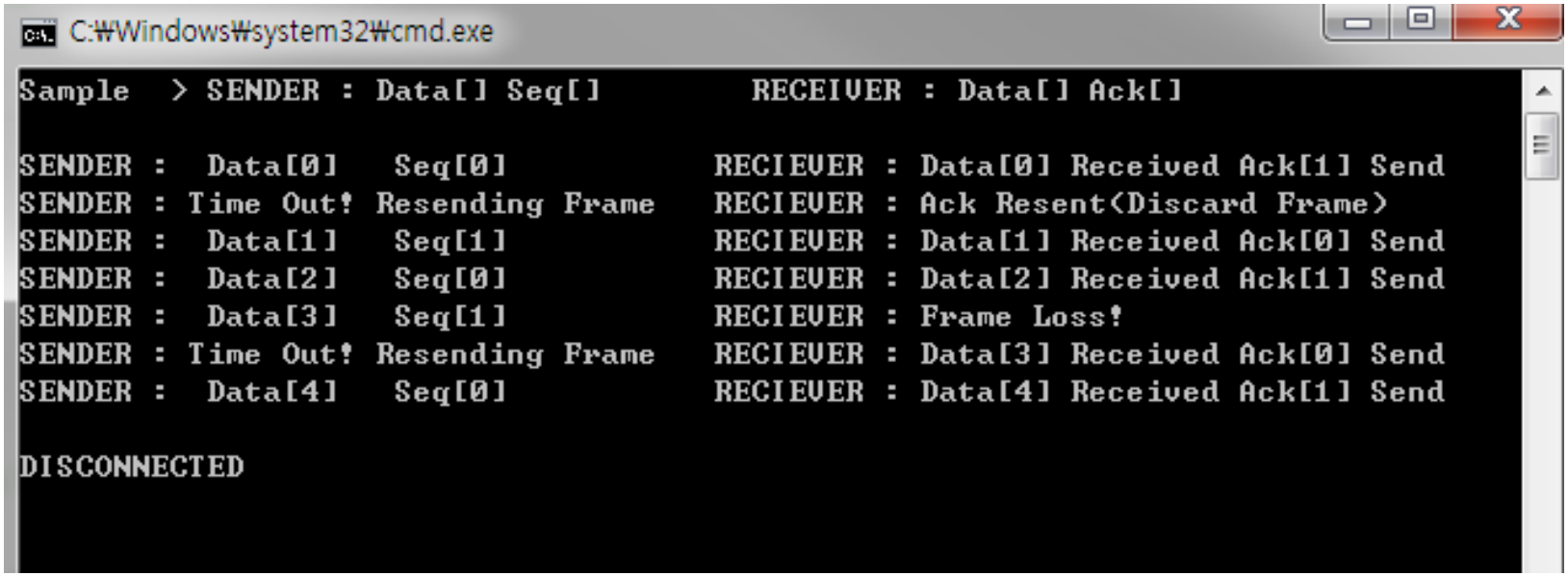
구현설명(JAVA를 이용한 소켓통신 STOP AND WAIT)

Sender

```
                -

                else { // 아니라면 Ack를 잃어버렸다고 가정하여 오류로 봄
                    System.out.println("Time Out! ReSending! [Ack loss]");
                    seq = (seq == 0) ? 1 : 0;
                    seq = seq % 2;
                }
            } catch (Exception e) {
            }
        } while (i < n + 1); // 보낼 데이터의 길이의 +1 만큼 반복한다
        System.out.println("Compelete! Connection ended    .\n");
    } catch (Exception e) {
    } finally {
        try {
            in.close();
            out.close(); // 버퍼 닫기
            sender.close(); // 소켓 종료
        } catch (Exception e) {
        }
    }
}
```

구현결과(C를 이용한 STOP AND WAIT)



구현결과(JAVA를 이용한 소켓통신 STOP AND WAIT)

```
RECEIVER
-------------------
Cenecting
Cennect!
------------------

Sample > Frame : [Sequence|Data]  Ack : Ack[NUM]


Received Frame : [0|2]
Send Ack : Ack[1]
Received Frame : [0|2] Duplation! Discard this Frame!
Send Ack : Ack[1]

Received Frame : [1|0]
Send Ack : Ack[0]

Received Frame : [0|1]
Send Ack : Ack[1]

Received Frame : [1|5]
Send Ack : Ack[0]
Received Frame : [1|5] Duplation! Discard this Frame!
Send Ack : Ack[0]

Received Date  : 2015
Compelete! Connection ended    .
```

```
SENDER
-------------------
Cenecting
connect!
-------------------
Write Data :
2015
Sample > Frame : [Sequence|Data]  Ack : Ack[NUM]

Send Frame : [0|2]
Time Out! ReSending! [Frame loss]

Send Frame : [0|2]
Time Out! ReSending! [Frame loss]

Send Frame : [0|2]
Time Out! ReSending! [Ack loss]

Send Frame : [0|2]
Time Out! ReSending! [Frame loss]

Send Frame : [0|2]
Received Ack[1]

Send Frame : [1|0]
Received Ack[0]

Send Frame : [0|1]
Received Ack[1]

Send Frame : [1|5]
Time Out! ReSending! [Frame loss]

Send Frame : [1|5]
Time Out! ReSending! [Frame loss]

Send Frame : [1|5]
Time Out! ReSending! [Ack loss]

Send Frame : [1|5]
Received Ack[0]
Compelete! Connection ended    .
```

프로젝트 후기

- 시뮬레이션을 통해 Stop and Wait 프로토콜의 동작방식을 더욱 쉽게 이해 할 수 있었음
- 코딩실력의 향상
- 프로토콜의 완벽한 구현이 아니여서 큰 아쉬움