# Assignment #3

양방향 Go-Back-N

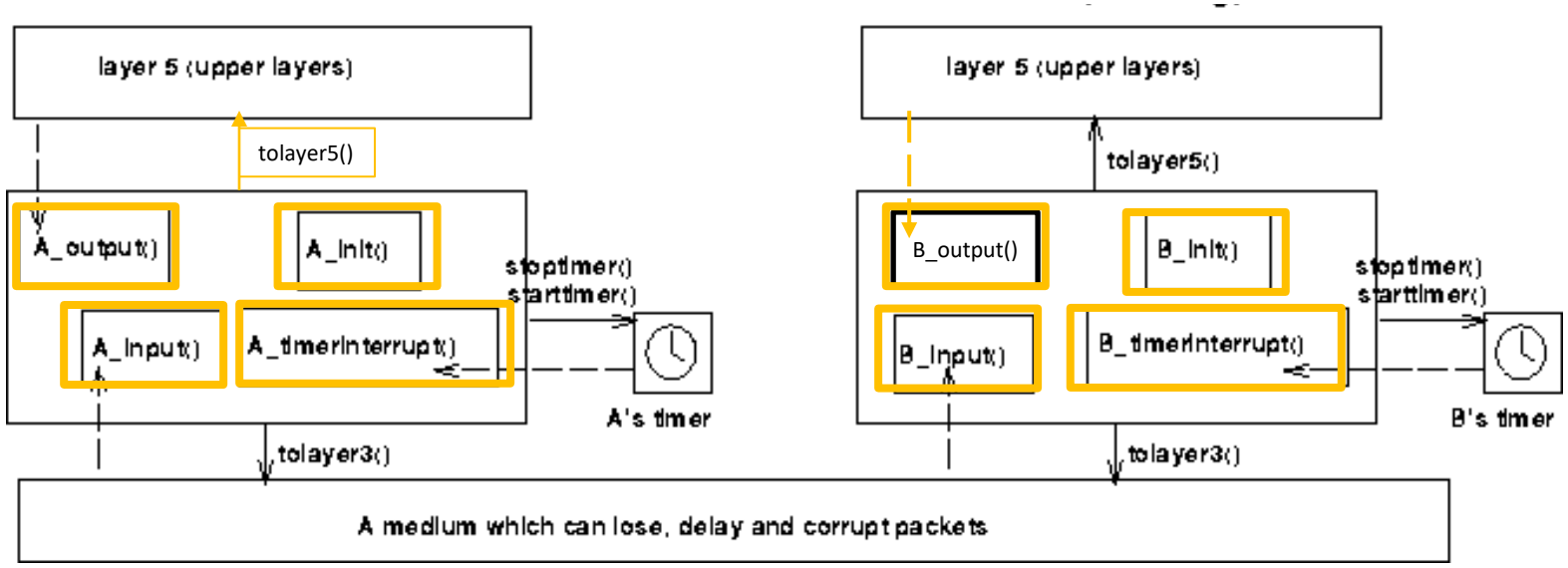
# 목차

- Simulator 구조
- 기본 코드의 Event simulation 동작
- 소프트웨어 인터페이스
- piggyback
- Routine 별 주요 구현내용

# 양방향 Go-Back-N

## 양방향 Go-Back-N 에서 구현할 루틴 목록

| | |
|---|---|
| A_input | B side 에서 전송한 패킷이 A side 에 도착할 때 호출되며, 패킷은 B에서 보낸것. |
| A_output | B side 에 보낼 데이터를 포함하며 A의 upper layer에서 보낼 메시지가 있을 때 호출됨 |
| A_timerinterrupt | 타이머가 만료될 때 호출됨, 패킷 재전송을 제어하기 위해 사용 |
| A_init | A side 루틴이 호출되기전에 한 번 호출됨. 초기화를 위해 사용 |
| B_input | A side 에서 전송한 패킷이 B side에 도착 될 때 호출됨 |
| B_output | A side 에 보낼 데이터를 포함하며 B의 upper layer에서 보낼 메시지가 있을 때 호출됨 |
| B_timerinterrupt | 타이머가 만료될 때 호출됨, 패킷 재전송을 제어하기 위해 사용 |
| B_init | B side 루틴이 호출되기전에 한 번 호출됨. 초기화를 위해 사용 |



# 기본 코드 event simulation

- Main 에서 init() output() input() timerinterrupt() 루틴을 호출

- Init() 루틴
  - 기본 코드에서 호출

- Output() 루틴
  - 기본 코드에서 랜덤하게 message 를 생성하고, 생성 한 message 를 Output 의 인자로 전달
  - 전달 된 message 를 패킷으로 만들고, 만든 패킷을 tolayer3() 루틴을 호출하여 layer3 로 전송

- Input() 루틴
  - 기본 코드에서 layer3 에 전달된 패킷을 input 루틴을 호출하며 input 루틴의 인자로 전달
  - Tolayer5() 루틴을 호출하여, 상위 계층에 data 를 전송

- Timerinterrupt() 루틴
  - 기본 코드에서 호출

# 소프트웨어 인터페이스

- Starttimer(calling_entity, increment)
  - Calling_entity 0 (A-side 타이머 시작용) 혹은 1(B-side 타이머 시작용)
  - Increment 는 float 값이며 timerinterrupt 가 작동하기 전까지 경과할 시간을 나타내는 값
  - A-side 타이머는 A-side 루틴에 의해서만 시작 (또는 중지) 되어야 하며 B-side 의 경우에도 마찬가지

- Stoptimer(calling_entity)
  - Calling_entity 는 0(A-side 타이머 정지) 1(B-side 타이머 정지)

- Tolayer3(calling_entity, packet)
  - Calling_entity 0 (A-side 송신의 경우) ,1(B-side 송신의 경우) 이고, 패킷은 pkt 의 structure
  - 이 루틴을 호출하면 패킷이 다른 엔티티로 향하는 네트워크로 전송

- Tolayer5(calling_entity, message)
  - Calling_entity 0 (A-side to layer 5 delivery) ,1(B-side to layer 5 delivery) 이고, 메시지는 msg 의 structure 타입
  - 이 루틴을 호출하면 message를 상위계층에 전달

# 기본 인터페이스

- Next_seqnum
  - 다음 몇번 패킷을 보낼지 결정 (초기값 :1)
  - Sender에서 보낼 번호가 windowsize 를 넘어가는지 확인하고, 넘어가지 않으면 패킷으로 만들어서 전송
  - 패킷 전송 후 Next_seqnum 1 증가

- Expected_seqnum (초기값 :1)
  - 패킷을 받았을 때, 맞는 번호의 패킷이 왔는지 검증용으로 사용
  - 패킷을 수신하고, corrupt 되지 않은 경우 Expected_seqnum 과 패킷의 seqnum 을 비교하여 순서 확인
  - 순서가 맞지 않은 패킷이 들어올 경우 receiver 에서는 누적ACK 전송
  - Expected_seqnum 은 맞는 번호의 패킷을 corrupt 없이 수신 한 경우 다음 수신될 패킷의 번호 검증을 위해 1 증가

- Base
  - 윈도우의 시작 위치

# ACKstate

- ACK 를 보내야하는 상태인지, ACK 를 보내지 않는 상태인지를 구분하는 값

- ACKstate : 0 or 1
- ACKstate = 0 : ACK 를 보내지 않는 상태
- ACKstate = 1 : ACK 를 보내야 하는 상태
  - 실행 초기 패킷을 받지 않은 상황에 output() 루틴이 호출 될 경우 ACKnum 을 ACK 역할을 하지 않는 999 로 설정하여 패킷에 담아 전송
  - Receiver 가 패킷 수신 후 ACKstate 를 1 로 변경해주어 ACK 를 보낼 상태가 되었음을 의미하도록 사용함
  - Sender 는 패킷을 전송 한 후 ACKstate 를 0 으로 변경해주어 ACK 를 보내지 않는 상태를 의미하도록 사용함
  - Receiver 에서 패킷 수신 후 ACKstate가 1이 된 후 sender 가 패킷을 전송하기 전까지 ACKstate =1

# ACKnum

- 초기 ACKnum = 0
  - ACKnum 은 패킷을 정상적으로 수신한 뒤 Expected_seqnum 으로 갱신
  - 순서가 맞지 않는 패킷이 수신되었을 경우 누적ACK 전송
  - 패킷이 손상되었을 때, 누적ACK 전송
    - ACKnum 의 초기값은 0 , receiver 가 첫번째 패킷을 수신할 때, 첫 패킷이 제대로 수신되지 않은 경우, sender 에 ACKnum이 0 인 ACK 패킷을 전송, 이를 통해 sender 에서 timer 시작.
  - piggyback 방식으로구현된 ACKnum 은 sender 에서 패킷에 넣어 전송

# Piggyback

- 피기백 방식은 데이터를 보낼 때 확인 필드를 함께 보내는 방식
  - 보낼 데이터가 없으면, 일정시간 이후 ACK 패킷 전송
  - 양쪽에서 일정시간마다 Data 를 보내기 때문에 이번 과제에서 이는 고려하지 않음

# Bidirectional GBN

base = 1
my_next_seqnum = 1 //sender 로서의 next_seqnum
Expected_seqnum = 1
ACKstate = 0
my_ACKnum = 0 //0(초기), 999 (Send only data) , receiver 로서의 ACKnum

Data from above :

If ( next_seqnum < base + window size ) {
If (ACKstate = 0) ACKnum = 999
Make_pkt(data, my_next_seqnum, my_ACKnum, checksum16())
Sender window(array)에 저장
tolayer3() 이용 패킷 전송
ACKstate = 0
If ( window내 패킷이 없다면 ) starttimer()
my_next_seqnum ++
}
Else data 전송 거부

Timeout :

현재 base 부터 next sequence number-1 까지 전송
Starttimer()

Packet Received:

ACKstate = 1
If (checksum() != 0) {
next_seqnum, ACKnum 추출
If (Expected_seqnum == next_seqnum) { //receiver 로서 동작
Deliver_data(data)
my_ACKnum = Expected_seqnum
Expected_seqnum++
}
If (ACKnum != 999){ //sender 로서 동작
ACKnum 추출, ACKnum + 1 만큼 base 이동
If (base == my_next_seqnum) stoptimer()
Else starttimer()
}
}

# Init routine

- A_init() & B_init()
- Init() 은 각 루틴이 호출되기 전에 호출하여, 필요한 초기화를 수행하는 루틴

- 주요 구현사항 :
  1. Base, next_seqnum, ACKstate, ACKnum 초기화
  2. 패킷 저장 버퍼

# output routine

- A_output & B_output
- 각각 side 에서 각각의 upper layer 에서 반대 side 에 보낼 메시지가 있을 때 호출이 되는 루틴
- 실제로 layer 5 에서 보낼 메시지가 있는 경우가 아닌 from_layer5 라는 이벤트가 발생 할 경우 호출이 되는 루틴
- 패킷에 정보를 담아 medium 인 layer3 에 tolayer3() 호출하여 전송하는 루틴
- 패킷에는 nextseqnum,acknum,checksum,data 가 담겨져서 medium 으로 전송
- output 루틴은 main에서 호출

- 주요 구현사항 :
  1. 함수의 인자로 어떤것을 전달받는가
  2. 어떤 기능이 필요한가
     1. 패킷 전송할 정보 (data, nex_seqnum, ACKnum, checksum16) 담기
     2. Window size 가 over 될 경우, 패킷을 거부
     3. ACK State 가 0 일때 ACKnum 999 를 넣어서 전송하기
     4. ACKState 가 1 일때 ACKnum 을 넣어서 전송하기
     5. 패킷 전송 후 ACKstate 갱신
     6. 패킷 전송 후 Timer 제어
     7. 패킷 전송 후 Seqnum 갱신

# Input routine

- A_input() & B_input()
- 각각 반대 side 에서 보낸 패킷이 각각 side 에 도착 했을 때 호출이 되는 루틴
- Input은 layer3 에서 패킷을 받을때 호출
- 실제로 패킷을 받는것이 아닌, 코드에서 from_layer3 라는 이벤트가 발생했을 때를 의미
- Tolayer5() 루틴을 호출하여 layer5 로 가는 이벤트를 발생시킴
- Input 루틴은 main에서 호출

- 주요 구현사항 :
  1. _input() 함수의 인자로 무엇을 받는가
  2. 어떤 기능이 필요한가
     1. Corruption 판별
     2. Acknum 갱신
     3. Expect_seqnum 갱신
     4. Base update 를 통한 window 제어 (window 는 GBN 에서 패킷을 받는 창이다 Base 는 윈도우의 시작점)
     5. Timer 제어
        1. 패킷에 ACK number 가 있을 때 stoptimer
     6. 모든 과정 종료 후 ACKstate 갱신

# Timerinterrupt routine

- A_timerinterrupt & B_timerinterrupt
- 타이머가 만료될 때 호출이 되는 루틴으로, 패킷이 일정 시간이 지난 후에 재전송을 위해 호출이 되는 루틴입니다.

- 주요 구현사항 :
  1. Timerinterrupt 가 언제 작동하는가
  2. 어떤 기능이 필요한가
     1. 패킷 전송 후 starttimer (재전송을 하여도, 재전송 실패를 대비)
     2. Base ~ window size 재전송

# Checksum

다음시간에 설명

## 감사합니다.

- 조교 이메일 [redacted]