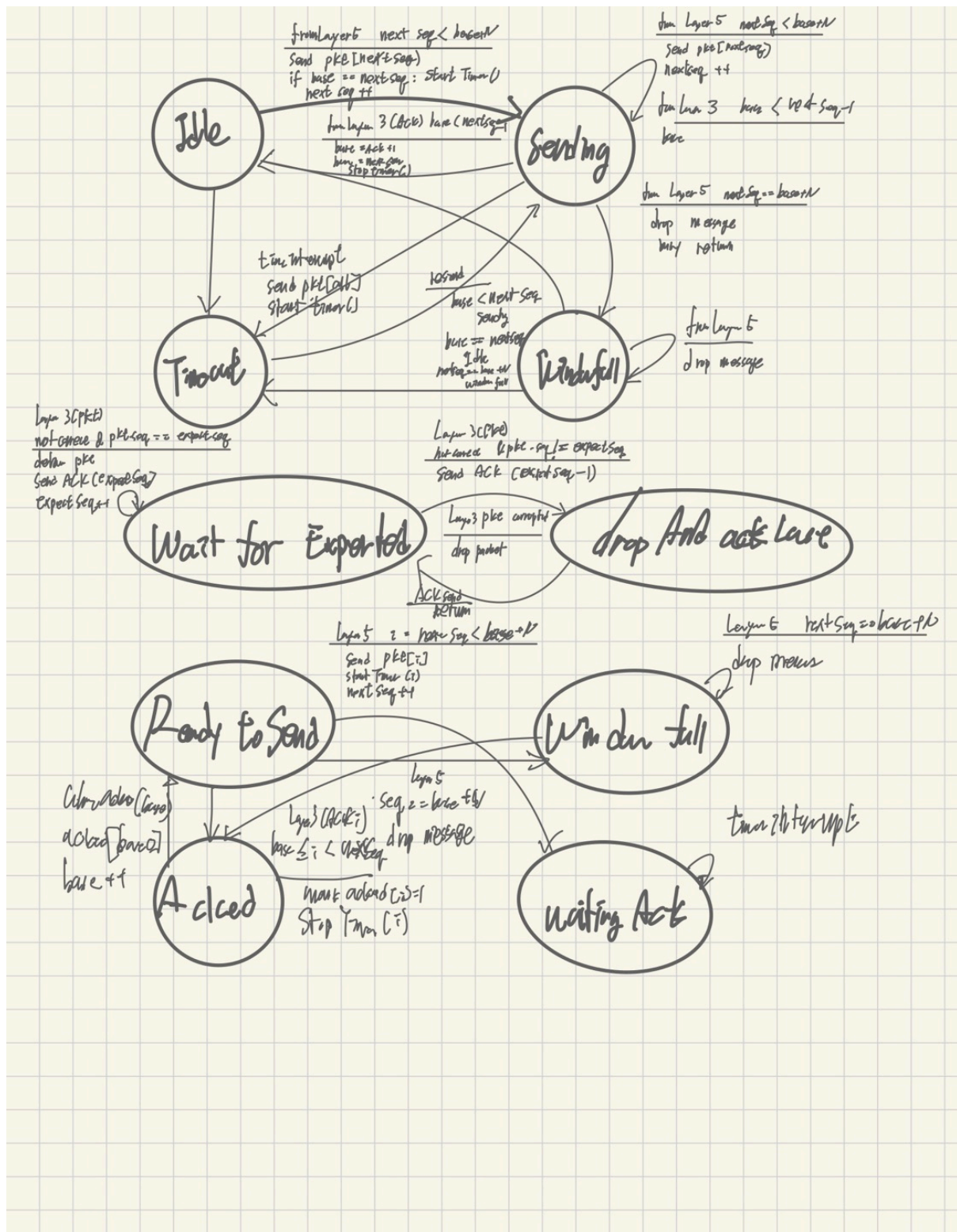
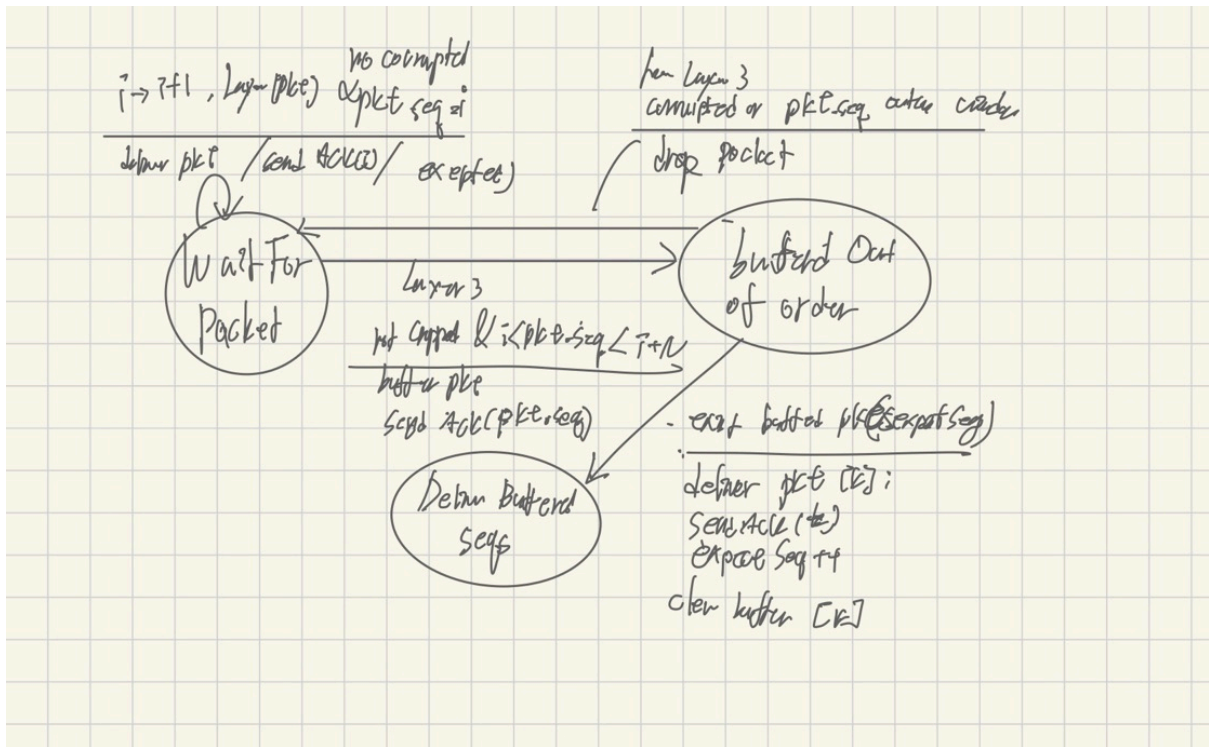


컴퓨터 네트워크 Assignment

prog_2021202003_강준우

FSM 설계도





구현 코드의 구조 및 설계 의도

1. 코드 구조 및 설계 의도 (main 함수 이전 부분)

Selective-Repeat 송신 측

A_init()

송신 측의 상태를 초기 세팅하는 역할을 합니다. 송신 윈도우의 시작 지점을 나타내는 `SR_A_base`와, 다음으로 보낼 패킷의 시퀀스 번호인 `SR_A_nextseqnum`을 둘 다 0으로 설정하여 첫 전송을 준비합니다. 또한 수신 측에서 처음으로 기대하는 패킷 번호인 `SR_A_expectedseqnum`도 0으로 초기화합니다. 이후, ACK 수신 여부를 추적하는 `SR_A_acked` 배열과 데이터 수신 여부를 나타내는 `SR_A_received` 배열의 모든 항목을 0으로 설정해, 상태 정보가 모두 초기화된 깨끗한 시작점을 만듭니다.

A_output(struct msg message)

상위 계층에서 메시지를 받으면, 먼저 현재 보낼 수 있는 상황인지 판단합니다. 이를 위해 다음에 보낼 순번인 `SR_A_nextseqnum`이 송신 윈도우의 끝 범위인 `SR_A_base + N`보다 작은지를 검사합니다. 전송이 가능한 경우, 해당 시퀀스 번호를 포함한 패킷을 새로 만들어

메시지 내용을 복사하고, 체크섬을 계산한 뒤 송신 버퍼에 저장합니다. 이후 이 패킷을 네트워크 계층에 전달하며, 그 시퀀스 번호에 해당하는 전용 타이머도 동시에 작동시킵니다. 마지막으로 **SR_A_nextseqnum** 값을 증가시켜 다음 패킷 전송을 준비합니다.

A_input(struct pkt packet)

네트워크 계층에서 패킷이 도착하면 먼저 체크섬을 확인하여 데이터가 손상되지 않았는지 검사합니다. 문제가 없는 **ACK** 패킷이라면, **ACK**에 담긴 번호가 현재 송신 윈도우 범위에 포함되는지 확인합니다. 만약 포함된다면 해당 시퀀스에 해당하는 항목을 **SR_A_acked** 배열에서 수신 완료로 표시하고, 그 번호에 설정된 타이머를 멈춥니다. 그 후에는 **SR_A_base**부터 시작해 연속적으로 **ACK**된 구간만큼 송신 윈도우의 시작 위치를 앞으로 이동시켜 윈도우를 슬라이딩합니다.

만약 도착한 패킷이 데이터 패킷이라면, 그 시퀀스 번호가 현재 기대하는 번호인 **SR_A_expectedseqnum**부터 그에 윈도우 크기를 더한 값보다 작은 범위에 있는지를 확인합니다. 조건을 만족하면 해당 위치의 버퍼에 패킷을 저장하고, **SR_A_received** 상태를 갱신합니다. 이후 즉시 **ACK**를 만들어 상대방에게 회신하고, 연속적으로 순서에 맞게 도착한 데이터가 있다면 **tolayer5**를 호출해 상위 계층에 데이터를 전달합니다. 이때마다 기대하는 시퀀스 번호도 하나씩 증가시킵니다.

반면 이미 수신했거나, 순서상 아직 수신할 차례가 아닌 패킷이 들어오면, 마지막에 보냈던 **ACK** 번호를 사용해 다시 **ACK** 패킷을 보내 중복 응답을 처리합니다.

A_timerinterrupt(int seqnum)

특정 시퀀스 번호에 대해 설정된 개별 타이머가 만료되면, 먼저 해당 번호에 해당하는 패킷을 송신 버퍼에서 찾아 꺼냅니다. 이후 이 패킷을 다시 네트워크 계층으로 전송하여 재전송을 수행합니다. 전송이 끝나면 같은 시퀀스 번호를 기준으로 타이머를 다시 작동시켜, 아직 **ACK**를 받지 못한 경우 주기적으로 재전송이 이뤄지도록 합니다.

Selective-Repeat 수신 측

B_init()

수신 측이 초기화될 때, 윈도우 관련 변수인 **SR_B_base**, **SR_B_nextseqnum**, 그리고 **SR_B_expectedseqnum**을 모두 0으로 설정하여 송수신 상태를 처음 위치로 맞춥니다. 또한 **ACK** 수신 여부를 추적하는 **SR_B_acked** 배열과 데이터 수신 여부를 나타내는 **SR_B_received** 배열의 모든 요소를 **false**로 초기화하여 깨끗한 상태의 버퍼와 윈도우를 구성합니다.

B_output(struct msg message)

수신 측이 독립적으로 메시지를 전송해야 하는 경우, 이 함수는 송신 측의 **A_output**과 동일한 방식으로 작동합니다. 먼저 현재 시퀀스 번호가 윈도우 크기 제한 내에 있는지를 확인한 뒤, 새로운 패킷을 생성하고 이를 네트워크 계층으로 전송합니다. 전송 직후에는 해당 시퀀스 번호에 대한 개별 타이머를 시작하여 이후 재전송을 위한 준비를 합니다.

B_input(struct pkt packet)

도착한 패킷이 수신되면 먼저 체크섬을 통해 손상 여부를 확인합니다. 손상되지 않았고 **ACK** 패킷인 경우, **ACK** 번호가 현재 송신 윈도우 범위 안에 있으면 해당 번호에 해당하는 항목을 **SR_B_acked** 배열에서 **true**로 표시하고, 연속적으로 **ACK**된 만큼 윈도우의 시작 위치를 **SR_B_base**를 기준으로 앞으로 이동시켜 줍니다. 데이터 패킷의 경우, 시퀀스 번호가 기대하는 범위 내에 있으면 이를 수신 버퍼에 저장하고, 수신 여부 배열을 갱신한 후 즉시 **ACK** 패킷을 생성하여 응답으로 보냅니다. 이어서 버퍼에 연속된 순서대로 도착한 데이터가 있다면 상위 계층으로 하나씩 순서에 맞게 전달하면서 기대 시퀀스 번호를 계속 증가시킵니다. 만약 이미 수신했거나 순서가 맞지 않는 패킷이 도착하면, 중복으로 간주하고 마지막으로 보낸 **ACK** 번호를 다시 전송합니다.

B_timerinterrupt(int seqnum)

특정 시퀀스 번호에 대한 타이머가 만료되면, 해당 번호의 패킷을 송신 버퍼에서 꺼내 다시 전송합니다. 이때 신뢰성을 유지하기 위해 재전송 직후에는 동일한 시퀀스 번호를 기준으로 타이머를 다시 시작하여 **ACK**이 도착할 때까지 주기적으로 재전송이 이뤄지도록 합니다.

Go-Back-N 송신 측

A_init()

전송을 위한 윈도우의 시작 시퀀스 번호인 **A_base**, 다음에 전송할 시퀀스 번호인 **A_nextseqnum**, 그리고 수신 시 기대하는 시퀀스 번호인 **A_expectedseqnum**을 모두 0으로 초기화하여 송수신의 시작 상태를 설정합니다. 이후 피기백 방식으로 **ACK**을 함께 보낼 수 있도록 하는 변수 **A_lastACKToSend**를 -1로 설정해, 아직 보낼 **ACK**이 없음을 명시합니다

A_output(struct msg message)

상위 계층에서 메시지를 전달받으면, 현재 시퀀스 번호인 **A_nextseqnum**이 윈도우 내에 있는지를 확인합니다. 이 값이 **A_base + N**보다 작으면 전송 가능한 상태로 판단하여, 새로운 패킷을 생성합니다. 이때 아직 보내지 못한 **ACK**이 있다면 **acknum** 필드에 그 값을 포함시켜 피기백합니다. 그렇지 않으면 **acknum**을 999 같은 무의미한 값으로 채워 넣습니다. 생성한 패킷은 송신 윈도우 버퍼에 저장되고, 네트워크 계층으로 전송됩니다. 만약 이번 전송이 윈도우에서 가장 앞선 패킷이라면 타이머를 시작해 재전송 대비를 합니다. 이후 다음 시퀀스를 위해 **A_nextseqnum**을 증가시키고, 피기백용 **ACK** 변수는 초기화합니다.

A_input(struct pkt packet)

수신한 패킷의 체크섬이 유효하면, 먼저 이 패킷이 ACK 패킷인지 확인합니다. ACK 번호가 A_base와 A_nextseqnum 사이에 있다면 해당 ACK을 수신한 것으로 간주하고 A_base를 ACK 번호 다음으로 이동시켜 윈도우를 슬라이딩합니다. 이후 남은 패킷이 있다면 타이머를 재시작하고, 없다면 타이머를 종료합니다.

만약 데이터 패킷이라면, 이 패킷의 시퀀스 번호가 A_expectedseqnum과 정확히 일치하는 경우에만 상위 계층으로 데이터를 전달하고, 기대 시퀀스 번호를 증가시킵니다. 또한 다음 전송 시 피기백할 수 있도록 A_lastACKToSend에 이 값을 저장합니다. 반대로 순서가 맞지 않거나 중복된 패킷이 도착하면, 가장 최근에 받은 데이터에 대응하는 ACK 번호를 기반으로 즉시 중복 ACK을 전송합니다.

A_timerinterrupt()

설정된 타이머가 만료되면 현재 윈도우 안에 있는 모든 패킷을 재전송합니다. 이를 위해 A_base부터 A_nextseqnum 미만까지 반복하며 각각을 네트워크 계층으로 다시 보낸 후, 타이머를 재시작하여 다음 재전송을 준비합니다.

Go-Back-N 수신 측

B_init()

전송 측과 동일하게 윈도우 시작 시퀀스 번호인 B_base, 다음 전송 번호인 B_nextseqnum, 그리고 수신 시 기대하는 시퀀스 번호인 B_expectedseqnum을 모두 0으로 설정합니다. 이후 피기백 ACK 번호를 저장할 B_lastACKToSend도 -1로 초기화하여, 현재 보낼 ACK이 없음을 명확히 표시합니다.

B_output(struct msg message)

수신 측이 직접 전송해야 할 메시지가 있을 경우, 전송 측과 거의 동일한 방식으로 동작합니다. 현재 B_nextseqnum이 윈도우 범위 내에 있으면 새 패킷을 생성하고, 이때 ACK 번호는 피기백하지 않고 0으로 고정합니다. 패킷을 전송한 뒤, 해당 번호가 윈도우 맨 앞이라면 타이머를 시작합니다. 전송이 끝난 후에는 시퀀스 번호를 증가시켜 다음 전송에 대비합니다.

B_input(struct pkt packet)

패킷이 도착하면 먼저 체크섬을 검증하여 손상 여부를 판단합니다. 유효한 패킷이라면 두 가지 경우로 나뉩니다.

ACK 패킷인 경우:

ACK 번호가 현재 윈도우(**B_base**와 **B_nextseqnum**) 범위에 있다면 그만큼 송신이 성공한 것으로 간주하고 **B_base**를 업데이트합니다. 만약 윈도우가 완전히 비었다면 타이머를 멈추고, 일부가 남아 있다면 타이머를 다시 설정하여 남은 전송을 감시합니다.

데이터 패킷인 경우:

수신된 데이터의 시퀀스 번호가 **B_expectedseqnum**과 정확히 일치할 경우, 정상적인 순서로 도착한 데이터이므로 상위 계층으로 전달합니다. 그리고 **B_expectedseqnum**을 증가시키고, 직전 시퀀스를 **ACK** 번호로 포함시켜 새로운 **ACK** 패킷을 만들어 즉시 전송합니다.

반면 시퀀스가 맞지 않거나 이미 처리한 데이터가 다시 도착하면, 중복된 것으로 판단하고 마지막으로 보낸 **ACK** 번호를 재전송하여 송신 측에 올바른 기대 시퀀스를 알립니다. **Go-Back-N**은 순서가 맞지 않으면 버퍼링 없이 무조건 폐기하기 때문에, 수신 측은 항상 **B_expectedseqnum**에 정확히 일치하는 패킷만 처리합니다.

B_timerinterrupt()

타이머가 만료되면, **B_base**부터 **B_nextseqnum**까지 현재 윈도우에 있는 모든 패킷을 순서대로 재전송합니다. 이와 함께 타이머를 다시 설정하여 향후 재전송을 감시합니다. **Go-Back-N**에서는 한 개의 타이머만 사용하고, 재전송 시에는 윈도우 전체를 다시 보내는 방식으로 신뢰성을 보장합니다.

시뮬레이션 결과 분석

Selective-Repeat 시뮬레이션에서는 **layer5**에서 50개의 메시지를 전송하였으며 시뮬레이션은 249.08 단위 시간에 종료되었습니다. 애플리케이션 계층에는 총 420바이트가 전달되어 21개의 메시지가 성공적으로 수신되었으며 성공률은 약 42%, 처리량은 1.69바이트/초를 기록하였습니다. 이 과정에서 약 30회 이상의 타이머 인터럽트가 발생하였는데, 타이머 간격이 **RTT**(약 7단위)의 네 배로 설정되어 손실 시 재전송까지 대기 시간이 길어졌습니다. **ACK** 전용 패킷(**seqnum = -1**)이 자주 손실·손상되어 개별 패킷 재전송을 유발하면서 전체 오버헤드가 증가하였습니다. **out-of-order**로 도착한 패킷들은 즉시 **ACK**를 전송한 뒤 내부 버퍼에 저장되었고, 순차적으로 도착 가능한 패킷이 모일 때 한꺼번에 애플리케이션 계층으로 전달되어 이벤트 시각 49.36에서 “bbbb.../dddd.../eeee...” 세 개의 메시지가 한 번에 수신되는 현상이 관찰되었습니다.

Go-Back-N 시뮬레이션에서는 **layer5**에서 50개의 메시지를 전송하였으며 시뮬레이션은 261.16 단위 시간에 종료되었습니다. 애플리케이션 계층에는 총 320바이트가 전달되어 16개의 메시지가 성공적으로 수신되었으며 성공률은 약 32%, 처리량은 1.23바이트/초를 보여줍니다. **Go-Back-N** 방식에서는 윈도우 앞부분 패킷 하나가 손실되면 이후의 모든 미확인 패킷을 재전송해야 하므로 오버헤드가 크게 증가하였습니다. **tolayer3** 단계에서

초기 **seqnum03** RTT만큼 대기해야 했고 이 구간 동안 **tolayer5** 호출이 거의 중단되는 지연 현상이 나타났습니다.

결과 시각화

시뮬레이션 파라미터

프로토콜	drop rate	currrupt rate	송신 메시지 수
SR	0.20	0.10	50
GBN	0.20	0.10	50

시뮬레이션 결과

프로토콜	종료 시간	총 수신 바이트	처리량
SR	249.082550	420	1.69
GBN	261.163849	320	1.23

Selective Repeat(SR) 프로토콜은 손실되거나 손상된 패킷만 골라서 재전송하는 방식이기 때문에, 전체적인 전송 효율이 높습니다. 수신 측에서는 순서가 맞지 않는 패킷이라도 일단 받아서 버퍼에 저장해두었다가, 빠진 패킷이 나중에 도착하면 그때 순서대로 넘겨줄 수 있습니다. 이런 구조 덕분에 한두 개의 패킷이 손실되더라도 나머지는 문제없이 처리되며, 불필요한 재전송이 줄어듭니다. 이번 시뮬레이션에서도 **SR**은 **50**개의 메시지 중 **420**바이트를 수신했고, 처리량은 초당 **1.69**바이트로 비교적 높은 성능을 보여주었습니다.

반면 **Go-Back-N(GBN)**은 패킷이 하나라도 손실되면 그 이후에 전송된 것들을 전부 다시 보내야 해서, 오버헤드가 크게 발생합니다. 수신자는 순서가 맞지 않으면 일단 모두 버리기 때문에, 예를 들어 **packet 3**이 손실되면 그 뒤의 패킷들이 멀쩡히 도착하더라도 다시 받아야 하는 상황이 생깁니다. 그 결과 재전송이 누적되면서 전체 처리 속도가 느려지고, 수신된 데이터 양도 줄어들게 됩니다. 실제로 **GBN**은 **320**바이트만 수신되었고, 처리량도 초당 **1.23**바이트로 **SR**보다 낮았습니다. 이런 차이는 두 프로토콜이 어떻게 손실을 처리하느냐에서 비롯된다고 볼 수 있습니다.

```
COMNE [SSH: vm-ubuntu]
PROBLEMS OUTPUT TERMINAL ... bash - COMNE

EVENT time: 14.086071, type: 1, fromlayer5 entity: 0
  GENERATE NEXT ARRIVAL: creating new arrival
  MAINLOOP: data given to student: bbbbbbbbbbbbbbbbbbb
  TOLAYER3: packet 0 being lost
  START TIMER: starting timer at 14.086071 for sequence number 0

EVENT time: 15.398986, type: 1, fromlayer5 entity: 1
  GENERATE NEXT ARRIVAL: creating new arrival
  MAINLOOP: data given to student: ccccccccccccccccccc
  TOLAYER3: check: 1981 ccccccccccccccccccc
  TOLAYER3: packet 1 being corrupted
  TOLAYER3: scheduling arrival on other side at time 21.185289
  START TIMER: starting timer at 15.398986 for sequence number 1

EVENT time: 16.532869, type: 1, fromlayer5 entity: 0
  GENERATE NEXT ARRIVAL: creating new arrival
  MAINLOOP: data given to student: dddddddddddddddddd
  TOLAYER3: check: 2001 dddddddddddddddddd
  TOLAYER3: scheduling arrival on other side at time 21.805099
  START TIMER: starting timer at 16.532869 for sequence number 1

EVENT time: 16.718023, type: 2, fromlayer3 entity: 0
  TOLAYER3: check: -1
  TOLAYER3: scheduling arrival on other side at time 26.007740
  TOLAYER5: data received: aaaaaaaaaaaaaaaaaaa

EVENT time: 17.064447, type: 1, fromlayer5 entity: 0
  GENERATE NEXT ARRIVAL: creating new arrival
  MAINLOOP: data given to student: eeeeeeeeeeeeeeeeeee
  TOLAYER3: check: 2022 eeeeeeeeeeeeeeeeeee
  TOLAYER3: scheduling arrival on other side at time 30.071342
  START TIMER: starting timer at 17.064447 for sequence number 2

EVENT time: 21.185289, type: 2, fromlayer3 entity: 0

EVENT time: 21.805099, type: 2, fromlayer3 entity: 1
  TOLAYER3: check: 0
  TOLAYER3: packet -1 being corrupted
  TOLAYER3: scheduling arrival on other side at time 27.381901

EVENT time: 25.974073, type: 1, fromlayer5 entity: 0
  GENERATE NEXT ARRIVAL: creating new arrival
  MAINLOOP: data given to student: ffffffffffffffffffff
  TOLAYER3: check: 2043 ffffffffffffffffffff
  TOLAYER3: packet 3 being corrupted
  TOLAYER3: scheduling arrival on other side at time 37.273178
  START TIMER: starting timer at 25.974073 for sequence number 3

EVENT time: 26.007740, type: 2, fromlayer3 entity: 1
  STOP TIMER: stopping timer at 26.007740 for sequence number 0

EVENT time: 27.381901, type: 2, fromlayer3 entity: 0
```

Selective Repeat에서 손실 발생 후 재전송까지의 흐름, timerinterrupt에 의한 재전송 부분

```
118.0950 Aa .ab .* 2 of 2 ↑ ↓ ×
EVENT time: 118.095078, type: 1, fromlayer3 entity: 1
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: tttttttttttttttttttt
    TOLAYER3: check: 2327 tttttttttttttttttttt
    TOLAYER3: packet 7 being corrupted
    TOLAYER3: scheduling arrival on other side at time 124.863533
    START TIMER: starting timer at 118.095078 for sequence number 7

EVENT time: 118.811867, type: 2, fromlayer3 entity: 1
    STOP TIMER: stopping timer at 118.811867 for sequence number 3

EVENT time: 120.674812, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: uuuuuuuuuuuuuuuuuuuuu

EVENT time: 122.718498, type: 0, timerinterrupt entity: 1
    START TIMER: starting timer at 122.718498 for sequence number 5
    TOLAYER3: check: 2245 ppppppppppppppppppppp
    TOLAYER3: scheduling arrival on other side at time 134.063766

EVENT time: 123.712105, type: 2, fromlayer3 entity: 1
    TOLAYER3: check: 2
    TOLAYER3: scheduling arrival on other side at time 143.413040
    TOLAYER5: data received: ffffffffffffffffffffffff
    TOLAYER5: data received: hhhhhhhhhhhhhhhhhhhhhh
    TOLAYER5: data received: jjjjjjjjjjjjjjjjjjjjjj
    TOLAYER5: data received: kkkkkkkkkkkkkkkkkkkkkk
    TOLAYER5: data received: llllllllllllllllllllll
```

손상된 패킷 다시 전송되고 복구되는 시점


```
EVENT time: 14.086071, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: bbbbbbbbbbbbbbbbbbbb
    TOLAYER3: packet 0 being lost
    START TIMER: starting timer at 14.086071

EVENT time: 15.398986, type: 1, fromlayer5 entity: 1
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: cccccccccccccccccccc
    TOLAYER3: check: 1981 cccccccccccccccccccc
    TOLAYER3: packet 1 being corrupted
    TOLAYER3: scheduling arrival on other side at time 21.185289

EVENT time: 16.532869, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: dddddddddddddddddddd
    TOLAYER3: check: 3000 dddddddddddddddddddd
    TOLAYER3: scheduling arrival on other side at time 21.805099

EVENT time: 16.718023, type: 2, fromlayer3 entity: 0
    STOP TIMER: stopping timer at 16.718023
    START TIMER: starting timer at 16.718023

EVENT time: 17.064447, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: eeeeeeeeeeeeeeeeeeee
    TOLAYER3: check: 3021 eeeeeeeeeeeeeeeeeeee
    TOLAYER3: scheduling arrival on other side at time 30.823763

EVENT time: 21.185289, type: 2, fromlayer3 entity: 0

EVENT time: 21.805099, type: 2, fromlayer3 entity: 1

EVENT time: 24.245464, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: ffffffffffffffffffff
    TOLAYER3: packet 3 being lost

EVENT time: 30.823763, type: 2, fromlayer3 entity: 1

EVENT time: 31.639402, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: gggggggggggggggggggg
    TOLAYER3: packet 4 being lost

EVENT time: 34.320404, type: 1, fromlayer5 entity: 1
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: hhhhhhhhhhhhhhhhhhhh
    TOLAYER3: check: 2082 hhhhhhhhhhhhhhhhhhhh
    TOLAYER3: scheduling arrival on other side at time 38.478989

EVENT time: 35.210957, type: 1, fromlayer5 entity: 1
```

패킷 손실과 재전송이 일어나면서 누적 재전송을 보여줍니다.

```

EVENT time: 88.949722, type: 1, fromlayer5 entity: 1
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: tttttttttttttttttttt
    TOLAYER3: check: 2327 tttttttttttttttttttt
    TOLAYER3: packet 7 being corrupted
    TOLAYER3: scheduling arrival on other side at time 117.592262

EVENT time: 89.640083, type: 2, fromlayer3 entity: 0
    TOLAYER3: check: 0
    TOLAYER3: scheduling arrival on other side at time 107.350479

EVENT time: 90.064148, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: uuuuuuuuuuuuuuuuuuuu

EVENT time: 92.388580, type: 2, fromlayer3 entity: 1

EVENT time: 93.902679, type: 2, fromlayer3 entity: 1

EVENT time: 95.000702, type: 2, fromlayer3 entity: 1

EVENT time: 97.150085, type: 2, fromlayer3 entity: 1

EVENT time: 98.125992, type: 1, fromlayer5 entity: 0
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: vvvvvvvvvvvvvvvvvvvv

EVENT time: 98.465988, type: 2, fromlayer3 entity: 0
    TOLAYER3: check: 0
    TOLAYER3: scheduling arrival on other side at time 114.918991

EVENT time: 98.733955, type: 2, fromlayer3 entity: 1

EVENT time: 99.352875, type: 0, timerinterrupt entity: 1
    START TIMER: starting timer at 99.352875
    TOLAYER3: check: 1940 aaaaaaaaaaaaaaaaaaaaaa
    TOLAYER3: scheduling arrival on other side at time 121.765968
    TOLAYER3: check: 1981 cccccccccccccccccccccc
    TOLAYER3: scheduling arrival on other side at time 129.367981
    TOLAYER3: check: 2082 hhhhhhhhhhhhhhhhhhhhhh
    TOLAYER3: scheduling arrival on other side at time 138.498871
    TOLAYER3: check: 2103 iiiiiiiiiiiiiiiiiiiiii
    TOLAYER3: scheduling arrival on other side at time 142.876892
    TOLAYER3: check: 2164 llllllllllllllllllllll
    TOLAYER3: scheduling arrival on other side at time 146.198654
    TOLAYER3: check: 2225 oooooooooooooooooooooo
    TOLAYER3: packet 5 being corrupted
    TOLAYER3: scheduling arrival on other side at time 152.967102
    TOLAYER3: check: 2286 rrrrrrrrrrrrrrrrrrrrrr
    TOLAYER3: scheduling arrival on other side at time 156.685318
    TOLAYER3: check: 2327 tttttttttttttttttttt
    TOLAYER3: scheduling arrival on other side at time 161.556992

```


패킷 손상이후 재전송 재수신 모습을 담았습니다.

```
EVENT time: 198.781219, type: 2, fromlayer3 entity: 1
    TOLAYER5: data received: ffffffffffffffffffff

EVENT time: 199.828079, type: 2, fromlayer3 entity: 1
    TOLAYER5: data received: gggggggggggggggggggg

EVENT time: 199.951508, type: 1, fromlayer5 entity: 1
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: ooooooooooooooooooooo
    TOLAYER3: packet 11 being lost

EVENT time: 200.792282, type: 2, fromlayer3 entity: 0
    TOLAYER5: data received: rrrrrrrrrrrrrrrrrrrr

EVENT time: 205.149841, type: 0, timerinterrupt entity: 1
    START TIMER: starting timer at 205.149841
    TOLAYER3: check: 2225 ooooooooooooooooooooo
    TOLAYER3: packet 5 being corrupted
    TOLAYER3: scheduling arrival on other side at time 275.678802
    TOLAYER3: check: 2286 rrrrrrrrrrrrrrrrrrrr
    TOLAYER3: scheduling arrival on other side at time 282.492920
    TOLAYER3: packet 7 being lost
    TOLAYER3: check: 2408 xxxxxxxxxxxxxxxxxxxxxxxx
    TOLAYER3: scheduling arrival on other side at time 291.956238
    TOLAYER3: check: 2169 llllllllllllllllllll
    TOLAYER3: scheduling arrival on other side at time 296.378357
    TOLAYER3: check: 2190 mmmmmmmmmmmmmmmmmmm
    TOLAYER3: scheduling arrival on other side at time 303.103394
    TOLAYER3: check: 2231 ooooooooooooooooooooo
    TOLAYER3: scheduling arrival on other side at time 311.348877

EVENT time: 207.169449, type: 1, fromlayer5 entity: 1
    GENERATE NEXT ARRIVAL: creating new arrival
    MAINLOOP: data given to student: pppppppppppppppppppp
    TOLAYER3: check: 2252 pppppppppppppppppppp
    TOLAYER3: scheduling arrival on other side at time 319.521576

EVENT time: 207.800369, type: 2, fromlayer3 entity: 1
    TOLAYER5: data received: jjjjjjjjjjjjjjjjjjj

EVENT time: 209.144577, type: 2, fromlayer3 entity: 0
    TOLAYER5: data received: ttttttttttttttttttt

EVENT time: 210.260391, type: 2, fromlayer3 entity: 1
    TOLAYER3: check: 5
    TOLAYER3: scheduling arrival on other side at time 324.009766

EVENT time: 214.757614, type: 2, fromlayer3 entity: 0
    TOLAYER5: data received: xxxxxxxxxxxxxxxxxxxxxx
```

종료 없이 정상적으로 전송 수신되는 흐름을 담았습니다.