2024/09/26, 2024 네트워크 세미나

TCP/IP 프로토콜

-13장 전송 계층, 14장 UDP-

김 혜 정(hyejeong@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

- 전송 계층
- UDP(User Datagram Protocol)

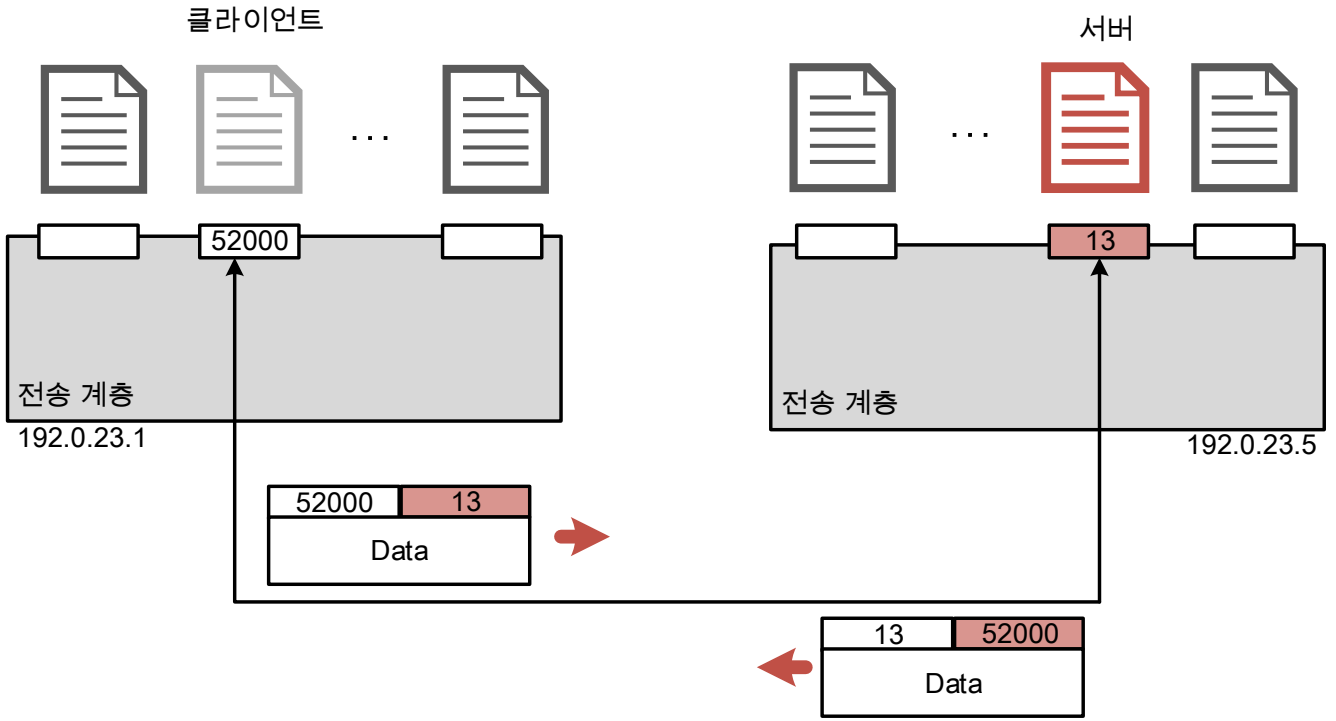
목 차

전송 계층

- 정의
  - OSI 모델의 네 번째 계층으로, 종단 간 데이터 전송을 관리하는 계층

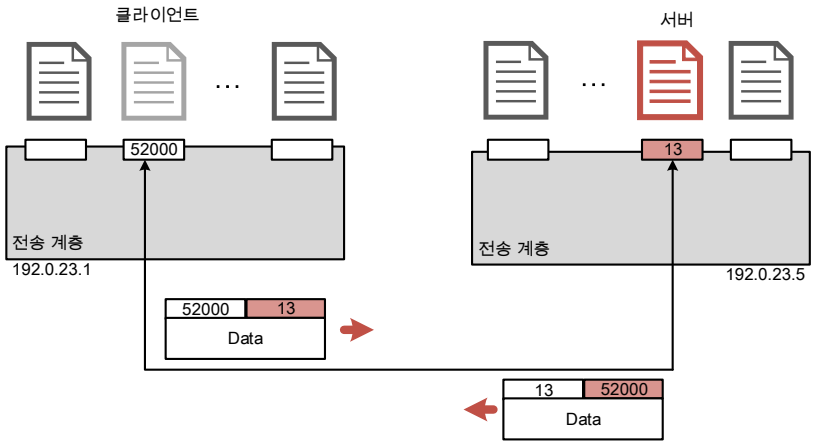
전송 계층

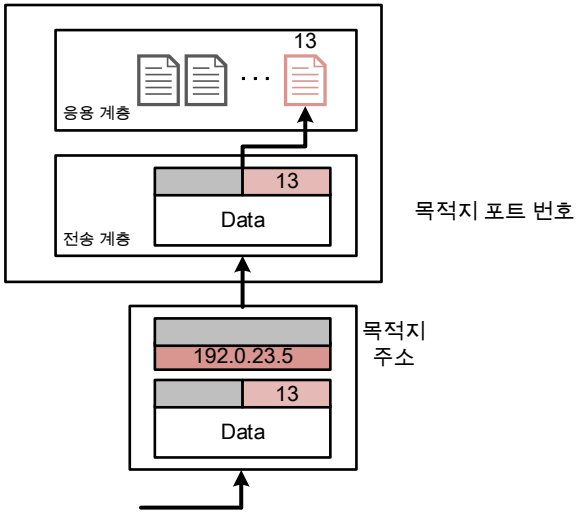
- 프로세스 간 통신(1/3)
  - 정의
    - 데이터 전송에서 그치지 않고 데이터를 포트 번호에 대응하는 프로세스에 전달하고 처리하는 과정
      - 즉, 호스트 대 호스트 통신이 아닌 프로세스 대 프로세스 통신



전송 계층

- 프로세스 간 통신(2/3)
  - 포트 번호(Port Number)
    - 정의
      - 컴퓨터 네트워크에서 특정 프로세스나 서비스와 통신하기 위해 사용되는 숫자
        - e.g., 192.0.23.5:13
    - 역할
      - 특정 호스트 내에 있는 여러 프로세스 중에서 하나의 프로세스에 접근하기 위함





전송 계층

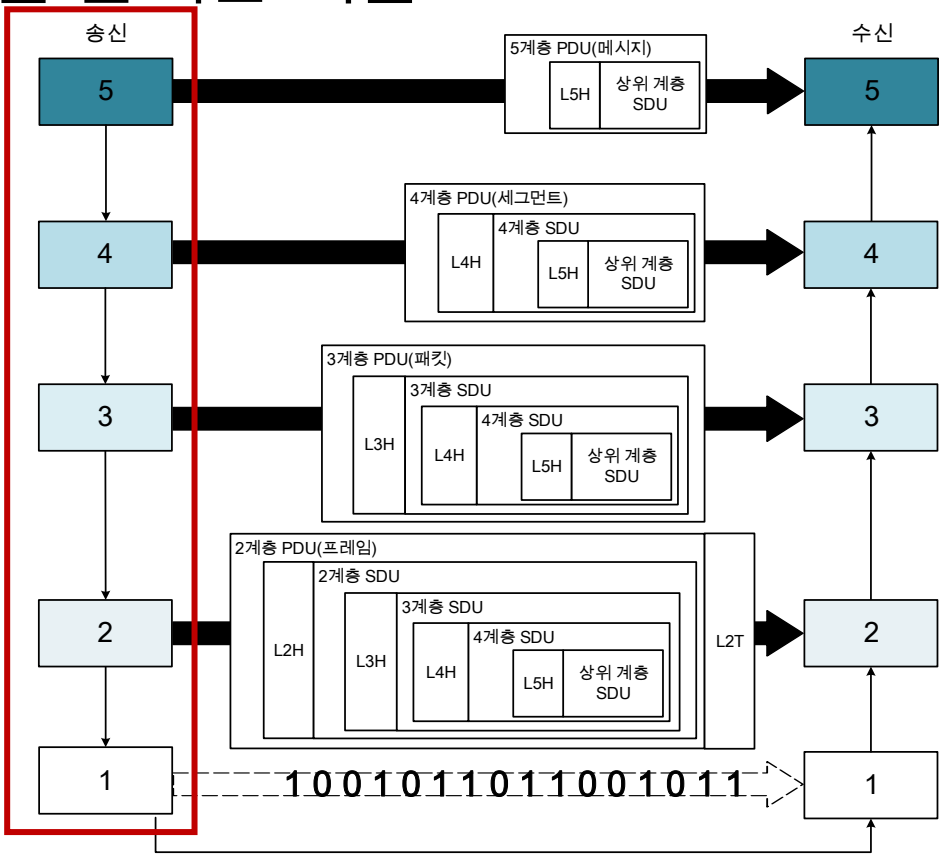
- 프로세스 간 통신(3/3)
  - 포트 번호(Port Number)
    - 종류
      - 잘 알려진 포트 번호(Well-known Port Number)
        - 0 ~ 1023, ICANN(Internet Corporation for Assigned Names and Numbers)에 의해 통제되는 번호
        - 범용적인 TCP/IP 애플리케이션을 위해 예약(고정)해 둔 번호로, 운영체제나 네트워크 프로토콜에 의해 사용됨
        - e.g., 80: HTTP, 443: HTTPS, 143: IMAP4
      - 등록 포트 번호(Registered Port Number)
        - 1024 ~ 49151, ICANN에 등록은 되어 있지만 통제되지 않는 번호
        - 특정 애플리케이션에 대해 등록된 번호로, 일반 사용자 프로세스의 필요에 따라 사용할 수 있음
      - 동적 포트 번호(Dynamic Port Number)
        - 49152 ~ 65535, 임시 또는 개인 포트 번호로서 사용될 수 있는 번호
        - 매번 접속할 때마다 포트 번호가 동적으로 부여됨

전송 계층

- 캡슐화(Encapsulation)
  - 정의
    - 송신 측에서 상위 계층으로부터 전달받은 데이터와 자신의 계층 특성을 담은 헤더를 붙이는 기술

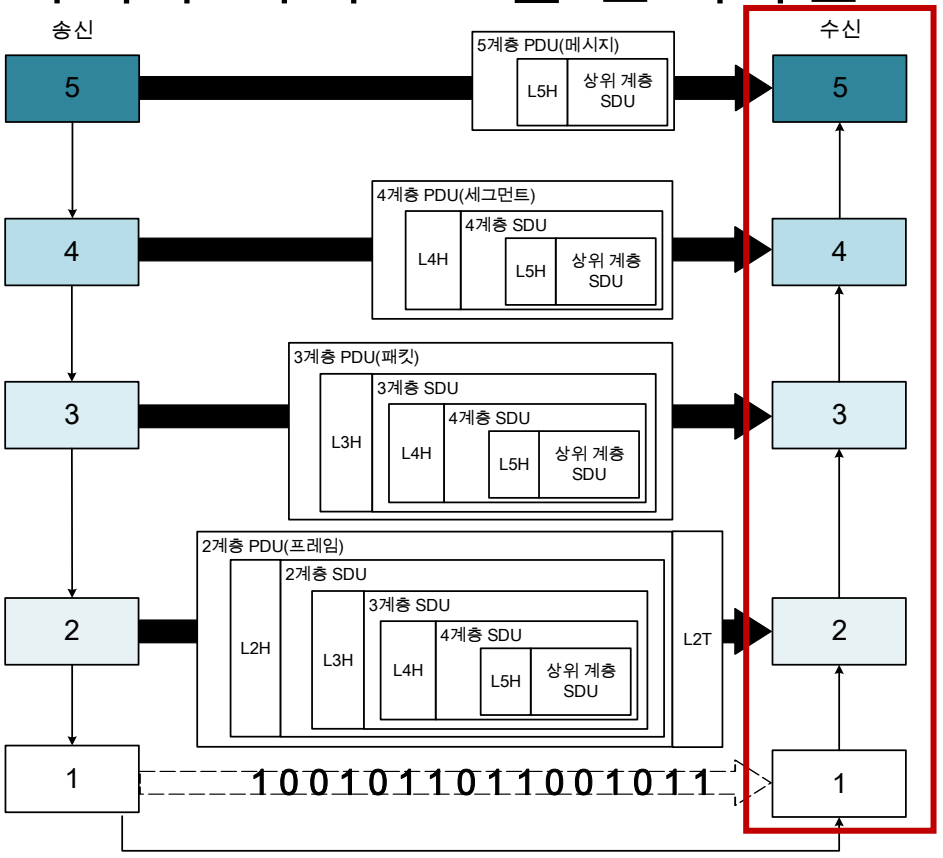
표기	설명
L(Layer)	계층
H(Header)	헤더
T(Trailer)	트레일러

- 프로토콜 데이터 유닛(PDU, Protocol Data Unit)
  동일 계층 간 교환되는 전체 데이터의 양
- 서비스 데이터 유닛(SDU, Service Data Unit)
  상위, 하위 계층 간 전달되는 실제 데이터



전송 계층

- 역캡슐화(Decapsulation)
  - 정의
    - 수신 측에서 하위 계층으로부터 전달받은 데이터에서 헤더와 페이로드를 분리하는 기술

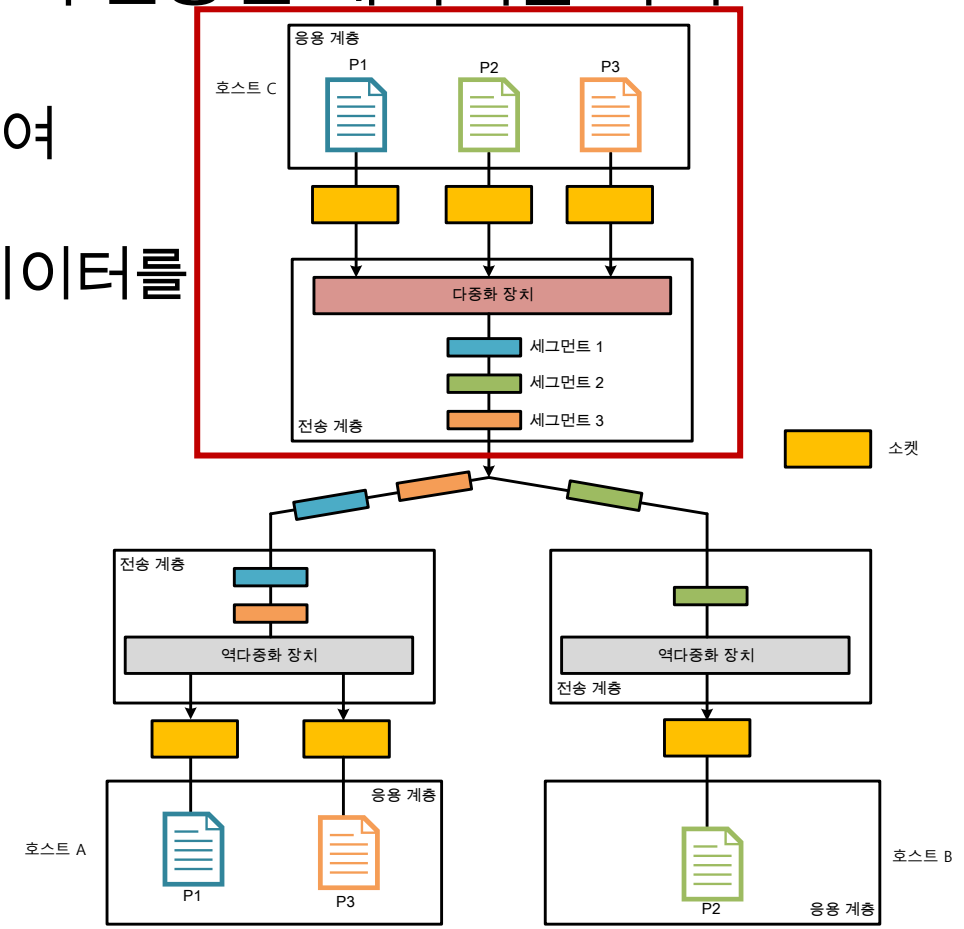


표기	설명
L(Layer)	계층
H(Header)	헤더
T(Trailer)	트레일러

- 프로토콜 데이터 유닛(PDU, Protocol Data Unit)
  동일 계층 간 교환되는 전체 데이터의 양
- 서비스 데이터 유닛(SDU, Service Data Unit)
  상위, 하위 계층 간 전달되는 실제 데이터

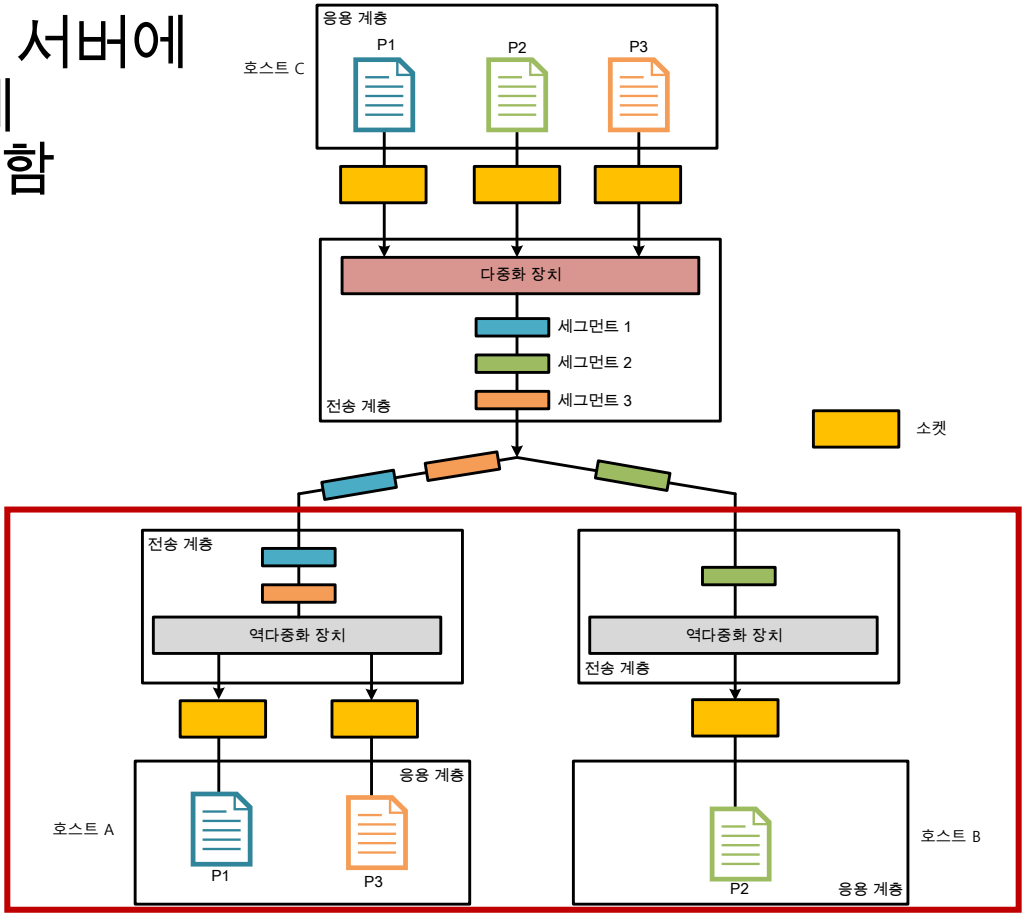
전송 계층

- 다중화(MUX, Multiplexing)
  - 정의
    - 응용 계층의 여러 소켓으로부터 전송된 데이터를 하나로 결합하는 기술
      - 여러 데이터를 동시에 전송하여 전송의 효율성을 높임
      - 여러 개의 소켓에서 전송된 데이터를 다중화하면 세그먼트가 됨



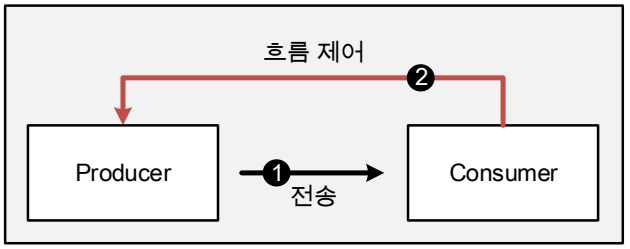
전송 계층

- 역다중화(DeMUX)
  - 정의
    - 결합된 데이터를 여러 개의 독립된 데이터로 분리하는 기술
      - 여러 클라이언트가 하나의 서버에 패킷을 보내면 포트 번호에 대응하는 프로세스로 전달함

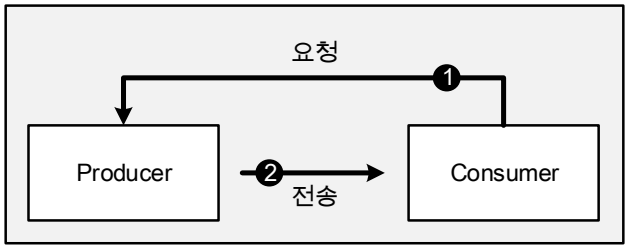


전송 계층

- 흐름 제어(1/3)
  - 데이터 전송 방식
    - Pushing
      - 생산 측에서 정보를 전송하는 경우의 전송 방식
      - 흐름 제어 필요
    - Pulling
      - 소비 측에서 데이터를 요청하여 생산 측에서 전송하는 경우의 전송 방식
      - 흐름 제어 불필요



Pushing



Pulling

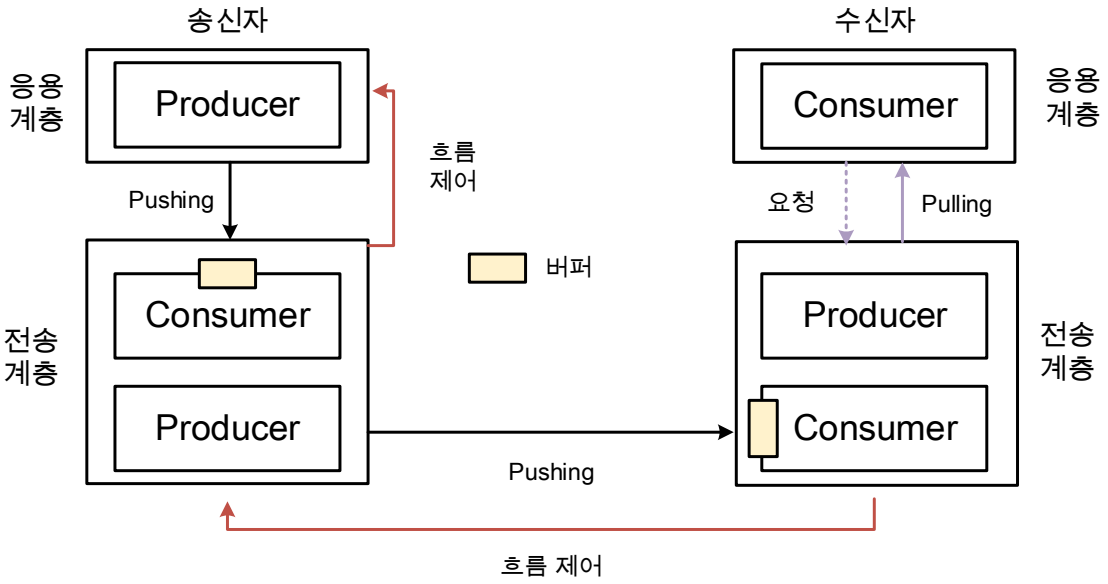
전송 계층

- 흐름 제어(2/3)
  - 정의
    - 송신 측에서 수신 측의 처리 능력을 초과하는 속도로 데이터를 보내지 않도록 속도를 조절하는 매커니즘
  - 기능
    - 데이터 손실 방지
      - 수신 측에 버퍼 오버플로우가 발생하여 패킷이 손실되는 것을 방지
    - 전송 속도 동기화
      - 송신 측과 수신 측간의 속도 차이를 동기화하여 전송 속도를 유지함

전송 계층

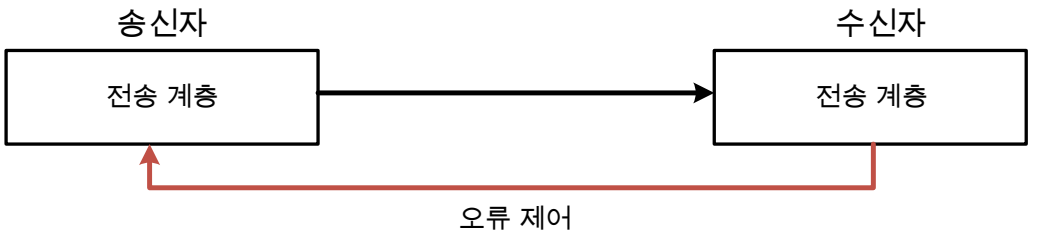
- 흐름 제어(3/3)
  - 필요한 경우
    - 송신 전송 계층에서 다음 메시지를 전송하라는 신호를 응용 계층에게 보내는 경우
    - 수신 전송 계층에서 다음 정보 전송을 위해 송신 전송 계층에게 확인응답을 보내는 경우

- 버퍼
  패킷을 저장할 수 있는 일련의 메모리 영역



전송 계층

- 오류 제어(1/3)
  - 정의
    - 데이터가 유실되거나 잘못된 데이터가 수신될 경우의 대처 및 예방하는 매커니즘
  - 기능
    - 데이터가 유실 및 훼손을 감지함
      - 데이터가 전송되는 동안 오류가 했는지 확인하기 위해 체크섬, 순환 중복 검사 등의 오류 검출 기법을 사용함
    - 데이터를 재전송함
      - 송신 측에서 패킷을 전송한 후 일정 시간 내에 응답이 없으면 해당 패킷이 유실된 것으로 간주하고 재전송을 시도함



전송 계층

- 오류 제어(2/3)
  - 확인응답(ACK, Acknowledgement)
    - 정의
      - 수신자가 전송된 데이터를 성공적으로 받았음을 송신자에게 알리는 메시지
    - 필요한 경우
      - 수신 측에서 패킷이 제대로 도착하지 않았음을 알리는 경우
        - 수신 측에선 패킷의 순서번호에 대응하는 확인응답 번호를 가진 ACK를 송신 측에 전달함
        - 이때, 수신된 것에 대한 패킷에만 ACK를 보내기 때문에 수신 측에 데이터 유실 및 훼손 패킷이 도착할 경우, 이전 순서 번호에 대응하는 ACK를 보내거나 ACK 자체를 보내지 않음

전송 계층

- 오류 제어(3/3)
  - 순서 번호(Sequence Number)
    - 정의
      - 전송된 패킷을 식별하는 일련의 번호
    - 필요한 경우
      - 수신 측에서 어떤 패킷이 재전송되어야 하는지 파악하는 경우
      - 송수신 측에서 어떤 패킷이 중복되었는지, 순서에 어긋나게 도착했는지 파악하는 경우
        - e.g., 순서 번호 1, 2, 3으로 수신 측에 도착하여야 하는 경우, 순서 번호 1, 1, 2로 도착한다면, 1이 중복되었다는 사실을 인지하고 폐기가 가능함

- 범위
  헤더에 포함되는 순서 번호가 m비트라면 순서 번호는 $2^m - 1 (\text{modulo } 2^m)$
  e.g., $m = 4$, 순서 번호 범위 0~15

전송 계층

- 혼잡 제어(1/2)
  - 정의
    - 송신 측에서 수신 측이 수용할 수 있는 패킷의 양보다 많은 패킷을 보내는 현상을 방지하고 제어하는 매커니즘
  - 기능
    - 혼잡 감지 및 방지
      - 혼잡이 발생한 지점을 파악하거나 네트워크 상태를 모니터링하여 혼잡을 미리 방지함
  - 필요한 경우
    - 다수의 송신자가 동시에 데이터를 전송하였을 경우
    - 네트워크 성능 저하 징후가 발생하였을 경우

전송 계층

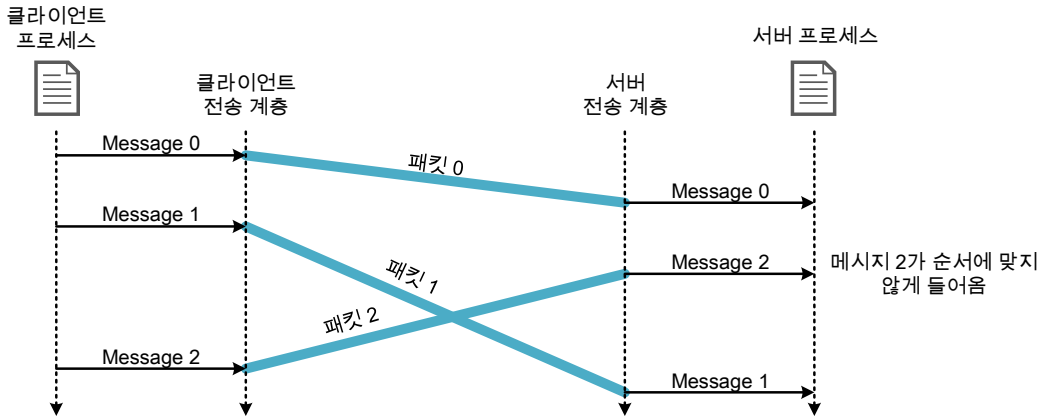
- 혼잡 제어(2/3)
  - 폐 루프 혼잡 제어(Closed-loop Congestion Control)
    - 정의
      - 혼잡이 발생한 후, 혼잡을 완화시키기 위한 혼잡 제어 방식
    - 기능
      - 피드백 기반 전송 조절
        - 수신자의 피드백을 통해 네트워크 상태를 파악하고 이를 바탕으로 전송 속도를 조절하는 매커니즘
      - 동적 윈도우 크기
        - 수신자의 수신 능력에 따라 윈도우의 크기를 동적으로 조절하는 매커니즘

전송 계층

- 혼잡 제어(3/3)
  - 개방 루프 혼잡 제어(Open-loop Congestion Control)
    - 정의
      - 혼잡이 일어나기 전, 혼잡을 방지하기 위한 혼잡 제어 방식
    - 기능
      - 재전송 정책
        - 전송 중 훼손 및 유실된 경우, 해당 패킷을 재전송하기 위해 재전송 타이머를 설정하는 매커니즘
      - 정적 윈도우 크기
        - 고정된 크기의 윈도우를 사용하여 패킷을 전송하는 매커니즘
      - 확인응답 정책
        - 수신자에게 성공적으로 패킷이 도착하였는지를 확인하기 위해 확인응답 값을 설정하는 매커니즘

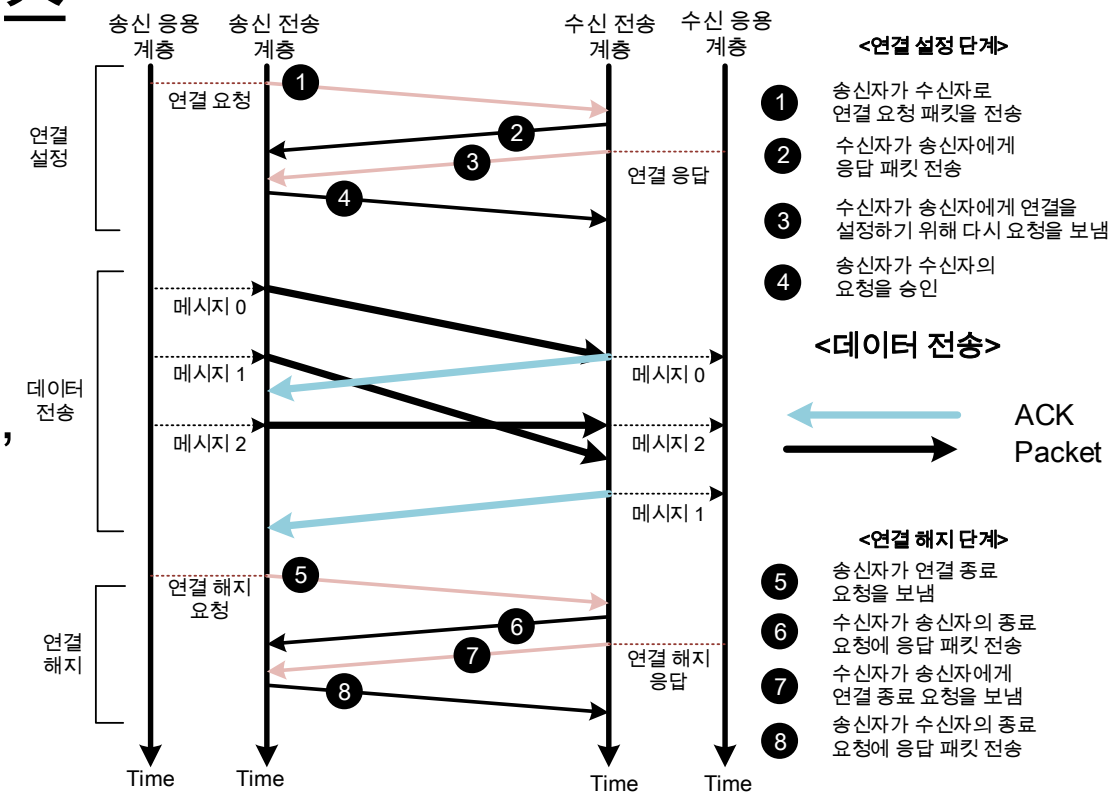
전송 계층

- 비연결형 서비스
  - 정의
    - 데이터 연결을 설정하지 않고 패킷을 전송하는 서비스
  - 특징
    - 패킷의 독립성
      - 여러 개의 메시지 조각들을 전송하는 경우, 패킷 간 연관성이 없기에 순서에 어긋나게 서버 프로세스에 도착할 수 있음
    - 빠른 전송
      - 연결 설정 및 해지 과정이 없기 때문에 데이터 전송이 빠름



전송 계층

- 연결형 서비스
  - 정의
    - 데이터 전송 전에 연결을 설정하고 해당 연결을 통해 데이터를 전송하는 서비스
  - 특징
    - 연결 & 해지 단계 존재
      - 연결이 설정된 후 정보 교환이 가능하며 교환 후, 연결 해지 단계를 거침
    - 신뢰성 보장
      - 데이터 전송 중 손실, 중복, 순서 변경 방지를 위한 메커니즘 제공



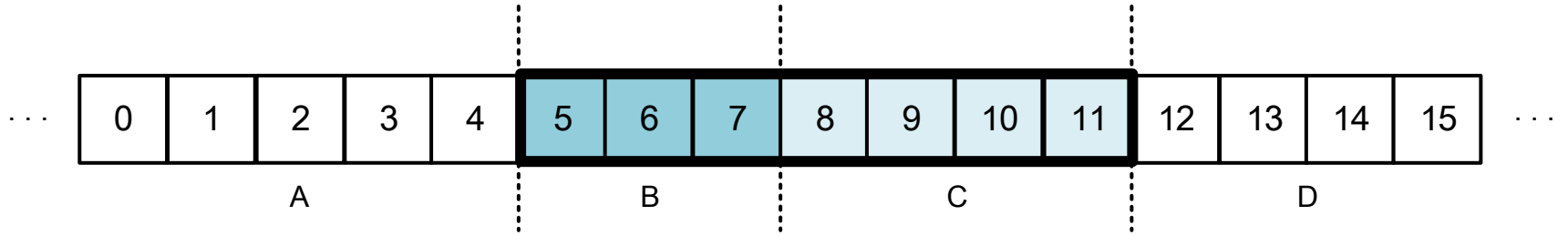
전송 계층

- 슬라이딩 윈도우(Sliding Window)(1/2)
  - 정의
    - 수신 측에서 일정한 윈도우 크기를 설정하고 수신 받은 패킷만큼 윈도우를 하나씩 이동시키는 기법

> • 윈도우
> 메모리 버퍼의 일정 영역

  - 영역

A	송신 후, 확인응답을 받은 상태
B	송신 후, 확인응답을 못 받은 상태
C	송신하진 않았지만, 전송할 수 있는 상태
D	윈도우가 이동할 때까지 송신할 수 없는 상태



전송 계층

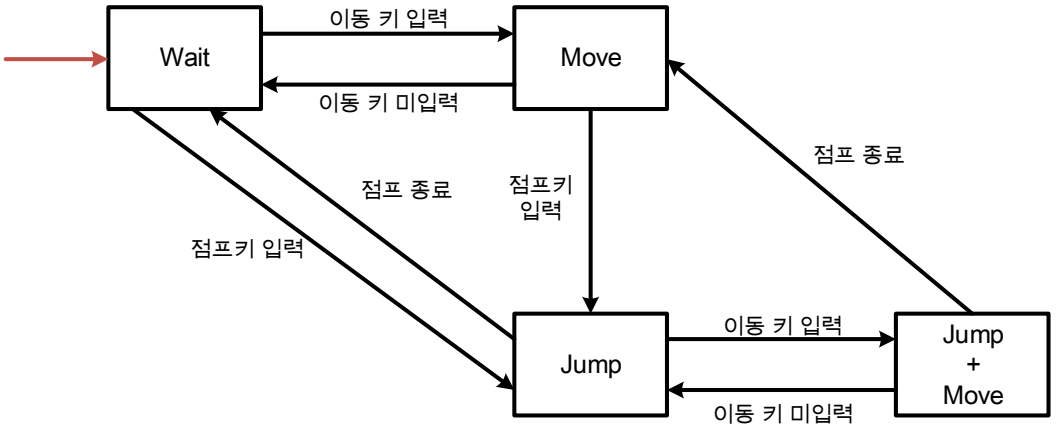
- 슬라이딩 윈도우(Sliding Window)(2/2)
  - 송수신 윈도우
    - 정의
      - 송수신자가 전송 및 수신할 수 있는 데이터의 크기
    - 특징
      - 네트워크 혼잡 상태와 송수신자의 처리 능력에 따라 변함
      - 세 개의 변수로 나뉘어짐(송신: S_f, S_n, S_{size} 수신: R_f, R_n, R_{size})
    - 이동
      - 순서 번호(ackNo)가 S_f과 S_n 사이의 값을 갖는 오류 없는 ACK를 수신하는 송신 윈도우는 하나 이상의 슬롯을 이동시킴

변수	설명
S_f, R_f	송수신 윈도우, 첫 번째(가장 오래된) 미해결 세그먼트
S_n, R_n	송수신 윈도우, 전송 및 수신할 다음 세그먼트
S_{size}, R_{size}	송수신 윈도우, 크기

전송 계층

- 유한 상태 기기(FSM, Finite State Machine)(1/2)
  - 정의
    - 입력을 받아 일련의 상태를 거치면서 출력을 생성해내는 추상적인 모델
      - 유한한 상태개수의 상태를 가진 머신
      - 상태를 기반으로 동작을 제어하는 방식을 구현하기 위해 사용됨
  - FSM 기본 개념

상태	특정 기간에 소프트웨어가 처한 상황
전이	한 상태에서 다른 상태로 이동하는 행위
이벤트	상태 간에 전이하게 만드는 어떠한 일
행동	이벤트에 대한 반응으로, 다른 상태로 전이하기 전에 하는 일



    - e.g., 게임 동작

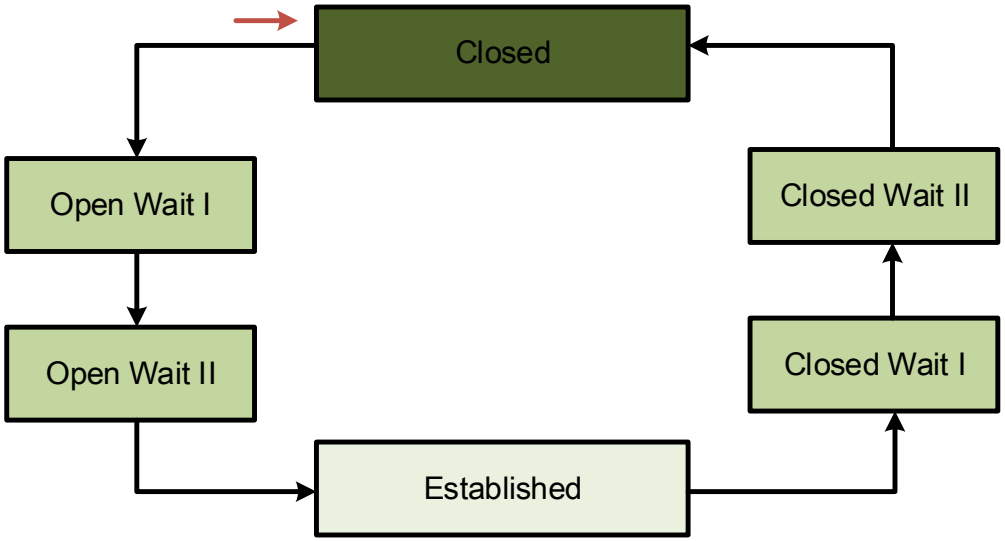
전송 계층

- 유한 상태 기기(FSM, Finite State Machine)(2/2)
  - FSM으로 표현된 비연결형 서비스
    - 한 개의 상태를 지님
      - Established: 데이터 연결 과정없이 데이터를 송수신할 수 있는 상태

송신자와 수신자의 상태

  - FSM으로 표현된 연결형 서비스
    - 세 개의 상태를 지님
      - 'Established', 'Wait', 'Closed'

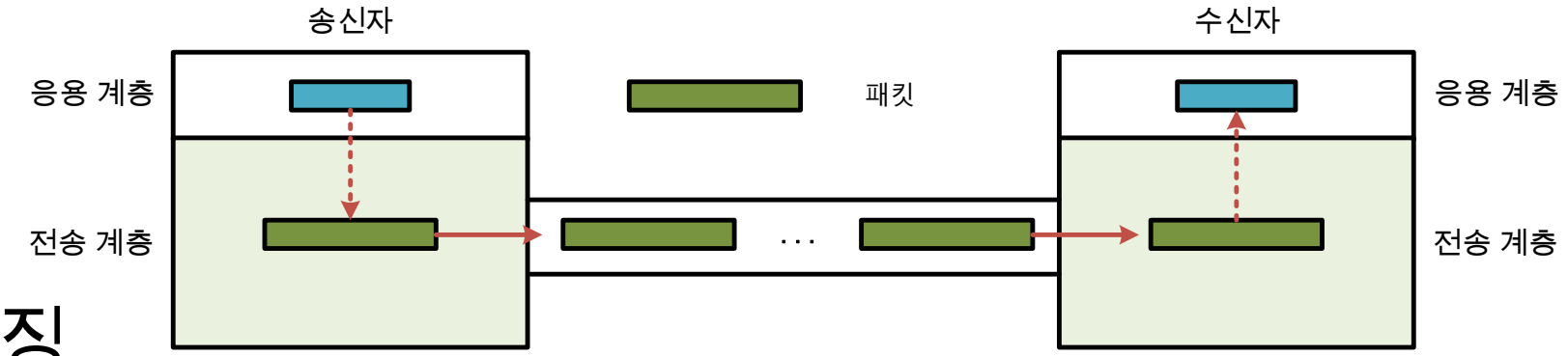
- Closed: 설정된 연결이 없는 상태
- Open Wait I: 연결 요청을 보낸 상태
- Open Wait II: 연결 확인응답을 받은 상태
- Established: 양방향 통신이 가능한 상태
- Closed Wait I: 연결 해제 요청을 보낸 상태
- Closed Wait II: 연결 해제 확인응답을 받은 상태



송신자와 수신자의 상태

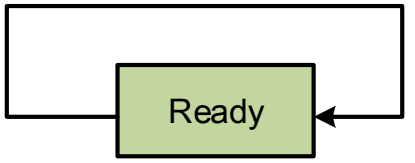
전송 계층

- 단순 프로토콜(Simple Protocol)
  - 정의
    - 흐름 제어 및 오류 제어가 없는 비연결형 프로토콜



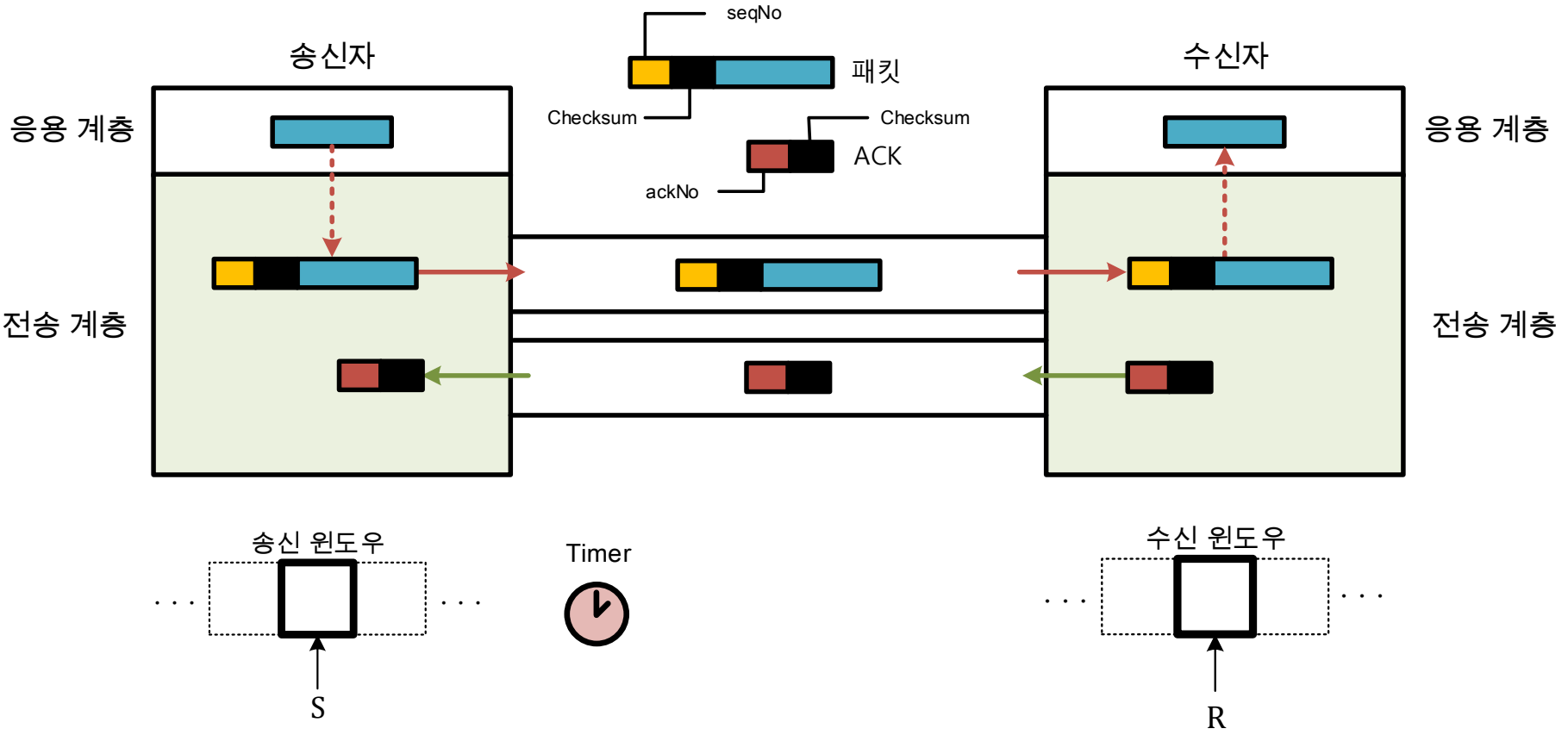
  - 특징
    - 송신 윈도우와 수신 윈도우의 크기가 각각 MTU 크기 1임
      - 수신 측에서 수신한 세그먼트를 즉시 처리할 수 있다고 가정한 모델
  - FSM
    - 하나의 상태를 지님
      - Ready: 세그먼트를 항상 보낼 수 있고, 받을 수 있는 상태

- 송신, 수신 측
  패킷을 항상 보낼 수 있고, 받을 수 있는 상태



전송 계층

- 정지 대기 프로토콜(Stop and Wait Protocol)(1/7)
  - 정의
    - 흐름 제어와 오류 제어가 있는 연결형 프로토콜



전송 계층

- 정지 대기 프로토콜(Stop and Wait Protocol)(2/7)
  - 특징
    - 패킷 훼손 및 손실 여부를 위한 메커니즘 구현
      - e.g., 검사합, 재전송, 확인응답
    - 확인응답을 받기전까지 다음 패킷을 전송하지 않기 때문에 비효율적임
      - 대역폭 지연 곱이 클수록 비효율적임
        - 대역폭 지연 곱 = 대역폭 × 왕복 시간
    - 송신 윈도우와 수신 윈도우의 크기가 각각 MTU 크기 1임
      - 송신 측이 한 번에 하나의 패킷을 전송하고 확인응답이 들어오기 전까지 다음 패킷을 전송하지 않음

- 정지 대기 프로토콜(Stop and Wait Protocol)(3/7)
  - 예제 13.1

전송 대기 시스템에서 회선의 대역폭이 1Mbps이고 1비트가 왕복하는 데 20m/s가 걸린다고 가정해보자. 대역폭 지연 곱은 얼마인가? 시스템의 패킷이 1000비트 길이이면 회선의 이용률은 얼마인가?

- 풀이
  - 대역폭 지연 곱 = 대역폭 x 왕복 시간 = $(1 \times 10^6) \times (20 \times 10^{-3}) = 20000$
  - 패킷이 전송되고 확인응답까지 돌아오는데 걸리는 시간 동안 20000비트를 전송할 수 있음
  - $1000/20000 = 0.05$

  $\therefore 5\%$

- 정지 대기 프로토콜(Stop and Wait Protocol)(4/7)
  - 예제 13.2

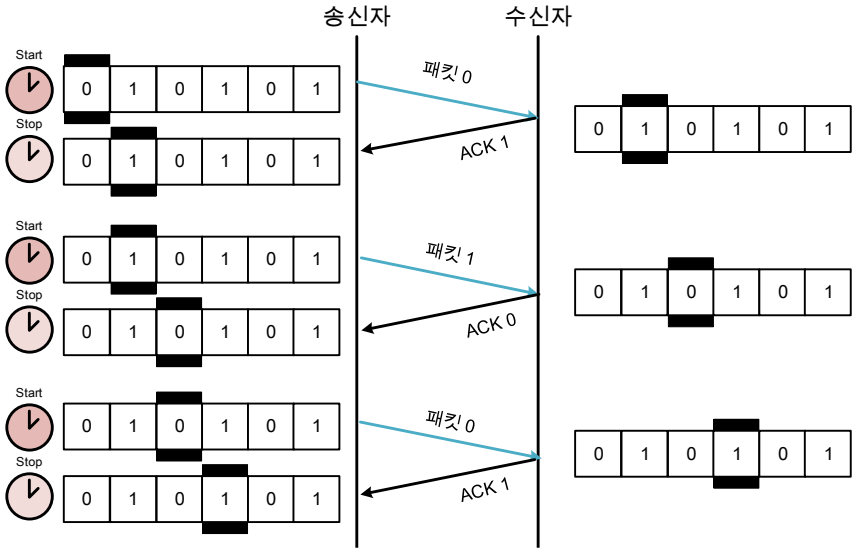
예제 13.1에서 확인응답에 대한 고려없이 최대 15개의 패킷을 전송할 수 있는 프로토콜을 이용하면 회선의 이용률은 얼마인가?

- 풀이
  - 시스템 왕복 시간동안 20000비트를 전송할 수 있기 때문에 15개의 패킷인 15000비트를 전송할 수 있음
  - $15000/20000 = 0.75$

  $\therefore 75\%$

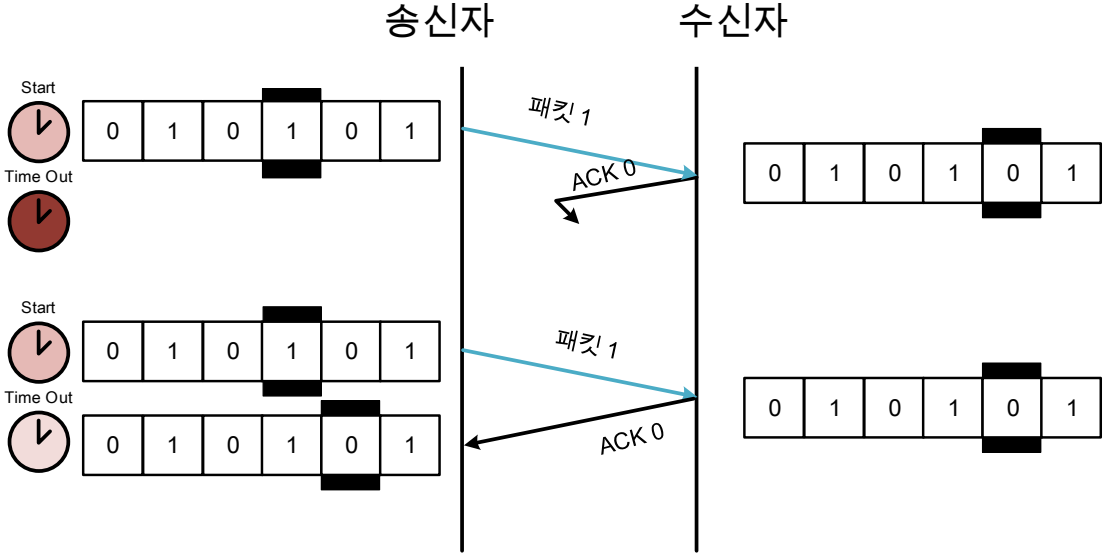
전송 계층

- 정지 대기 프로토콜(Stop and Wait Protocol)(5/7)
  - 과정
    - 패킷이 성공적으로 수신 측에 전송되었을 경우
      - 순서 번호 x를 전송한다고 가정
      - 송신 측에 확인응답이 도착하면 $x+1$의 순서 번호를 갖는 다음 패킷 전송
      - x와 $x+1$의 순서 번호를 갖는 패킷이 모두 확인응답 되면, 그 다음 패킷은 다시 x의 순서 번호를 갖더라도 송수신 양쪽에서 명확하게 구분할 수 있음
      - 즉, $x=0, x+1=1$로 설정될 수 있고, $0,1,0,1,0,1$의 순서를 가짐 (따라서 modulo 2의 연산)



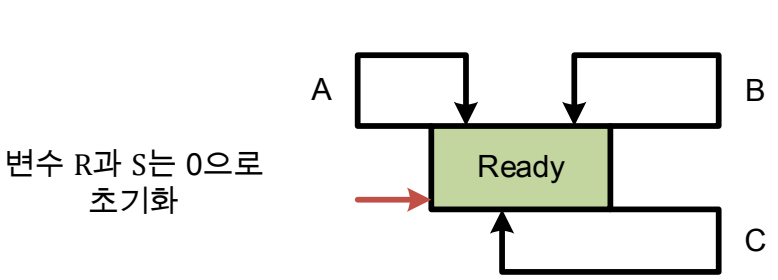
전송 계층

- 정지 대기 프로토콜(Stop and Wait Protocol) (6/7)
  - 과정
    - 패킷이 수신 측에 도착하지 않았을 경우
      - 송신 측은 타이머 만료 이후, x의 순서 번호를 갖는 패킷을 재전송
    - 패킷이 수신 측에 도착하였지만 훼손이 일어났을 경우
      - 송신 측은 타이머 만료 이후, x의 순서 번호를 갖는 패킷을 재전송
      - 수신 측은 x의 순서 번호를 갖는 패킷을 다시 한 번 전달받은 것이므로 중복 패킷이라는 사실 인지 후 폐기



전송 계층

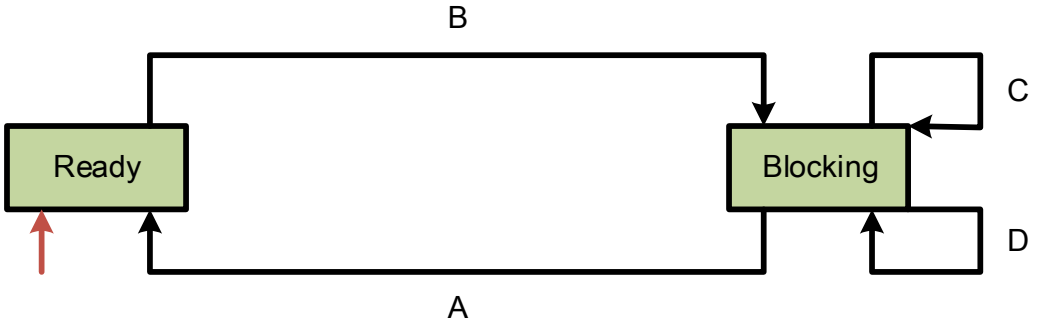
- 정지 대기 프로토콜(Stop and Wait Protocol)(7/7)
  - FSM
    - 송신 측은 두 개의 상태를, 수신 측은 하나의 상태를 지님
      - ‘Ready’, ‘Blocking’



수신 측

<Ready>
- A: SeqNo R이 올바르게 왔을 경우, R=(R+1) modulo 2의 ACK 전송
- B: 기대했던 seqNo가 아닌 경우, 기대했던 ackNo 전송
- C: 훼손되거나 중복된 세그먼트 폐기

- Blocking
  데이터 송수신을 받지 않고 기다리거나
  데이터를 처리하고 있는 상태



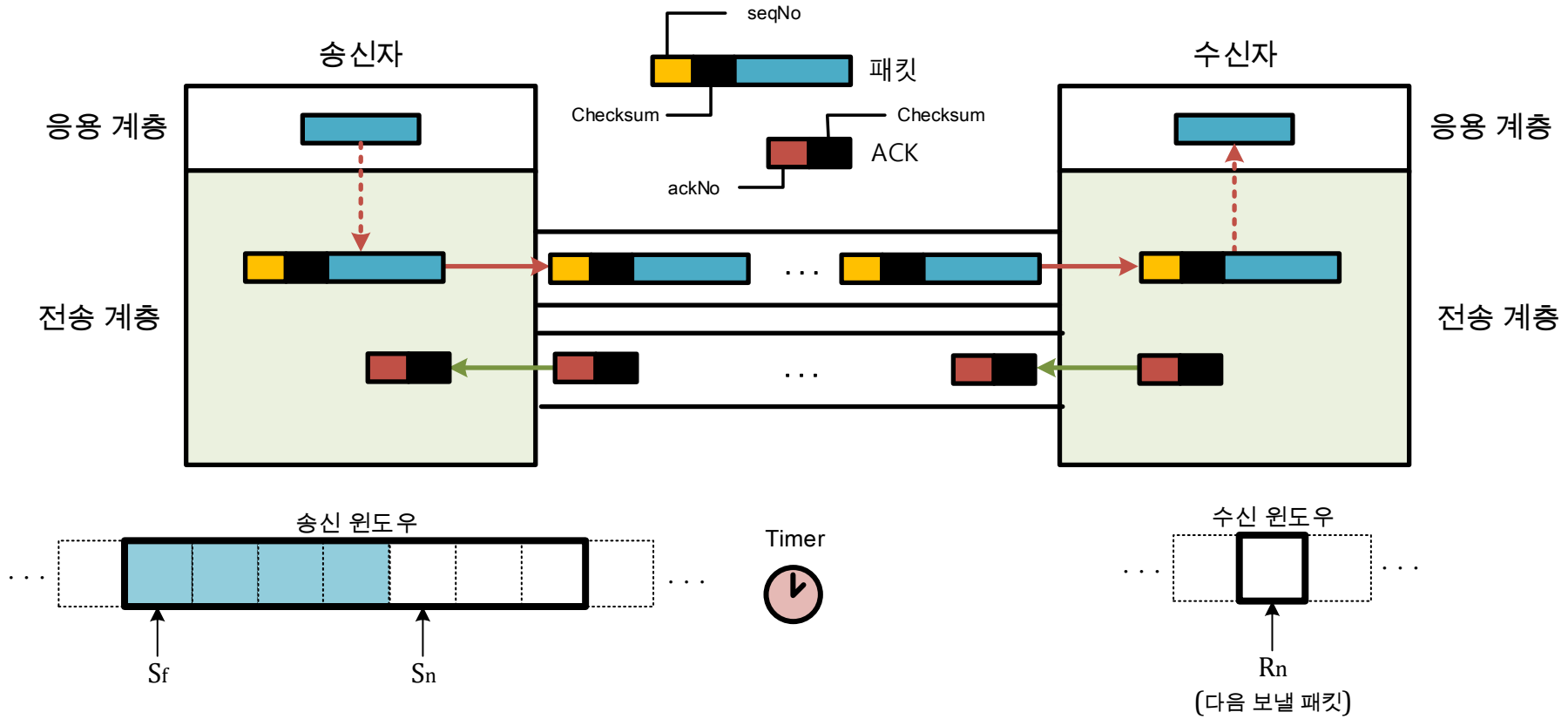
송신 측

<Ready>
- B: 응용 계층으로부터 요청이 들어오면 송신 측은 순서 번호를 S로 설정한 패킷을 만듦 / 송신 측은 타이머를 구동하고 Blocking 상태로 전이

<Blocking>
- A: 알맞은 ackNo((S+1) module 2)를 가진 패킷이 도착했을 경우, 타이머를 종료함 / 윈도우 S=(S+1) module 2로 변경하고 Ready 상태 전이
- C: 수신된 ACK가 훼손됐거나 기대했던 ackNo 아닐 경우, ACK 폐기
- D: 타임아웃이 발생하면, 송신 측은 패킷을 재전송하고 타이머를 재구동

전송 계층

- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - 정의
    - 송신 측이 확인응답을 기다리는 동안 여러 개의 패킷을 전송할 수 있는 프로토콜

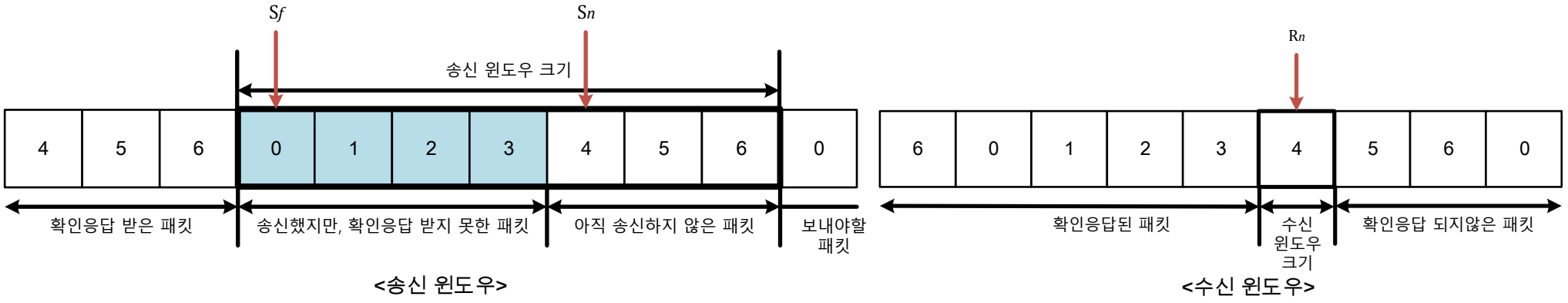


전송 계층

- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - 특징(1/3)
    - 확인응답 값은 누적된 값이며 수신되기를 기대하는 다음 패킷의 순서 번호를 의미함
      - e.g., ackNo가 7일 시, 0부터 6까지 순서 번호를 가지는 패킷이 모두 성공적으로 도착했음을 의미
      - e.g., ackNo가 2가 손실되었을 경우, 타이머 만료 전에 송신 측에 이후의 ackNo가 도착한다면 2가 도착했음을 의미
    - 정지 대기 프로토콜보다 전송 효율이 뛰어남
      - 정지 대기 프로토콜은 하나의 패킷을 보내고 확인응답을 받기 전까지 다른 패킷을 보낼 수 없음

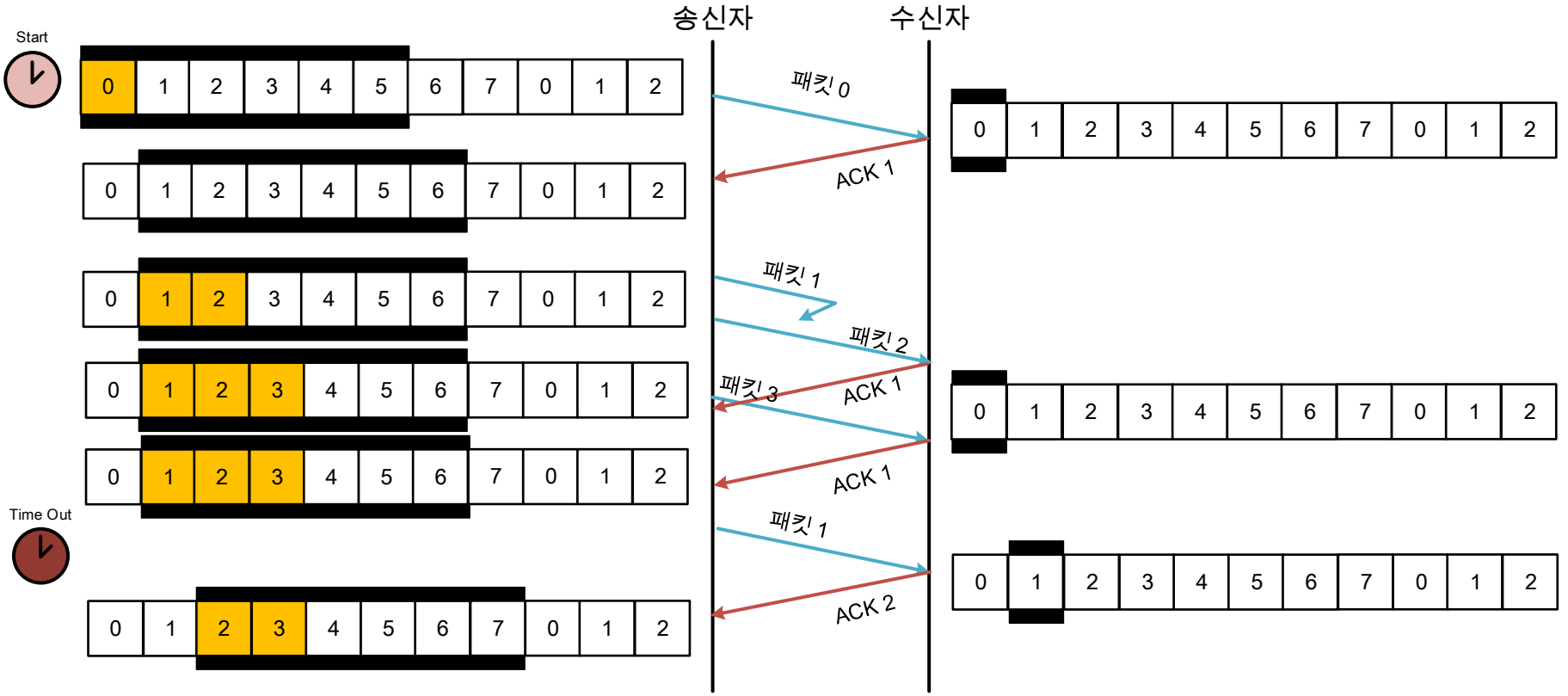
전송 계층

- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - 특징(2/3)
    - 송신 윈도우의 최대 크기가 $2^m - 1$ (m은 순서 번호의 비트 수)
      - 수신 윈도우의 크기는 1임



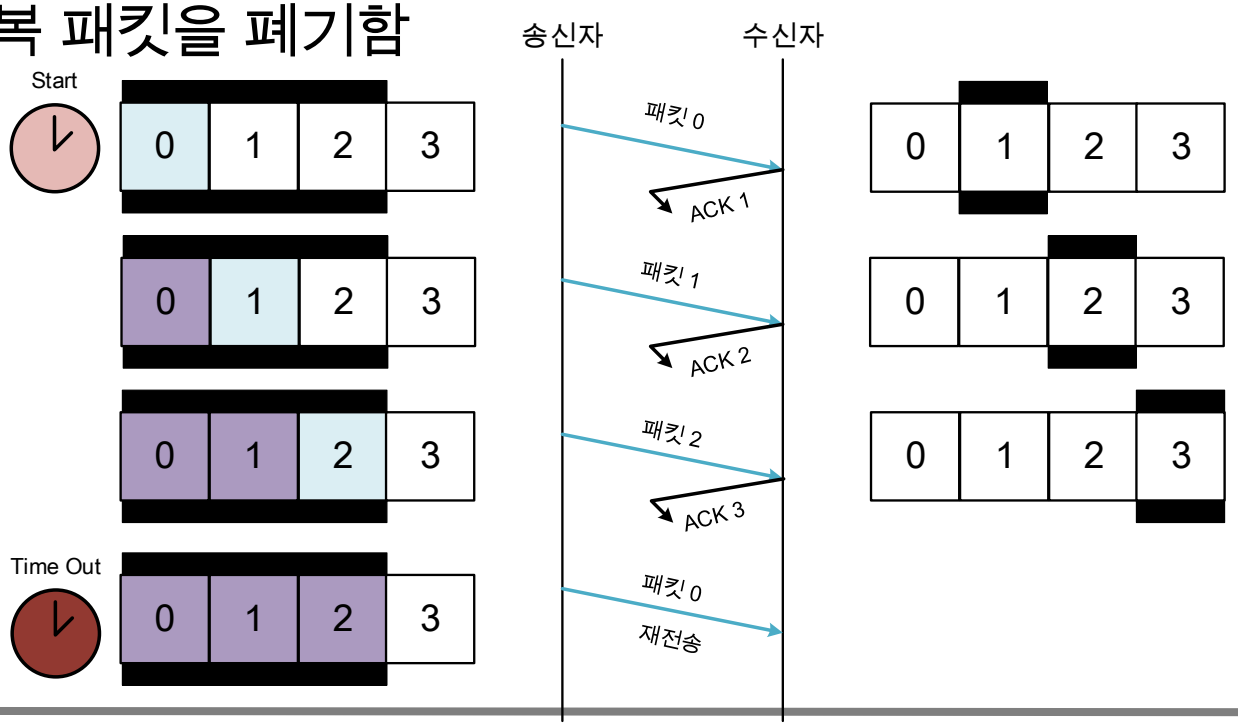
전송 계층

- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - 특징(3/3)
    - 수신 측에 패킷 n이 도착하지 않고 n+1 이후의 패킷이 온다면 ACK n을 송신 측에 보내며 n+1 이후의 패킷을 폐기함



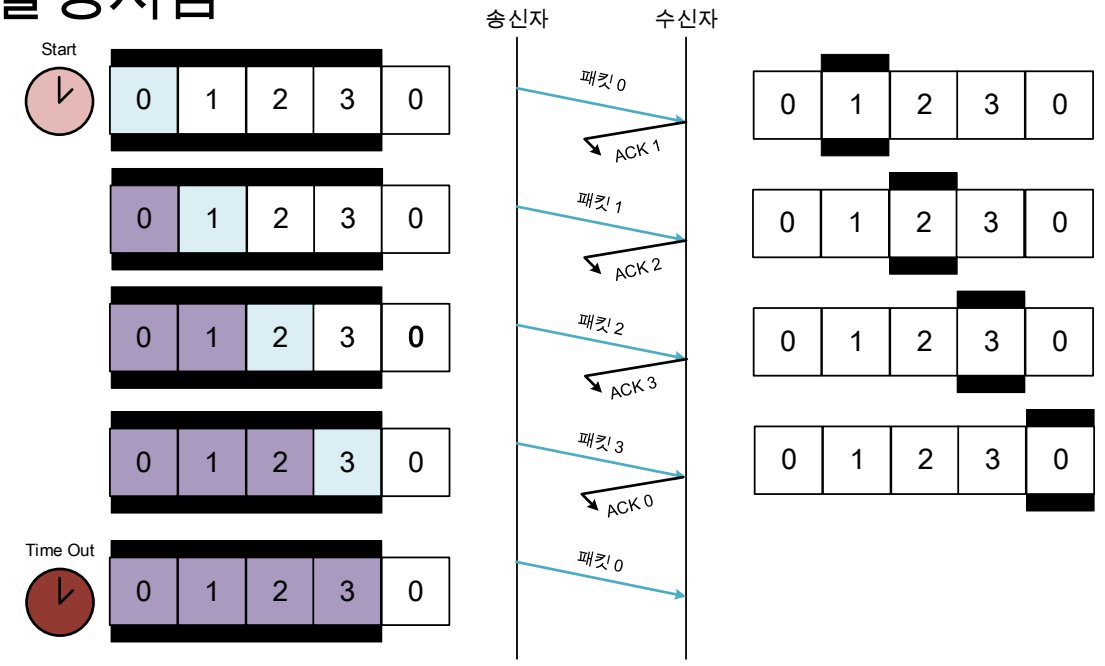
전송 계층

- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - (증명) 송신 윈도우 크기가 2^m보다 작아야 함(1/2)
    - m이 2이고, 윈도우 크기가 3일 경우
      - 확인응답 3개가 손실되었다면, 타이머가 만료된 후 패킷 0, 1, 2을 재전송함
      - 수신 측은 새로운 패킷 0을 원했으나 중복 패킷 3개가 도착하였으므로 중복 패킷을 폐기함



전송 계층

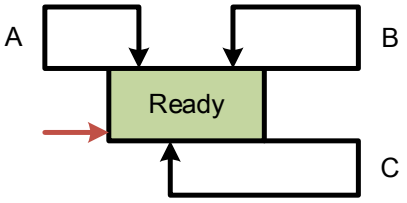
- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - (증명) 송신 윈도우 크기가 2^m보다 작아야 함(2/2)
    - m이 2이고, 윈도우 크기가 4일 경우
      - 확인응답 4개가 손실되었다면, 타이머 만료된 후 패킷 0, 1, 2, 3을 재전송함
      - 수신 측은 다시 온 패킷 0을 중복 패킷이 아닌 새로운 패킷을 받아들여 오류를 발생시킴



전송 계층

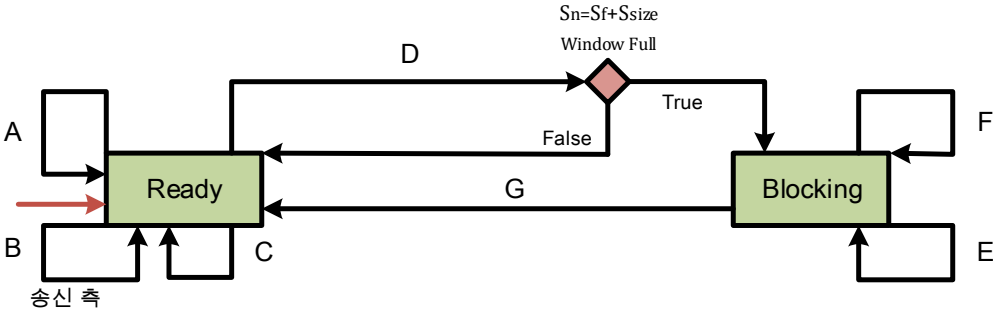
- N-프레임-후퇴 프로토콜(GBN, Go-Back-N Protocol)
  - FSM
    - 송신 측은 두 개의 상태를, 수신 측은 하나의 상태를 지님
      - ‘Ready’, ‘Blocking’

변수 Sf과 Sn는 0으로 초기화



수신 측

<Ready>
- A: SeqNo Rn이 올바르게 왔을 경우, 응용 계층에 정보를 전달하고 윈도우를 이동함(Rn=(Rn+1) modulo 2^m) / 마지막을 ackNo Rn을 전송
- B: 윈도우를 벗어난 seqNo가 올 경우, 해당 패킷을 폐기하고 ackNo=Rn인 패킷 전송
- C: 훼손되거나 중복된 패킷을 폐기

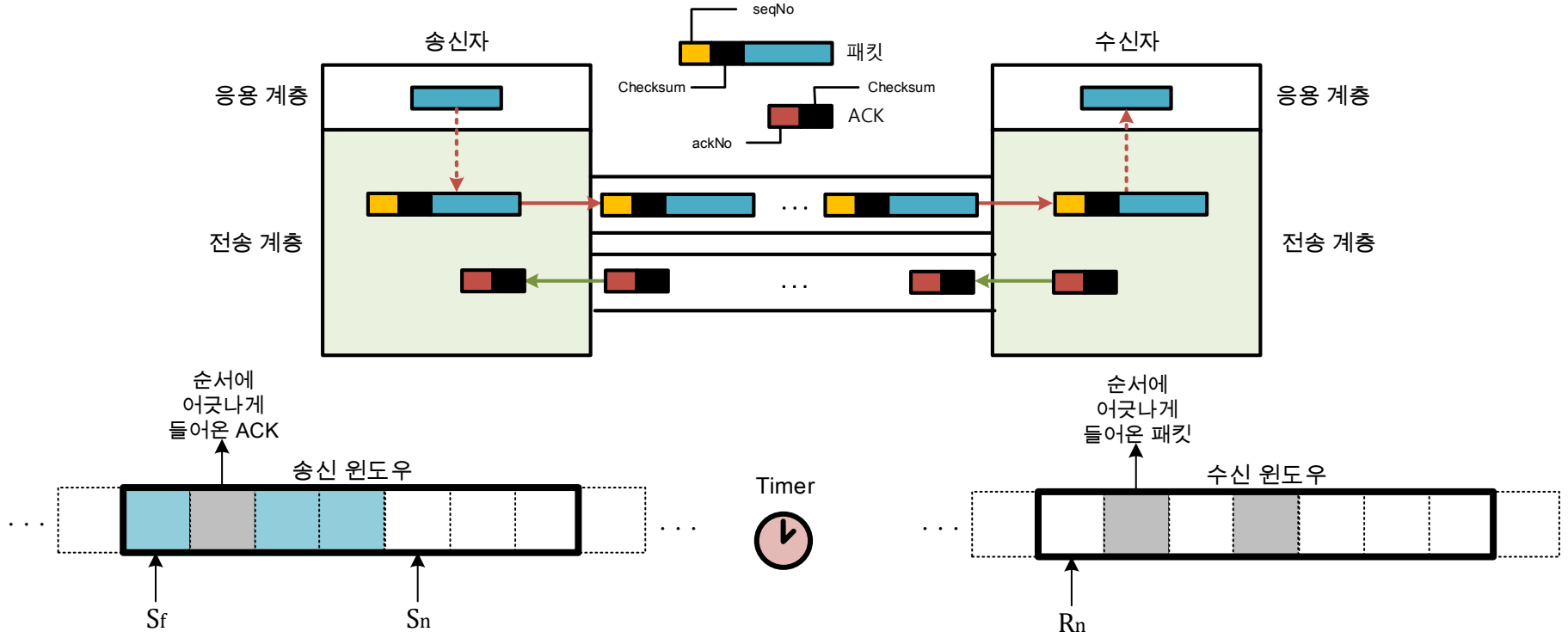


<Ready>
- D: 응용 계층으로부터 요청이 들어오면 송신 측은 순서 번호를 Sn로 설정한 패킷을 만듦 / 송신 측은 타이머를 구동하고 Sn = (Sn+1) module 2^m으로 증가(Sn = (Sf+$Ssize$) module 2^m 으로 꽉 찼다면 Blocking 상태로 이동)
- C: 미해결 패킷에 해당하는 ACK가 들어올 경우, 윈도우를 이동하고 타이머를 중단함 (미해결 패킷이 아직 남아있다면 타이머를 재구동)
- B: 수신한 ACK가 훼손되었거나 기대했던 ackNo가 아닐 경우, ACK 폐기
- A: 타임아웃이 발생하는 경우, 송신 측은 패킷을 재전송하고 타이머를 재구동

<Blocking>
- G: 미해결 패킷에 해당하는 ACK가 들어올 경우, 윈도우를 이동하고 타이머를 중단
- / 미해결 패킷이 아직 남아있다면 타이머를 재구동하고 Ready 상태로 전이
- E: 수신된 ACK가 훼손되었거나 기대했던 ackNo가 아닐 경우, ACK 폐기
- F: 타임아웃이 발생하는 경우, 송신 측은 패킷을 재전송하고 타이머를 재구동

전송 계층

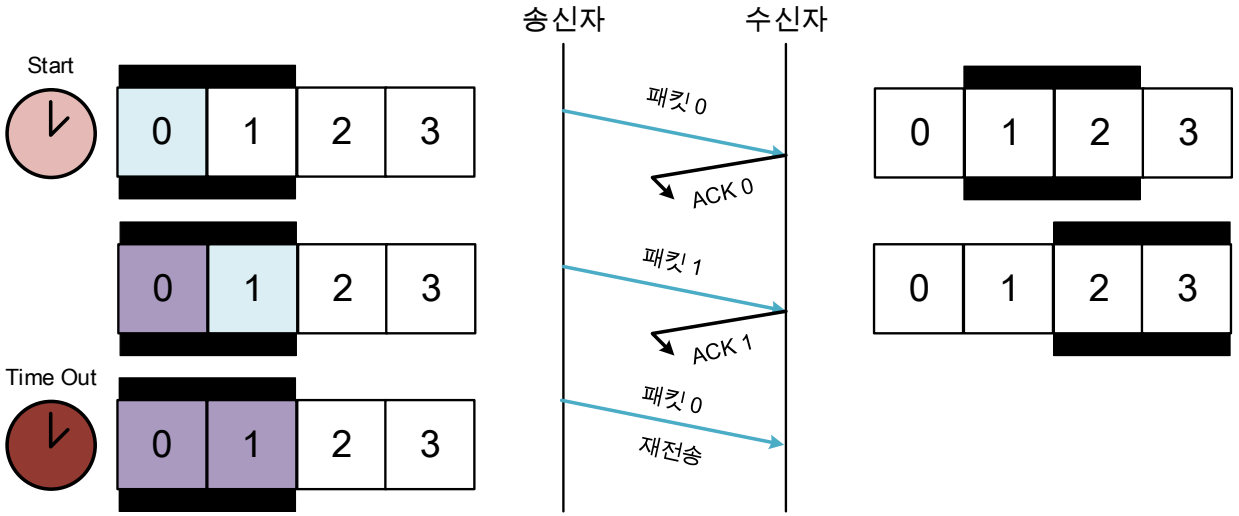
- 선택적 반복 프로토콜(SR, Selective Repeat Protocol)(1/6)
  - 정의
    - 송신 측이 여러 패킷을 동시에 전송하고, 수신 측이 수신한 패킷 중 오류가 발생한 패킷만 재전송을 요청하는 프로토콜



전송 계층

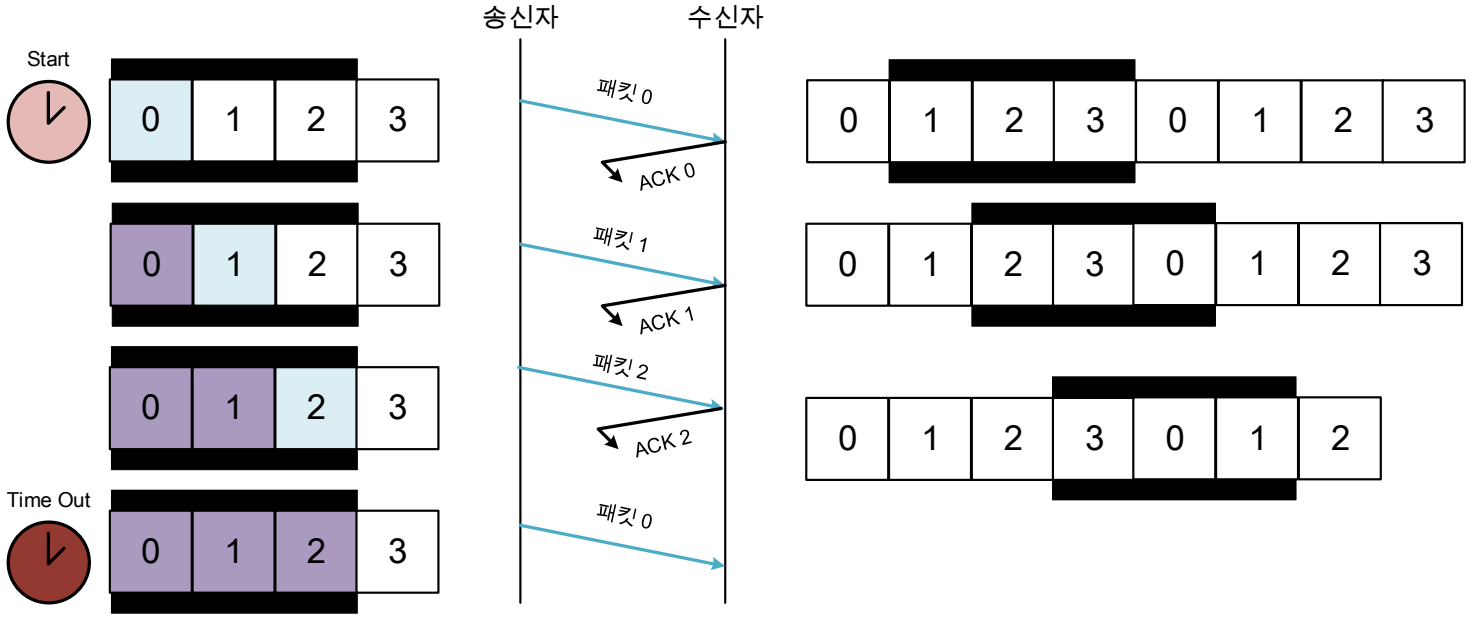
- 선택적 반복 프로토콜(SR, Selective Repeat Protocol)(2/6)
  - 특징
    - 송신 및 수신 윈도우의 최대 크기는 2^{m-1} (m은 순서 번호의 비트 수)
    - 미해결 패킷 하나마다 타이머를 설정해 독립적으로 처리
      - N-프레임 후퇴 프로토콜은 미해결 패킷을 하나의 그룹으로 처리
    - 확인응답 번호는 오류 없이 수신된 하나의 단일 패킷의 순서 번호를 의미함

- 선택적 반복 프로토콜(SR, Selective Repeat Protocol)(3/6)
  - (증명) 송신 및 수신 윈도우 크기가 2^{m-1}이어야 함(1/2)
    - m=2이고 윈도우 크기가 2일 때
      - 모든 확인응답이 손실되고 패킷 0에 대한 타이머가 만료된 경우, 송신 측이 패킷 0을 전달하지만 수신 측은 ACK 2를 보내며 패킷 2를 원하는 것을 표현
      - 송신 측이 보낸 중복 패킷 0은 폐기함



전송 계층

- 선택적 반복 프로토콜(SR, Selective Repeat Protocol)(4/6)
  - (증명) 송신 및 수신 윈도우 크기가 2^{m-1}이어야 함(2/2)
    - m=2이고 윈도우 크기가 3일 때
      - 모든 확인응답이 손실되고 패킷 0에 대한 타이머가 만료된 경우, 수신 측에선 재전송된 패킷 0을 새로운 패킷으로 받아들임



- 선택적 반복 프로토콜(SR, Selective Repeat Protocol)(5/6)
  - 예제 13.3

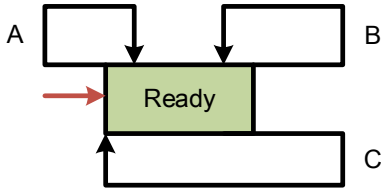
송신 측에서 패킷 0, 1, 2, 3, 4, 5의 6개의 패킷을 전송한다고 가정할 때, 송신 측이 ackNo=3을 수신하였다. 시스템이 GBN 또는 SR을 사용하는 경우에 ACK 수신은 무엇을 의미하는가?

- 풀이
  - GBN
    - 0, 1, 2의 3개 패킷을 훼손되지 않고 잘 수신되었음을 의미하고, 다음 패킷 3을 기다리고 있음
  - SR
    - 패킷 3이 훼손되지 않고 잘 수신되었음을 의미하고, 다른 패킷에 대한 정보는 없음

전송 계층

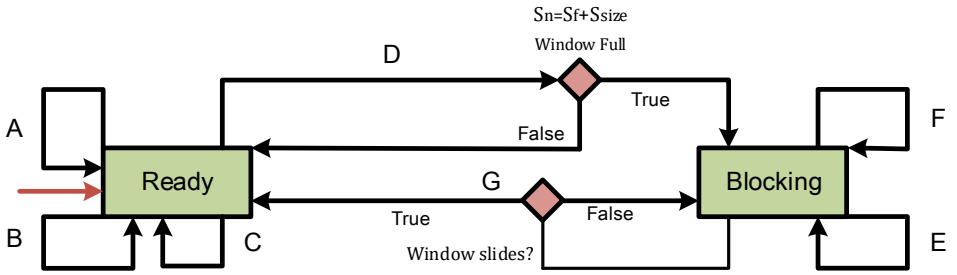
- 선택적 반복 프로토콜(SR, Selective Repeat Protocol)(6/6)
  - FSM
    - 송신 측은 두 개의 상태를, 수신 측은 하나의 상태를 지님
      - ‘Ready’, ‘Blocking’

변수 Sf과 Sn는 0으로 초기화



수신 측

<Ready>
- A: seqNo가 올바르게 왔을 경우, ackNo=seqNo를 가진 ACK가 전송됨 / seqNo=Rn이면, 해당 패킷과 이전에 도착한 연속적인 패킷은 응용 계층으로 전달되며 Rn은 빈 슬롯 첫번째를 가르킴
- B: 윈도우를 벗어난 seqNo가진 패킷이 올 경우, 해당 패킷을 폐기하고 ackNo=Rn인 패킷을 송신 측에 전송함(seqNo < Rn을 가진 패킷과 관련된 ACK가 손실되는 경우에 송신 측에서 윈도우를 이동할 수 있도록 하기 위하여 필요함)
- C: 훼손되거나 중복된 패킷 폐기



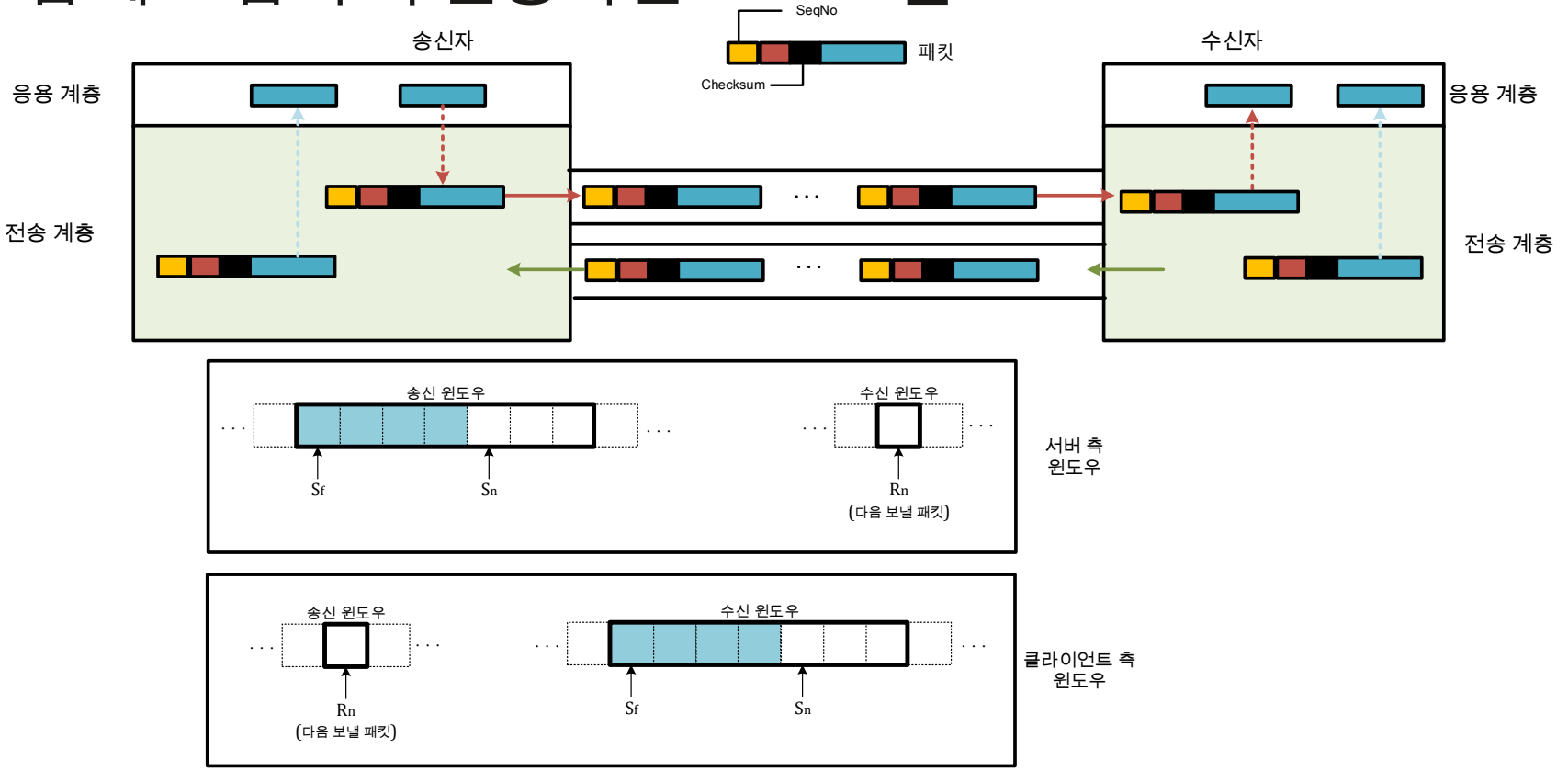
송신 측

<Ready>
- D: 응용 계층으로부터 요청이 들어오면 순서 번호를 Sn으로 설정된 패킷을 만듦 / 송신 측은 타이머를 구동하고 $Sn = (Sn+1) \bmod 2^m$으로 증가($Sn = (Sf+Ssize) \bmod 2^m$ 으로 꽉 찼다면 Blocking 상태로 이동)
- C: 미해결 패킷에 해당하는 ACK가 들어올 경우, 해당 패킷은 확인응답 표시가 됨 / 만일 ackNo=Sf이면, 윈도우는 Sf가 첫 번째 미확인 패킷을 지시하는 곳으로 이동(미해결 패킷이 아직 남아있다면 타이머를 재구동)
- B: 수신한 ACK가 훼손되었거나 기대했던 ackNo가 아닐 경우, ACK 폐기
- A: 타임아웃이 발생하는 경우, 미해결 패킷 모두 재전송하고 타이머 재구동

<Blocking>
- G: 미해결 패킷에 해당하는 ACK가 들어올 경우, 해당 패킷은 확인응답 표시가 됨 / Sf = ackNo이면, 윈도우는 Sf를 첫 번째 미확인 패킷을 지시하는 곳까지 이동 / 윈도우가 오른쪽으로 이동하는 경우, 송신 측은 Ready 상태로 전이
- E: 수신된 ACK가 훼손되었거나 기대했던 ackNo가 아닐 경우, ACK 폐기
- F: 타임아웃이 발생하는 경우, 송신 측은 패킷을 재전송하고 타이머를 재구동

전송 계층

- 양방향 프로토콜: 피기배킹(Piggybacking)(1/2)
  - 정의
    - 송신자가 데이터를 전송할 때 수신자로부터의 응답 메시지를 함께 포함하여 전송하는 프로토콜



전송 계층

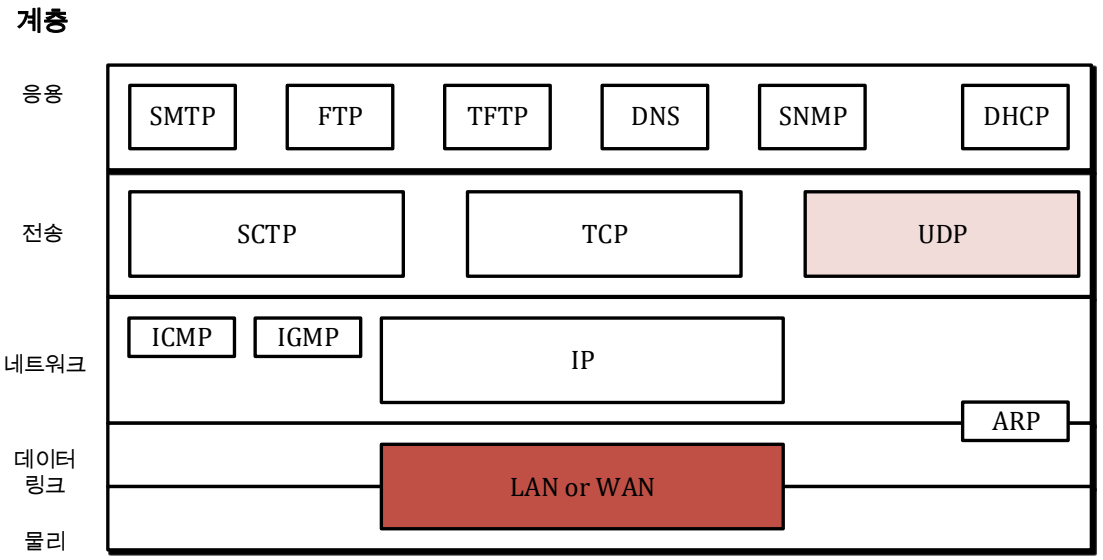
- 양방향 프로토콜: 피기배킹(Piggybacking)(2/2)
  - 특징
    - 고정적인 송수신 호스트의 구분 없이 양방향으로 데이터와 확인응답을 교차하며 정보를 전송할 수 있음
      - 응답 패킷의 전송 횟수를 감소시켜 전송 효율을 증가시킬 수 있음
    - 데이터 전송 지연 및 재전송으로 인한 혼잡 상태를 막기 위해 짧은 시간의 타이머를 사용함
    - 응답 메시지와 데이터를 동시에 전달하기 때문에 프로토콜 구현의 복잡성이 증가할 수 있음

목 차

- 전송 계층
- UDP(User Datagram Protocol)

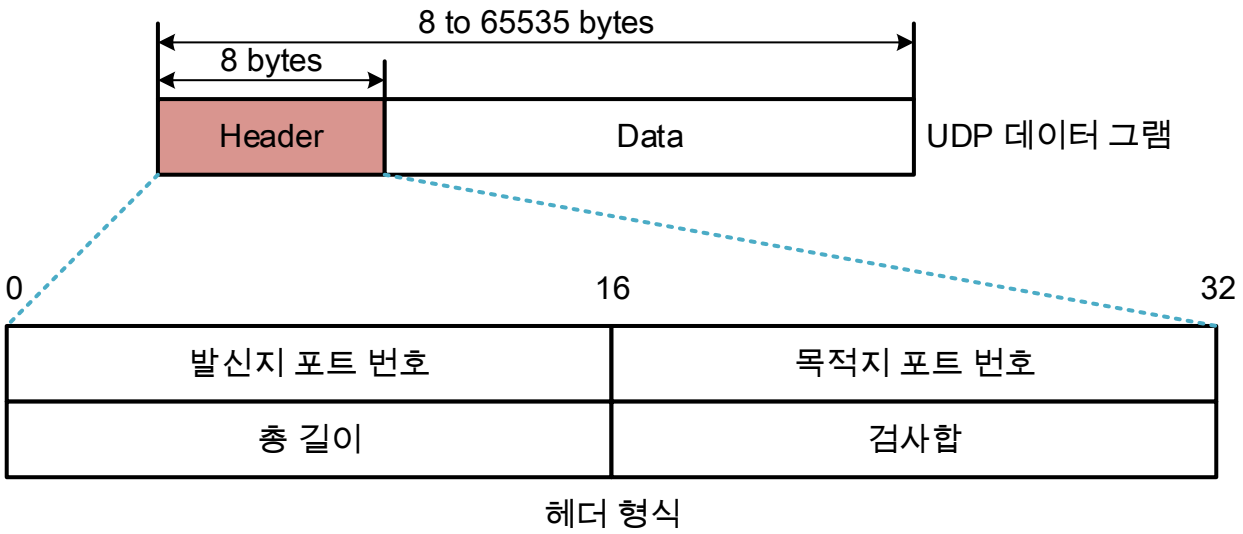
UDP

- 정의
  - 전송 계층에서 사용되는 비연결형 프로토콜
- 특징
  - 신뢰성 부족
    - 흐름 제어, 혼잡 제어, 확인응답 기능 결여
  - 빠른 전송 속도
    - 연결 설정 및 오류 검출과 같은 오버헤드가 없음



UDP

- 데이터그램
  - 포맷



헤더 형식

필드	크기(Byte)	설명
발신지 포트 번호	2	발신지 호스트 상에서 수행되는 프로세스에 의해 사용되는 포트 번호
목적지 포트 번호	2	목적지 호스트 상에서 수행되는 프로세스에 의해 사용되는 포트 번호
길이	2	헤더와 데이터를 합한 데이터그램의 총 길이
검사합	2	헤더와 데이터를 모두 포함한 데이터그램 전체에 대해 오류를 탐지

UDP

- 데이터그램
  - 예제 14.1

16진수의 형식으로 UDP 헤더를 덤프한 것이다. 다음 질문에 답하라.

CB84000D001C001C

- 문제&풀이
  a. 발신지 포트 번호는 얼마인가?
     - 상위 비트에서 16비트이므로 $\mathrm{CB84}_{16}$이며 10진수로 52100임
  b. 목적지 포트 번호는 얼마인가?
     - 발신지 포트 번호 다음 16비트이므로 $\mathrm{000D}_{16}$이며 10진수로 13임
  c. 데이터그램의 총 길이는 얼마인가?
     - 세 번째 네 개의 16진수 $\mathrm{001C}_{16}$이며 10진수로 28바이트임
  d. 데이터 길이는 얼마인가?
     - 데이터 길이는 전체길이에서 헤더의 길이를 뺀 값이므로 28 – 8 = 20 바이트임
  e. 데이터의 전송 방향이 클라이언트로부터 서버 쪽인가?
     - 목적지 포트 번호가 잘 알려진 포트 번호 13이므로 클라이언트로부터 서버로 패킷이 전송됨

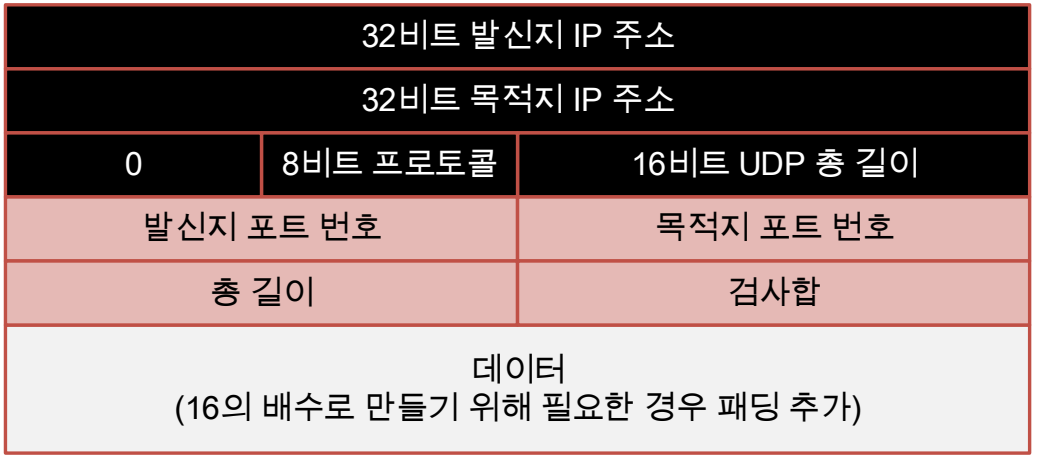
UDP

- 프로세스 간 통신
  - IP 주소와 포트 번호를 사용하여 목적지 프로세스에 데이터그램을 전송함

포트	프로토콜	설명
7	Echo	수신한 데이터그램을 송신자에게 다시 보냄
9	Discard	수신된 데이터그램을 모두 폐기함
11	Users	활동 중인 사용자
13	Daytime	날짜와 시간을 반환
17	Quote	오늘의 인용문을 반환
19	Chargen	일련의 문자를 반환
53	Domain	도메인 이름을 IP 주소로 변환
67	Bootps	부트스트랩 정보를 다운로드할 서버 포트
68	Bootpc	부트스트랩 정보를 다운로드할 클라이언트 포트
69	TFTP	간단한 파일 전송 프로토콜
111	RPC	원격 프로시저 호출
161	SNMP	네트워크 장비를 모니터링하고 관리
162	SNMP	SNMP 트랩 메시지를 수신하는 데 사용

UDP

- 오류 검출
  - 검사합을 제외한 오류 제어 메커니즘이 없음
    - 의사 헤더, UDP 헤더, 응용 계층으로부터 온 페이로드를 검사함
    - 송신자는 검사합을 포함하지 않고 데이터그램을 전송할 수 있음
      - 이 경우, 필드를 0으로 채우고 UDP 패킷을 전송
    - 검사합 결과가 모두 0인 경우, 검사합을 모두 1로 변경하고 UDP 패킷 전송



UDP

- 오류 검출
  - 예제 14.2

다음 가상 상황에서 체크섬으로 어떤 값이 전송되는가?

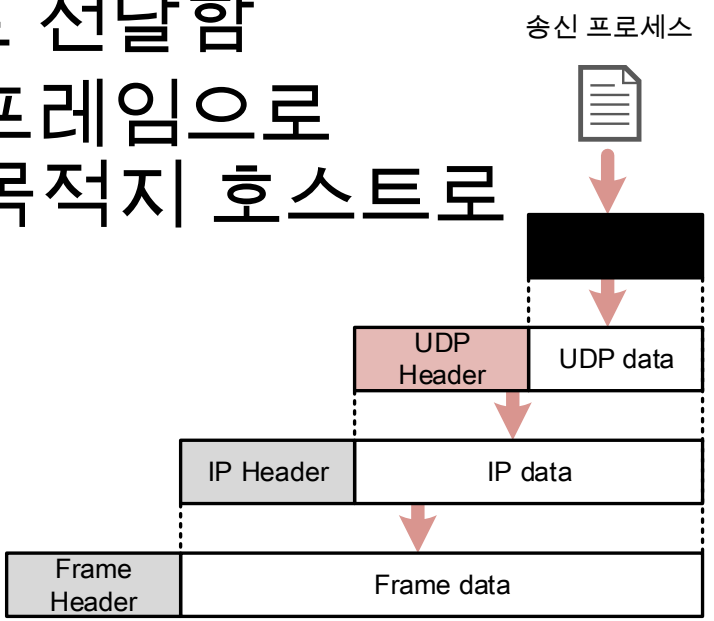
- 문제 & 풀이
  a. 송신자는 검사합을 포함하지 않는 경우
     - 검사합이 계산되지 않은 것을 알리기 위해 검사합 필드를 모두 0으로 채움

  b. 송신자는 검사합을 포함하기로 결정했으나, 검사 값이 모두 0인 경우
     - 송신자가 합에 보수를 취한 결과가 모두 0이라면, 송신자는 전송하기 전에 결과 값에 다시 보수를 취함
     - 즉, 검사합으로 전송되는 값은 모두 1임(a 경우와 혼돈을 막기 위해)

UDP

- 캡슐화

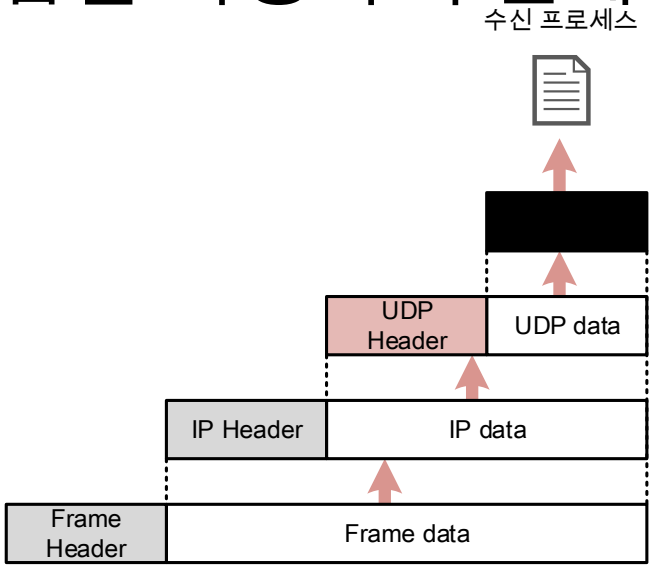
1. 프로세스가 UDP를 통하여 보낼 메시지가 있는 경우, 소켓 주소와 데이터 길이를 전송 계층에 전달함
2. UDP는 데이터에 UDP헤더를 추가하여 데이터그램을 네트워크 계층으로 전달함
3. IP는 프로토콜 필드 값을 보고 UDP임을 인지하고 IP 헤더를 추가해서 하위 계층으로 전달함
4. IP 패킷은 데이터 링크 계층에서 프레임으로 캡슐화되어 물리 계층을 거친 뒤 목적지 호스트로 전송함



UDP

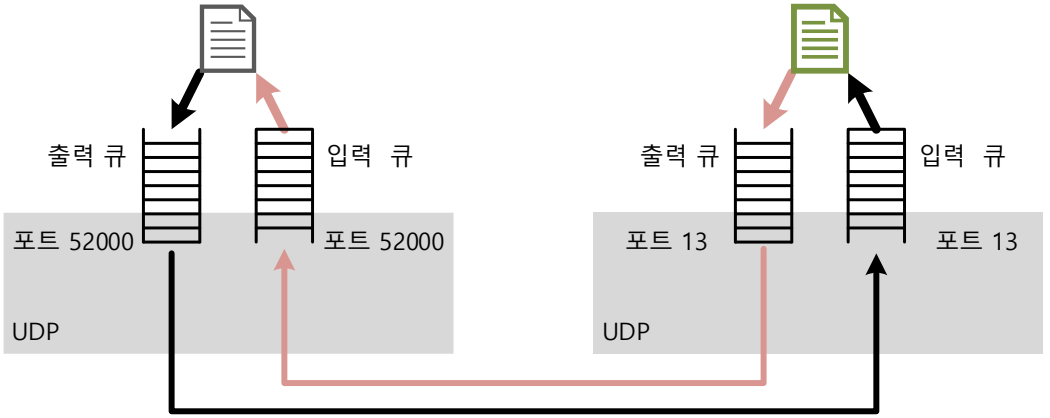
- 역캡슐화

1. 물리 계층은 신호를 비트로 전환하고 데이터 링크 계층으로 전달
2. 데이터 링크 계층은 프레임을 점검하고 오류가 없으면 헤더를 제거한 프레임을 IP에 전달함
3. IP는 헤더 검사 후, 헤더를 없앤 패킷을 UDP로 보냄
4. UDP는 검사합을 사용하여 전체 데이터그램을 검사함



UDP

- 큐잉(Queuing)
  - 정의
    - 데이터나 작업을 처리하기 위해 대기열에 넣어 순차적으로 처리하는 방식
  - 특징
    - 큐에서 오버플로우가 일어날 수 있음
      - 송신 측은 운영체제가 송신 측 프로세스에게 더 이상 메시지를 전달하지 않을 것을 요청
    - 각 프로세스의 포트번호에 의거하며 입력 큐와 출력 큐를 가짐

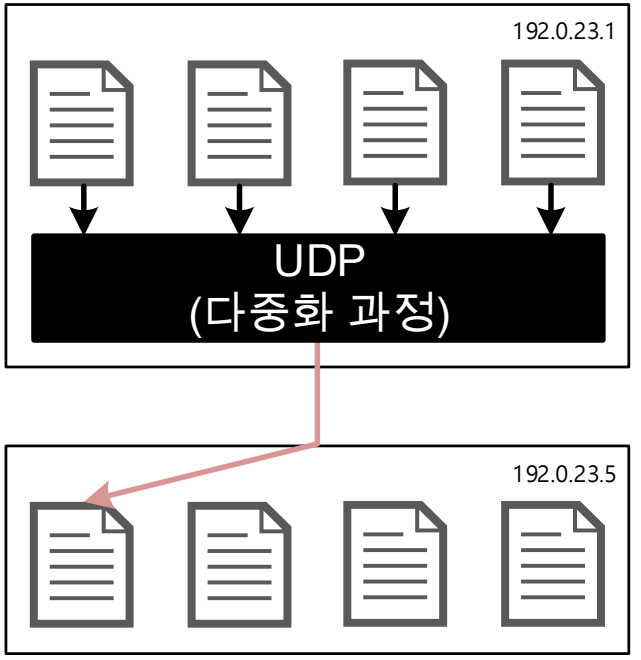


UDP

- 큐잉(Queuing)
  - 송신 UDP
    - 송신 측에서 프로세스가 시작되면 운영체제에게 포트 번호를 요청함
      - 한 프로세스가 여러 프로세스와 연결되기를 원하더라도 하나의 입력 큐와 출력 큐를 관리함
  - 수신 UDP
    - 송신 측에서 프로세스가 시작되면 운영체제에게 포트 번호를 요청함
    - 수신 측에 메시지가 도착하였을 때, 목적지 포트 번호로 지정된 입력 큐가 있는지 점검함
      - 입력 큐가 생성되었다면 특정 이벤트가 발생했음을 수신 프로세스에 알림
      - 입력 큐가 생성되지 않았다면 UDP는 데이터그램을 폐기하고 ICMP 프로토콜에 'Port unreachable' 메시지를 수신 측에 보낼 것을 요청

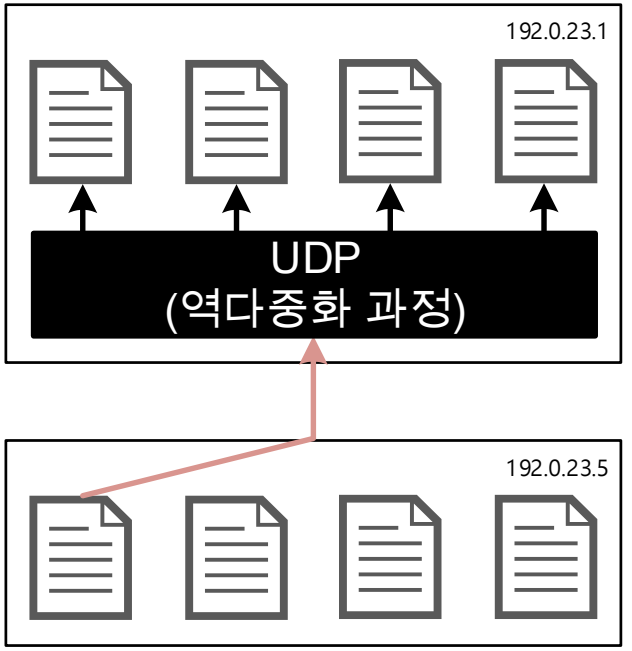
UDP

- 다중화
  - 송신 측에서 데이터를 보내고자 하는 프로세스가 여러 개 있는 경우 사용
    - 동일한 IP 주소 구분
      - 각 프로세스에 할당된 포트 번호를 사용하여 프로세스를 구분하고 메시지를 전송함
    - 동시 전송
      - 여러 프로세스의 데이터를 하나의 서버로 동시에 보낼 수 있음



UDP

- 역다중화
  - 수신 측에서 데이터를 받고자 하는 프로세스가 여러 개 있는 경우 사용
    - 헤더 정보 활용
      - 수신된 데이터그램의 헤더에 포함되어 있는 포트번호를 기반으로 세그먼트를 적절한 프로세스에 전달
    - 독립적인 처리
      - 여러 데이터그램이 동시에 수신되더라도 각 데이터그램을 별개의 포트로 전달됨
    - 빠른 처리
      - 비연결형 프로토콜이기 때문에 확인응답 과정을 거치지 않아도 됨

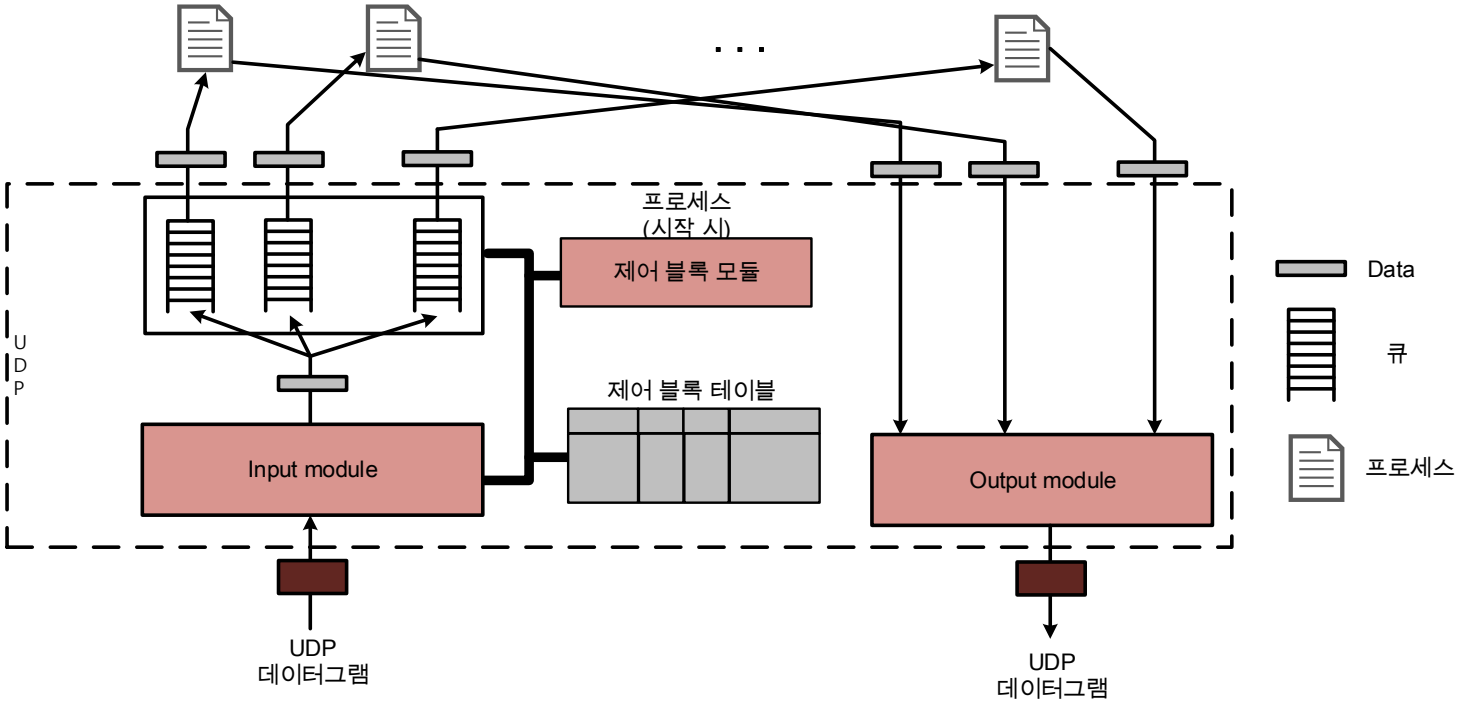


UDP

- 응용
  - UDP에 흐름 제어와 오류 제어를 통한 신뢰 제공을 위해 다른 프로토콜과 함께 사용하는 경우
    - e.g., TFTP(Trivial File Transfer Protocol)
  - 멀티캐스팅을 하는 경우
    - 다량의 데이터를 전송하는 경우, 연결 및 종료 지연이 송신자에게 부담이 됨
  - 짧은 요청과 짧은 응답을 원하는 경우
    - DNS와 같은 클라이언트 서버 응용은 클라이언트가 짧은 요청을 보내고 빠른 응답을 원하기 때문에 UDP 사용
    - SMTP와 같은 클라이언트 서버 응용은 비순차적 도착으로 메시지가 의도대로 전달되지 않을 수 있기에 UDP를 사용하지 않음

UDP

- 패키지



- 구성
  - 제어 블록 테이블
  - 입력 큐
  - 제어 블록 모듈
  - 입력 모듈
  - 출력 모듈

UDP

- 패키지
  - 제어 블록 테이블
    - 각 UDP 연결에 대한 상태 정보를 저장하는 테이블
    - 구성
      - 상태
        - FREE, IN-USE
      - 프로세스 ID
      - 포트 번호
      - 상응하는 큐 번호

상태	프로세스 ID	포트 번호	큐 번호
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
FREE			
IN-USE	4,652	52,012	38
FREE			

UDP

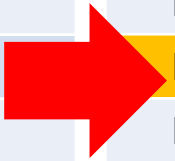
- 패키지
  - 제어 블록 모듈
    - UDP의 동작을 관리하고 제어하는데 필요한 데이터 구조와 기능을 포함하는 시스템의 구성요소
      - 제어 블록 테이블의 관리를 담당
    - 시작된 프로세스로부터 프로세스 ID와 포트 번호를 넘겨 받아 포트와 바인딩하고, 바인딩 정보를 제어 블록 테이블에 저장

```
UDP_Control_Block_Module (processID, portNumber) {
        Search the table for a FREE entry

        if (FREE entry is not found)
                Delete one entry using a predefined policy // 보통, 오랫동안 쓰이지 않은 Entry를 비움
        Create a new entry with the state IN-USE
        Enter the process ID and the port number // 아직, Incomming Queue는 할당하지 않음

        Return
} // End Module
```

UDP

- 패키지
  - 제어 블록 모듈 동작
    - 시작하는 프로세스는 운영체제에 포트 번호를 요청하고 52,014를 할당하려고 함
      - FREE 상태 엔트리에 수신된 정보를 입력함
      - 자신의 프로세스 ID(4,789)와 포트 번호를 제어 블록 모듈에 보내 테이블의 엔트리를 생성
        - 목적지 프로세스에 데이터그램이 도착하지 않았을 경우, 모듈은 큐를 배당하진 않음

상태	프로세스 ID	포트 번호	큐 번호
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
FREE			
IN-USE	4,652	52,012	38
FREE			

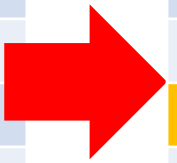


상태	프로세스 ID	포트 번호	큐 번호
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
IN-USE	4,789	52,014	
IN-USE	4,652	52,012	38
FREE			

UDP

- 패키지
  - 제어 블록 모듈 동작
    - 52,011 포트에 데이터그램이 도착하면, 입력 모듈은 테이블을 검사하여 이 목적지를 위한 큐가 아직 배당되지 않았음을 인지했다면 모듈 43번 큐를 생성
    - 등록 되지 않은 포트 번호의 데이터그램이 도착했다면 해당 데이터그램은 폐기하고 'Port unreachable' ICMP 메시지를 전송

상태	프로세스 ID	포트 번호	큐 번호
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
IN-USE	4,789	52,014	
IN-USE	4,652	52,012	38
FREE			



상태	프로세스 ID	포트 번호	큐 번호
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	43
IN-USE	4,789	52,014	
IN-USE	4,652	52,012	38
FREE			

UDP

- 패키지
  - 입력 모듈
    - 수신된 UDP 데이터그램을 처리하는 기능
    - 각 포트에는 저마다 입력 큐가 존재하여 입력 모듈은 도착한 데이터그램들의 목적지 포트 번호를 확인하여 큐를 삽입함

```
UDP_Input_Module (user_datagram) {
   Look for the entry in the Control_Block_Table  // 입력된 데이터그램의 목적지 포트 번호에 일치하는 엔트리가 있는지 검색
   if(Target entry is found) {
      Check to see if a Queue is allocated
      if(Queue is not allocated)  // 아직 Incomming Queue가 할당되지 않았다면, 할당
         Allocate a Queue        // 즉, 현재 User_datagram이 첫 입력값임을 의미
      Enqueue the data           // 해당 Queue로 데이터를 삽입
   }
   else {  // 목적지 포트 번호에 해당되는 프로세스가 없는 경우
      Ask ICMP to send an "Unreachable port" Message
      Discard the user datagram
   }
   Return
}  // End Modul
```

UDP

- 패키지
  - 출력 모듈
    - 생성된 UDP 데이터그램을 출력값으로 내보냄
    - 프로세스에서 생성된 데이터그램을 출력하는 단순한 구조로 구현됨

```
UDP_Output_Module (Data) {
   Create a user datagram    // 데이터를 user_datagram으로 Encapsulation하여
   Send the user datagram    // 내보냄
   Return
}
```

Thanks!

김 혜 정(hyejeong@pel.sejong.ac.kr)