2025/09/26, 2025 네트워크 세미나

# TCP/IP 프로토콜

## -13장 전송 계층 개요, 14장 UDP-

송 민 희(minhee@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

# 목 차

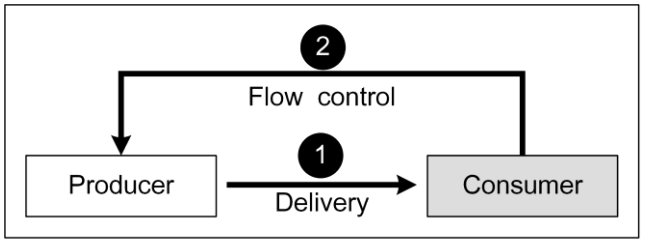
- 보충
- 전송 계층 개요
- UDP (User Datagram Protocol)

# 목 차

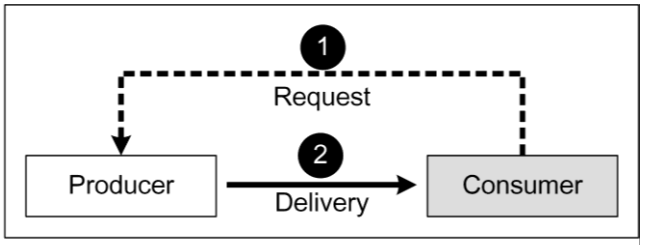

- 보충
- 전송 계층 개요
- UDP (User Datagram Protocol)

# 보충

- 전송 계층 서비스
  - 흐름 제어(Flow Control)
    - 밀기와 끌기
      - 밀기
        - 송신측에서 데이터가 준비되는 대로 즉시 전송
        - 수신측이 준비되지 않은 경우 흐름제어 필요
      - 끌기
        - 수신측이 필요 할 때 데이터 요청
        - 송신측은 요청이 있을 때만 데이터를 전송하여 흐름제어 불필요



a. Pushing



b. Pulling

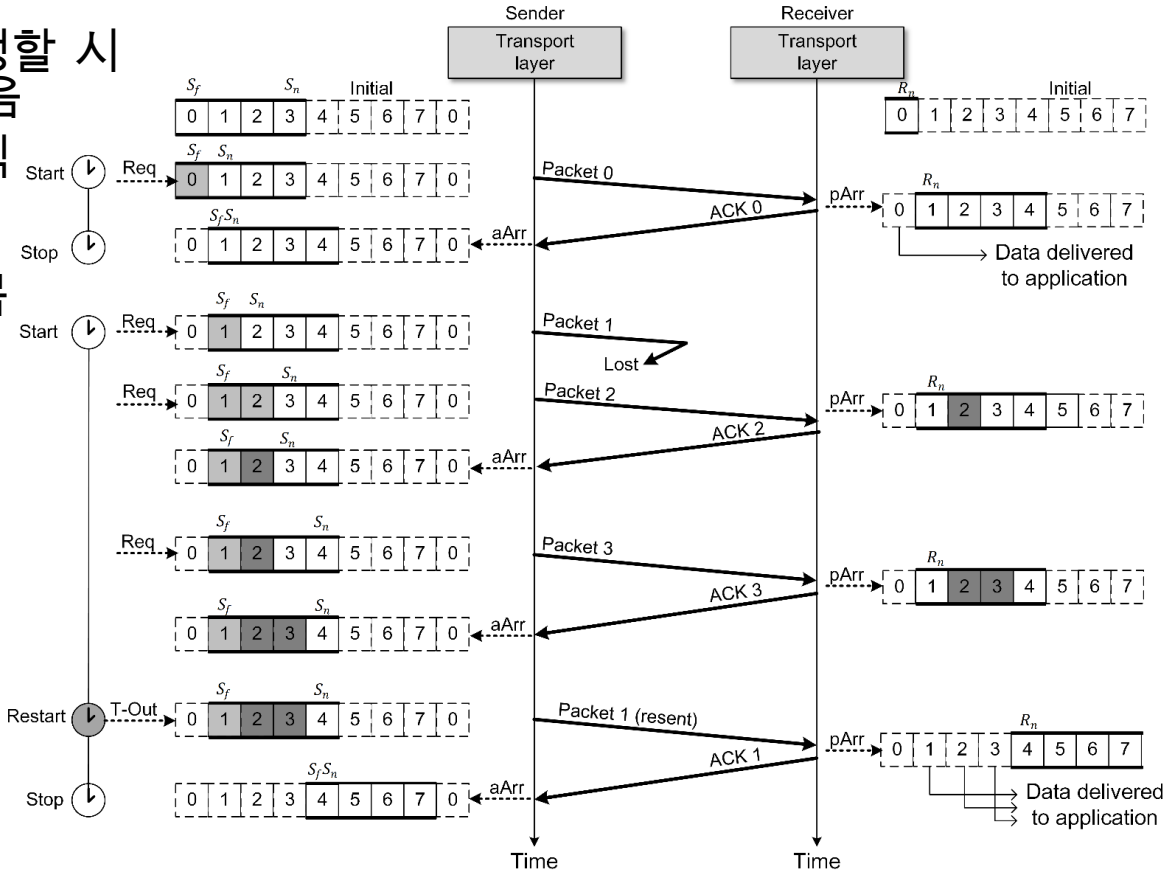
    - 버퍼
      - 송신측과 수신측에서 패킷을 저장할 수 있는 메모리 영역
      - 수신측의 버퍼가 가득 차면, 수신측에서 패킷의 전송을 멈추도록 송신측에 패킷의 윈도우 크기 필드에 작성하여 알림
      - 수신측의 버퍼에 빈 공간이 생기면 수신측에서 패킷을 다시 전송하도록 송신측에 패킷의 윈도우 크기 필드에 작성하여 알림

# 보충

- 전송 계층 서비스

*로드
네트워크로 전송되는 패킷의 수

  - 혼잡 제어(Congestion Control) – 종류
    - 개방 루프 혼잡 제어
      - 재전송 정책
        - 불필요한 재전송이 발생할 시 혼잡을 가중시킬 수 있음
        - 재전송으로 인한 트래픽 증가를 줄이기 위해 타이머와 확인 응답을 관리하여 불필요한 중복 재전송을 억제함



# 보충

- UDP 패키지 – 구성요소
  - 출력 모듈
    - 개요
      - 사용자 데이터그램을 생성하고 보냄
    - 의사코드

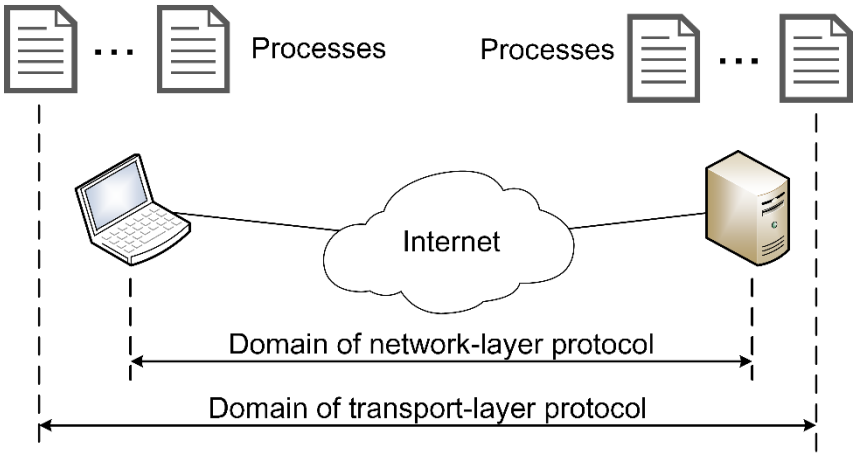
```
UDP_Input_Module (User Datagram)
{
      Create a user datagram
      Send the user datagram
      Return
}    // End module
```

# 목 차


- 보충
- 전송 계층 개요
- UDP (User Datagram Protocol)

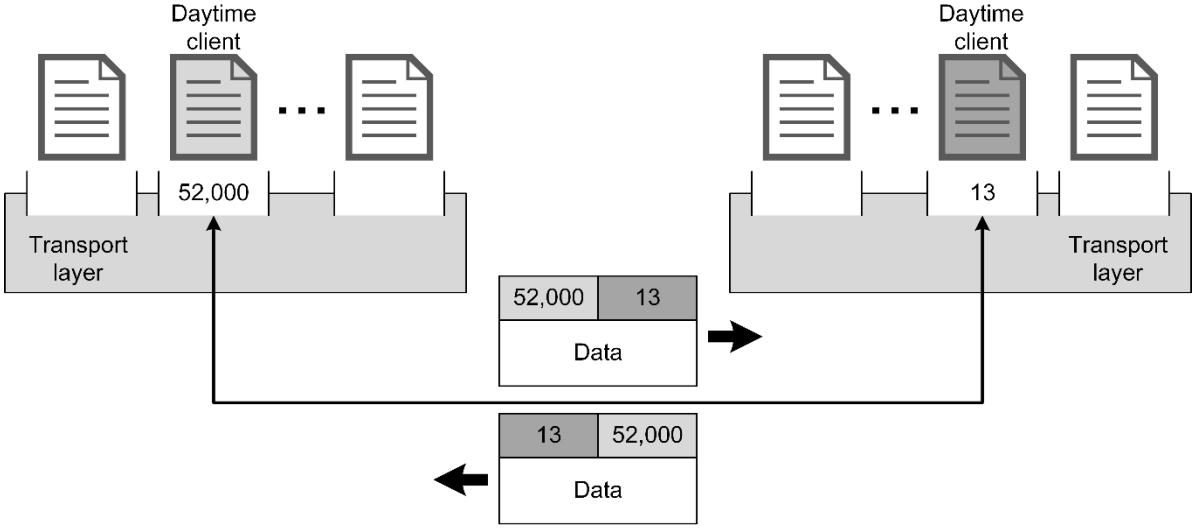
# 전송 계층 개요

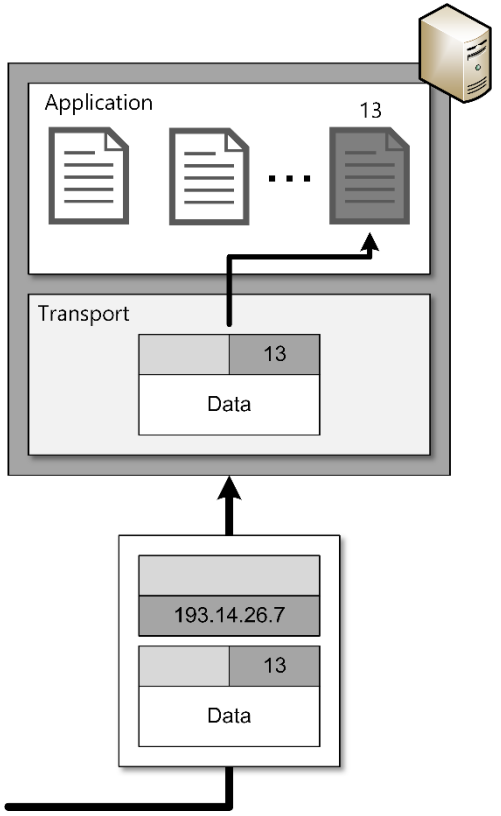
- 정의
  - 네트워크 계층이 제공하는 호스트 간 통신을 기반으로, 프로세스 간 신뢰성 있는 데이터 전송을 담당하는 계층
- 역할
  - 네트워크 계층에서 전달한 메시지를 올바른 프로세스에서 처리하도록 전달
    - 프로세스 대 프로세스 통신 제공
    - 포트 번호를 이용하여 프로세스 식별



# 전송 계층 개요

- 전송 계층 서비스 (1/21)
  - 포트번호 (1/3)
    - 개요
      - 프로세스를 식별하기 위한 주소 체계
      - 호스트 내부에서 프로세스 식별을 위해 사용
        - IP 주소만으로는 호스트까지만 식별 가능
      - ICANN (Internet Corporation for Assigned Names and Numbers)에서 범위 관리



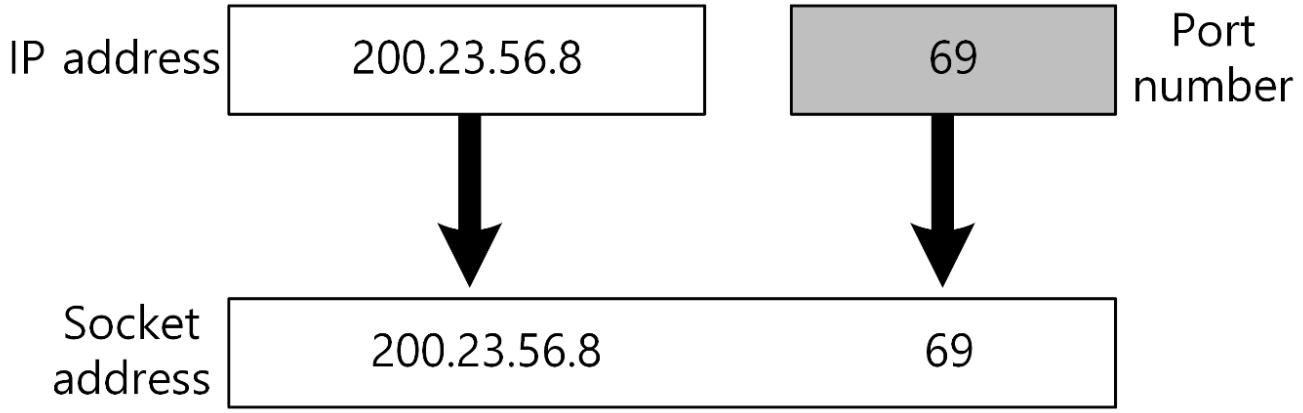
# 전송 계층 개요

- 전송 계층 서비스 (2/21)
  - 포트번호 (2/3)
    - 범위
      - 잘 알려진 포트 (Well-known Port)
        - 0~1,023내 에서 ICANN에 의해 배정되고 제어되는 범위
      - 등록된 포트(Registered Port)
        - 1,024~49,151내 에서 중복을 피하기 위해 ICANN에 등록할 수 있는 범위
      - 동적 포트(Dynamic Port)
        - 49,152~65,535내 에서 임시 또는 개인 포트 번호로 사용할 수 있는 범위



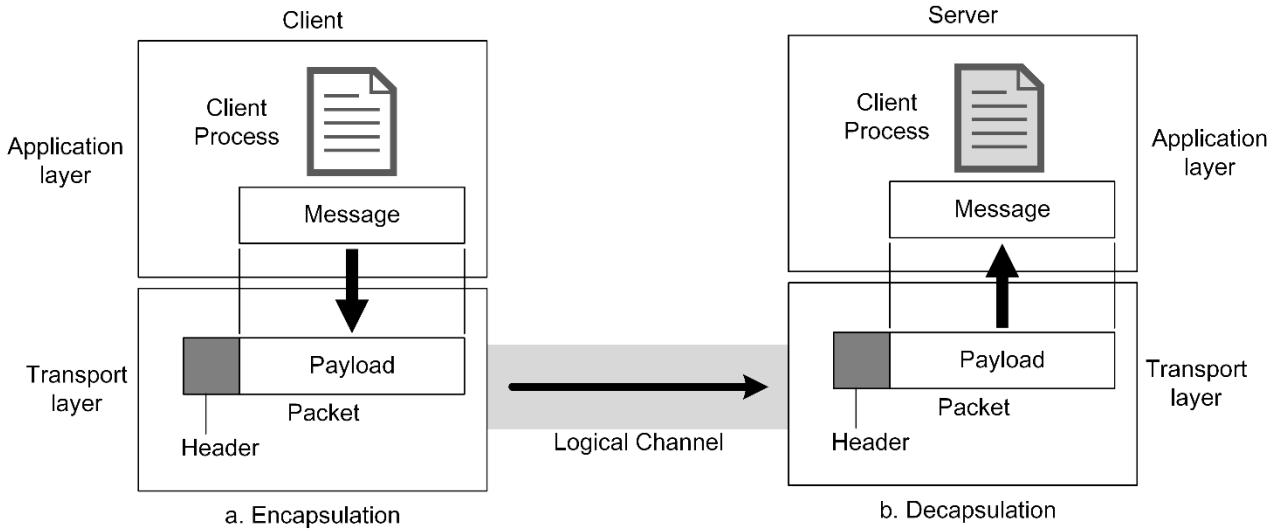
# 전송 계층 개요

- 전송 계층 서비스 (3/21)
  - 포트번호 (3/3)
    - 소켓 주소(Socket Address)
      - IP 주소와 포트 번호의 조합
      - 클라이언트와 서버 프로세스를 각각 유일하게 정의함
      - IP 주소는 네트워크 계층 패킷 헤더에, 포트 번호는 전송 계층 패킷 헤더에 저장되어 전달됨



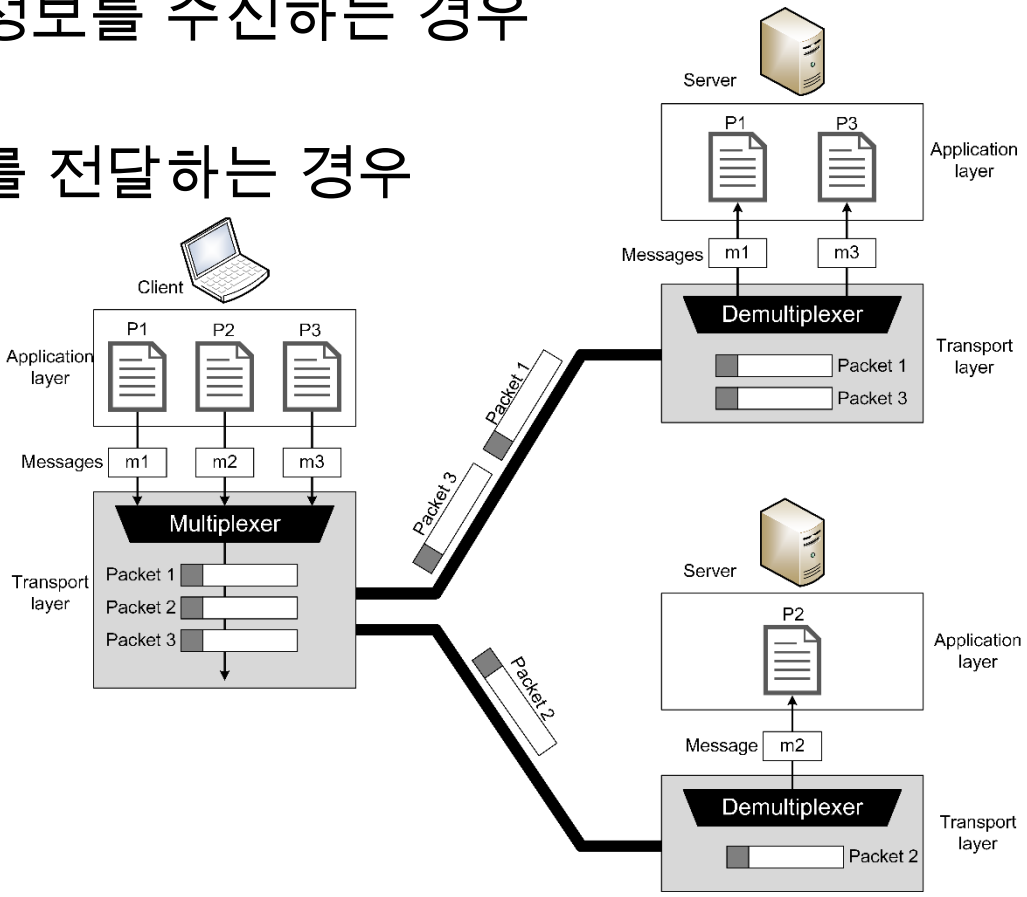
# 전송 계층 개요

- 전송 계층 서비스 (4/21)
  - 캡슐화(Encapsulation)와 역캡슐화(Decapsulation)
    - 캡슐화
      - 전송할 메시지가 있는 송신측 프로세스에서 필요한 정보를 전송 계층 헤더로 붙이는 과정
    - 역캡슐화
      - 메시지를 수신한 전송 계층이 헤더를 없애고 메시지를 응용 계층에서 구동되는 프로세스로 패킷을 전달하는 과정



# 전송 계층 개요

- 전송 계층 서비스 (5/21)
  - 다중화(Multiplexing)와 역다중화(Demultiplexing)
    - 다중화
      - 엔티티가 여러 소스로부터 정보를 수신하는 경우
    - 역다중화
      - 엔티티가 여러 소스로 정보를 전달하는 경우

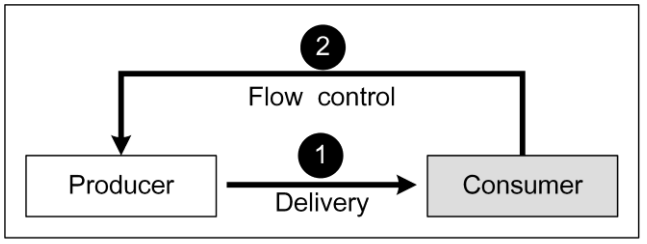


# 전송 계층 개요

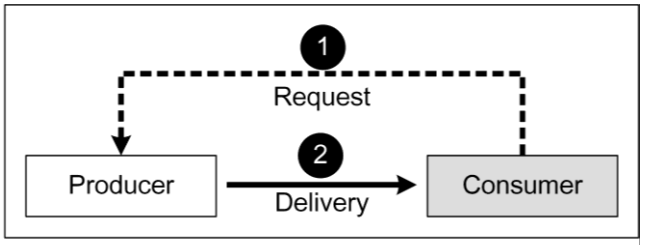
- 전송 계층 서비스 (6/21)
  - 흐름 제어(Flow Control) (1/3)
    - 개요
      - 송신측이 수신측의 처리 속도를 고려하여 전송 속도를 조절하는 기능
      - 데이터가 과도하게 수신되어 정보의 일부가 손실되는 것을 방지하기 위한 수단
    - 방식
      - 밀기(Pushing)와 끌기(Pulling)
      - 버퍼(Buffer) 사용

# 전송 계층 개요

- 전송 계층 서비스 (7/21)
  - 흐름 제어(Flow Control) (2/3)
    - 밀기와 끌기
      - 밀기
        - 송신측에서 데이터가 준비되는 대로 즉시 전송
        - 수신측이 준비되지 않은 경우 흐름제어 필요
      - 끌기
        - 수신측이 필요 할 때 데이터 요청
        - 송신측은 요청이 있을 때만 데이터를 전송하여 흐름제어 불필요



a. Pushing

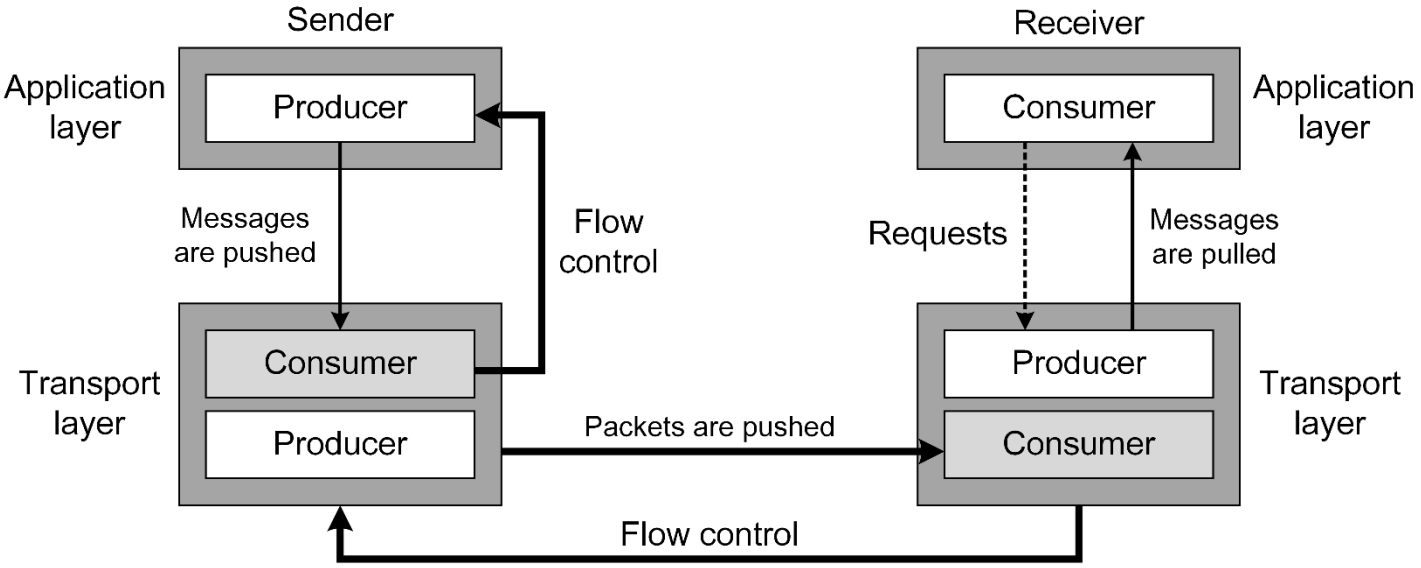


b. Pulling

    - 버퍼
      - 송신측과 수신측에서 패킷을 저장할 수 있는 메모리 영역
      - 수신측의 버퍼가 가득 차면, 수신측에서 패킷의 전송을 멈추도록 송신측에 패킷의 윈도우 크기 필드에 작성하여 알림
      - 수신측의 버퍼에 빈 공간이 생기면 수신측에서 패킷을 다시 전송하도록 송신측에 패킷의 윈도우 크기 필드에 작성하여 알림

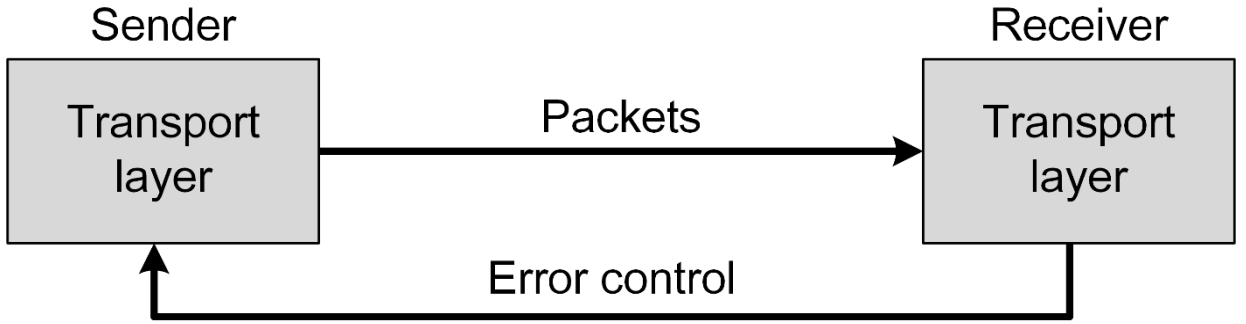
# 전송 계층 개요

- 전송 계층 서비스 (8/21)
  - 흐름 제어(Flow Control) (3/3)
    - 전송 계층에서의 흐름 제어
      - 송신 전송 계층은 메시지를 패킷으로 캡슐화한 후 밀기 방식을 이용하여 수신 전송 계층으로 전달함
      - 수신 전송 계층은 수신한 메시지를 역캡슐화하고 끌기 방식을 이용하여 응용 계층으로 전달함
        - 전송 계층은 응용 계층 프로세스가 메시지를 요청하기 전까지 대기함



# 전송 계층 개요

- 전송 계층 서비스 (9/21)
  - 오류 제어(Error Control) (1/3)
    - 개요
      - 전송 과정에서 발생할 수 있는 오류를 검출하고 수정
      - 전송 계층에 신뢰성을 제공함
    - 기능
      - 훼손된 패킷의 감지 및 폐기
      - 손실되거나 제거된 패킷을 추적하고 재전송
      - 중복 수신 패킷을 확인하고 폐기
      - 손실된 패킷이 도착할 때까지 순서에 어긋나게 들어온 패킷을 버퍼에 저장



# 전송 계층 개요

- 전송 계층 서비스 (10/21)
  - 오류 제어(Error Control) (2/3)
    - 순서 번호
      - 개요
        - 각 전송 패킷에 부여되는 번호
        - 어떤 패킷을 재전송해야 하는지, 중복, 손실 등이 발생했는지를 확인하기 위해 사용
      - 특징
        - 훼손되거나 손실된 패킷에 대해 재전송 여부 확인 가능
        - 순서번호의 동일 여부를 통해 패킷의 중복 수신 확인 가능
        - 순서 번호의 간격을 통해 패킷의 순서 확인 가능
        - 순서 번호를 위해 $m$비트가 설정된 경우 0부터 $2^m - 1$의 범위를 가지며 순서 번호의 값은 $mod\ 2^m$

# 전송 계층 개요

- 전송 계층 서비스 (11/21)
  - 오류 제어(Error Control) (3/3)
    - 확인 응답
      - 개요
        - 패킷의 전달에 따라 전송되는 긍정과 부정 신호
      - 특징
        - 정상 수신된 패킷에 대해서는 ACK 응답을 보냄
        - 손실되거나 손상된 패킷에 대해서는 NAK 응답을 보냄
        - 송신측은 타이머가 만료되기 전까지 수신측으로 부터 ACK가 오지 않으면 패킷을 재전송함

# 전송 계층 개요

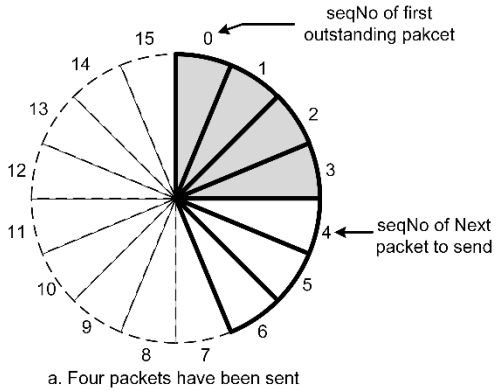
- 전송 계층 서비스 (12/21)
  - 흐름과 오류 제어의 결합 (1/3)
    - 개요
      - 송신측과 수신측에 각각 번호를 매긴 버퍼를 통해 흐름제어와 오류 제어를 결합할 수 있음
    - 송신측 동작 과정
      1. 패킷의 순서 번호로 버퍼 내의 비어있는 영역의 제일 앞 부분 번호인 $x$를 사용함
      2. 패킷이 전송 되면 패킷의 복사본이 메모리 영역 $x$에 저장됨
      3. 상대방의 확인 응답을 기다림
      4. 확인 응답이 도착하면 패킷은 제거되고 메모리 영역은 비워짐
    - 수신측 동작 과정
      1. 순서 번호 $y$를 갖는 패킷이 도착하면 응용 계층에서 수신할 준비가 될 때까지 패킷을 메모리 영역 $y$에 저장함
      2. 패킷 $y$의 도착을 알리기 위해 확인 응답을 전송함

# 전송 계층 개요

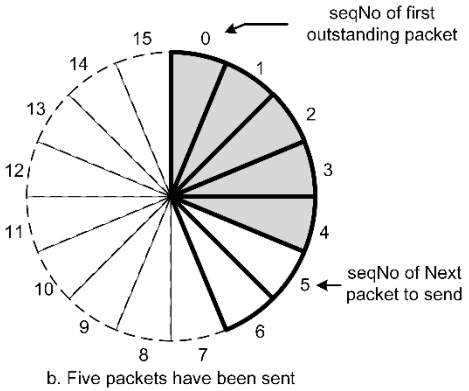
- 전송 계층 서비스 (13/21)
  - 흐름과 오류 제어의 결합 (2/3)
    - 슬라이딩 윈도우(Sliding Window) (1/2)
      - 개요
        - 순서 번호를 원형으로 표현할 때 버퍼를 표현하는 일련의 조각 집합
      - 동작 방식
        - 송신측에서 패킷이 전송되면 관련 조각이 메시지를 응용계층으로 수신할 수 없음을 표현하기 위해 마크 됨
        - 확인응답이 수신되면 해당 조각의 마크가 해제 됨
        - 윈도우의 시작으로부터 연속적인 조각의 마크가 해제되며 윈도우의 끝에서 더 많은 빈 조각을 허용하기 위해 윈도우를 돌림

# 전송 계층 개요

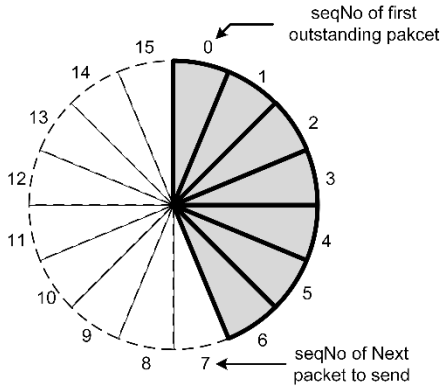
- 전송 계층 서비스 (14/21)
  - 흐름과 오류 제어의 결합 (3/3)
    - 슬라이딩 윈도우(Sliding Window) (2/2)
      - 순서 번호가 $mod\ 16\ (m = 4)$이며 윈도우의 크기가 7인 경우



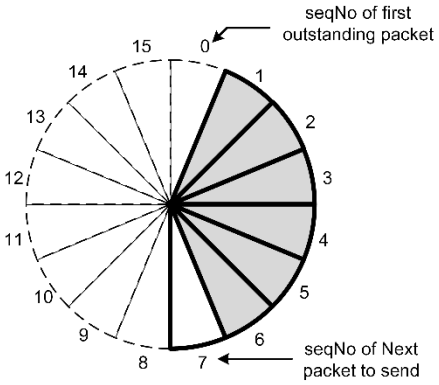
a. Four packets have been sent



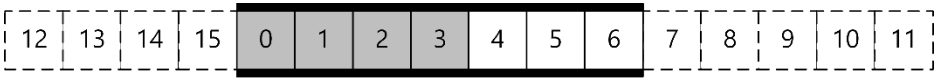
b. Five packets have been sent



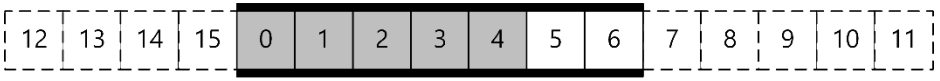
c. Seven packets have been sent window is full



d. Packet 0 has been acknowledged, window slides



a. Four packets have been sent



b. Five packets have been sent



c. Seven packets have been sent window is full



d. Packet 0 has been acknowledged and window slid

# 전송 계층 개요

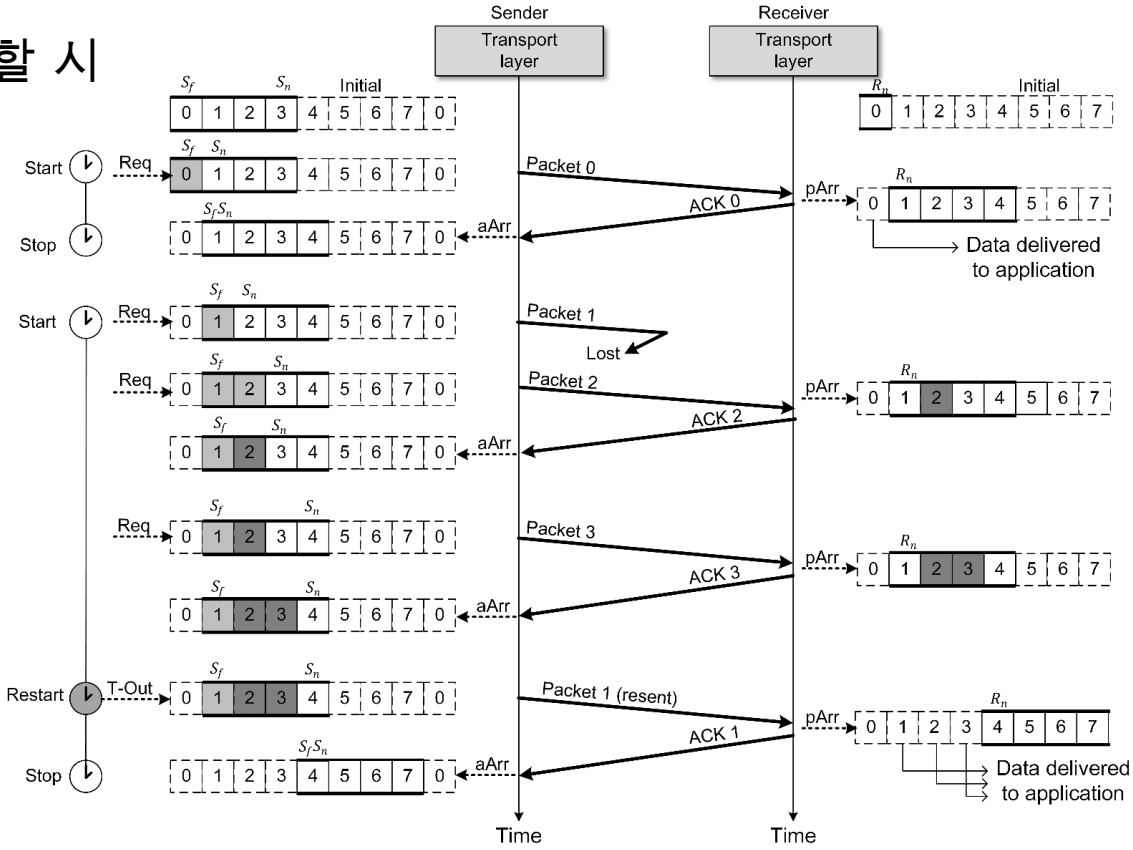
- 전송 계층 서비스 (15/21)
  - 혼잡 제어(Congestion Control)
    - 혼잡
      - 네트워크에서 처리할 수 있는 패킷의 수를 초과하는 경우 혼잡 발생
      - 라우터나 스위치가 패킷을 저장하기 위한 큐에서 발생
        - 패킷이 큐에 저장되고 전송 순서를 기다리는 과정에서 큐의 용량이 유한하고, 용량을 초과해서 패킷을 저장할 수 없음
    - 종류 (1/3)
      - 개방 루프 혼잡 제어(Open-Loop Congestion Control)
        - 혼잡을 미연에 방지하기 위한 정책을 적용하는 방식
        - 소스나 목적지에서 수행됨
      - 폐 루프 혼잡 제어(Closed-Loop Congestion Control)
        - 혼잡이 발생한 후에 혼잡을 완화시키기 위한 방식

# 전송 계층 개요

- 전송 계층 서비스 (16/21)

*로드
네트워크로 전송되는 패킷의 수

  - 혼잡 제어(Congestion Control) – 종류 (2/3)
    - 개방 루프 혼잡 제어 (1/2)
      - 재전송 정책
        - 불필요한 재전송이 발생할 시 혼잡을 가중시킬 수 있음
        - 재전송으로 인한 트래픽 증가를 줄이기 위해 타이머와 확인 응답을 관리하여 불필요한 중복 재전송을 억제함



# 전송 계층 개요

- 전송 계층 서비스 (17/21)

*로드
네트워크로 전송되는 패킷의 수

  - 혼잡 제어(Congestion Control) – 종류 (3/3)
    - 개방 루프 혼잡 제어 (2/2)
      - 윈도우 정책
        - 송신 윈도우 크기를 지나치게 크게 설정할 시 네트워크 혼잡이 발생함
        - 윈도우 크기를 조절하여 네트워크 혼잡을 줄임
      - 확인 응답 정책
        - 수신측에서 수신한 패킷에 대해 확인 응답을 전송하지 않는다면 송신측에서의 전송속도가 느려져 혼잡이 발생함
        - 수신측에서 전송할 패킷이 있거나 특정한 타이머가 만료되는 경우에만 확인응답을 전송하게 하여 네트워크의 *로드 를 줄임
    - 폐 루프 혼잡 제어
      - 송/수신측 네트워크 장비가 혼잡 상태를 탐지하고 윈도우 크기나 전송률을 조절하도록 함

# 전송 계층 개요

- 전송 계층 서비스 (18/21)
  - 비연결형과 연결 지향 서비스 (1/4)
    - 비연결형(Connectionless) 서비스
      - 동일한 메시지에 속하는 서로 다른 데이터그램이 서로 다른 경로를 거칠 수 있음
      - 데이터들의 관계를 고려하지 않고 각각의 데이터 조각을 독립적인 하나의 단위로 간주함
      - 비연결형 전송 계층을 통해 전달되며 각 조각들이 순서에 어긋나게 목적지에 도착할 수 있음
      - 첫번째 패킷이 언제 도착할지, 모든 패킷이 다 도착했는지 알 수 없음
      - 흐름 제어나 오류 제어, 혼잡 제어가 구현될 필요가 없음

# 전송 계층 개요

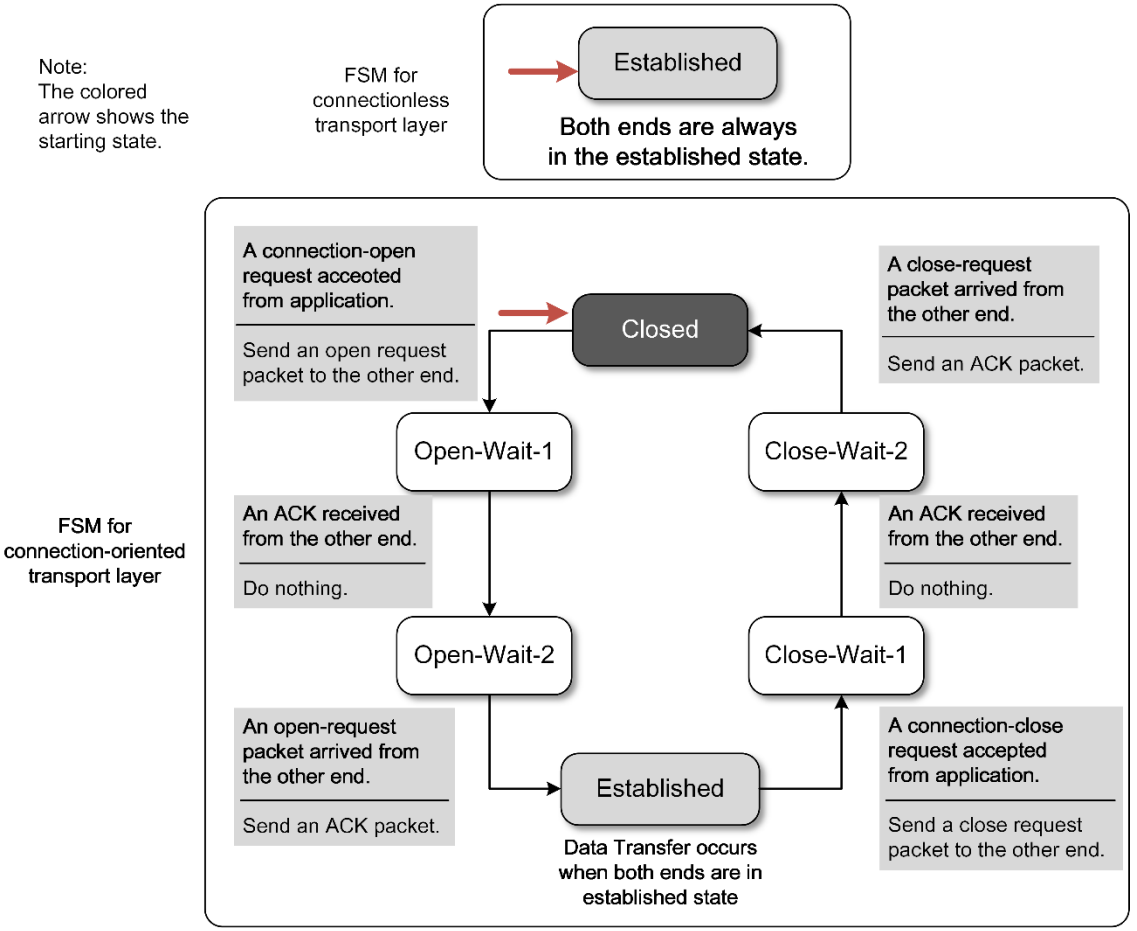
- 전송 계층 서비스 (19/21)
  - 비연결형과 연결 지향 서비스 (2/4)
    - 연결 지향(Connection-Oriented)서비스
      - 클라이언트와 서버가 연결 설정을 우선적으로 수행함
        - 연결이 설정된 이후 데이터 교환 가능
      - 두 개의 종단 호스트만을 연결 지향 서비스에 포함하며, 서비스는 종단에서만 이루어짐
      - 흐름 제어, 오류 제어, 혼잡 제어 등이 구현될 수 있음

# 전송 계층 개요

- 전송 계층 서비스 (20/21)
  - 비연결형과 연결 지향 서비스 (3/4)
    - 유한 상태 기기(FSM, Finite State Machine)
      - 정의
        - 유한한 개수의 상태를 가질 수 있는 수학적 모델
      - 구성
        - 상태: 현재 어떤 상황에 있는지를 나타냄
        - 이벤트: 수행해야 할 액션 리스트
        - 액션: 이벤트가 발생했을 때 수행하는 동작

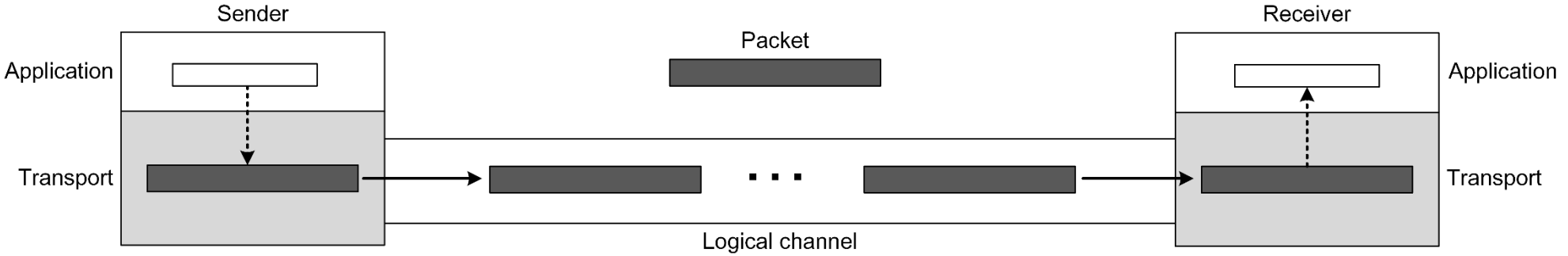
# 전송 계층 개요

- 전송 계층 서비스 (21/21)
  - 비연결형과 연결 지향 서비스 (4/4)
    - FSM로 나타낸 비연결형과 연결 지향 서비스



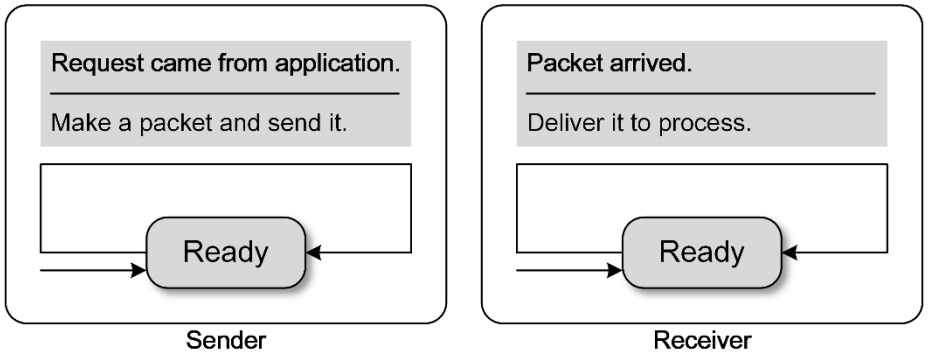
# 전송 계층 개요

- 전송 계층 프로토콜
  - 단순(Simple) 프로토콜
    - 개요
      - 에러 제어, 흐름 제어가 없는 비연결형 프로토콜
      - 빠르게 수신된 데이터를 처리할 수 있음
    - 동작
      1. 전송 계층이 응용 계층으로부터 메시지를 받음
      2. 메시지를 패킷으로 만든 후 해당 패킷을 전송함
      3. 수신측 전송 계층은 네트워크 계층을 통해 들어온 패킷을 수신함
      4. 수신측 전송 계층은 해당 패킷으로부터 메시지를 추출하고 자신의 응용 계층으로 전달함

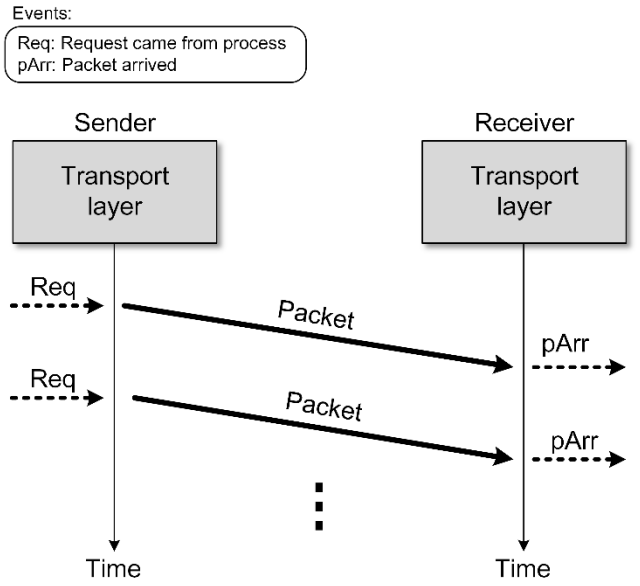


# 전송 계층 개요

- 전송 계층 프로토콜 (1/29)
  - 단순 프로토콜
    - FSM




  - 단순 프로토콜을 이용한 통신 예시
    - 송신측은 수신측을 고려하지 않고 하나씩 패킷을 전송함

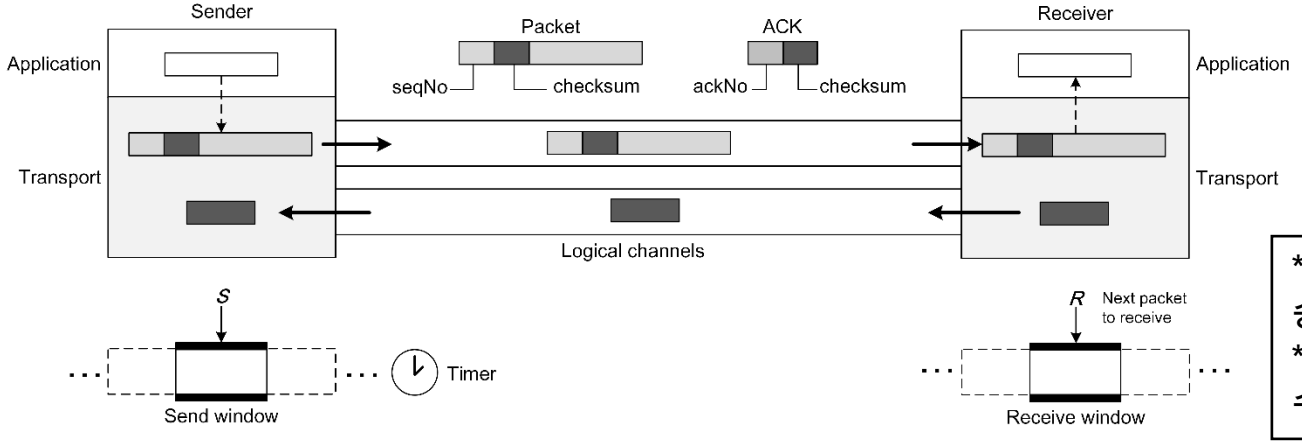


# 전송 계층 개요

- 전송 계층 프로토콜 (2/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (1/7)
    - 개요
      - 매 전송 데이터에 대해 수신자의 확인 응답(ACK)을 받아야 다음 데이터를 전송하는 연결 지향 프로토콜
      - 송신측과 수신측 모두 크기가 1인 슬라이딩 윈도우를 사용함
      - 패킷의 훼손 여부를 확인하기 위해 체크섬(Checksum)을 사용함
        - 수신측은 패킷이 훼손되었으면 추가적인 신호 없이 폐기함
      - 송신측은 패킷을 전송할 때마다 타이머를 구동함
        - 타이머가 만료되면 송신측은 해당 패킷이 손실되었거나 훼손되었다고 판단하여 재전송함
      - 수신 여부를 ACK로 확인할 수 있음
      - 손상/손실된 프레임에 대해 타임아웃 후 재전송을 통해 오류 제어
      - ACK 응답을 통해 송신자는 수신자의 처리속도를 기다려 흐름 제어

# 전송 계층 개요

- 전송 계층 프로토콜 (3/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (2/7)
    - 동작
      1. 송신자는 seqNo와 체크섬을 포함한 패킷을 보내고 타이머를 가동함
         - 타이머가 만료되면 패킷을 재전송 함
      2. 수신자는 패킷을 수신하고 체크섬을 확인함
         - 체크섬 값이 잘못되었으면 패킷을 폐기함
      3. 수신자는 패킷에 오류가 없으면 패킷을 응용 계층으로 전달함
      4. 수신자는 ackNo를 포함한 ACK 패킷을 송신자에게 전달함
      5. 송신자는 ACK 패킷을 수신하여 ackNo를 확인함




*seqNo
송신측에서 보내는 패킷에 입력되는 번호
*ackNo
수신측에서 보내는 패킷에 입력되는 번호

# 전송 계층 개요

- 전송 계층 프로토콜 (4/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (3/7)
    - 순서 번호
      - 패킷을 중복 수신하지 않도록 하기 위하여 순서 번호(seqNo, Sequnce Number)와 확인응답 번호(ackNo, Acknowledgement Number)를 사용함
      - 패킷의 헤더에 순서번호 필드가 추가됨
      - $modulo\ 2$ 연산을 통해 1비트의 필드만 사용함

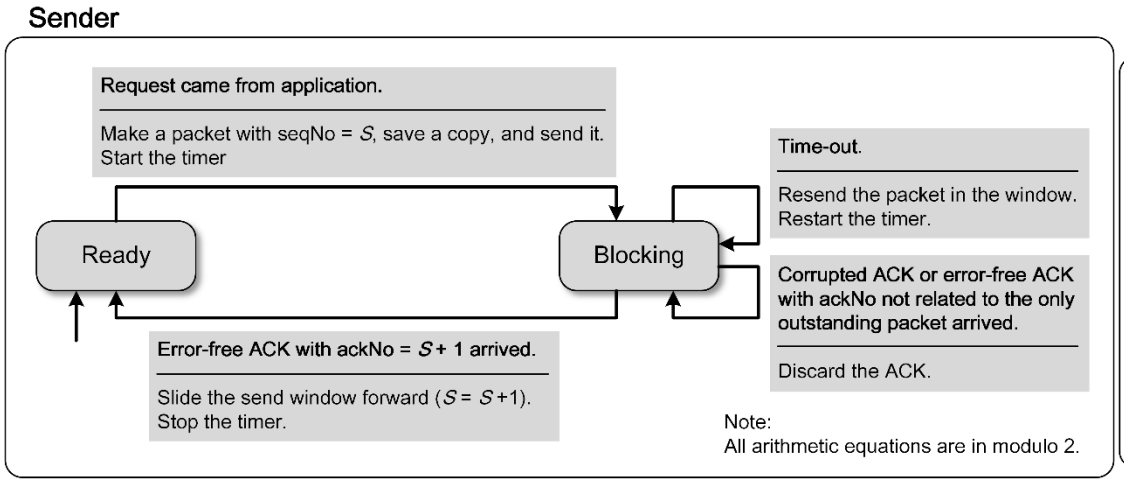
송신측에서 순서 번호 $x$의 패킷을 전송하고 가정한다.
패킷이 오류 없이 안전하게 수신측에 도착하면, 수신측은 확인응답을 전송한다.
이후, 아래와 같은 두 가지 상황이 발생할 수 있다.

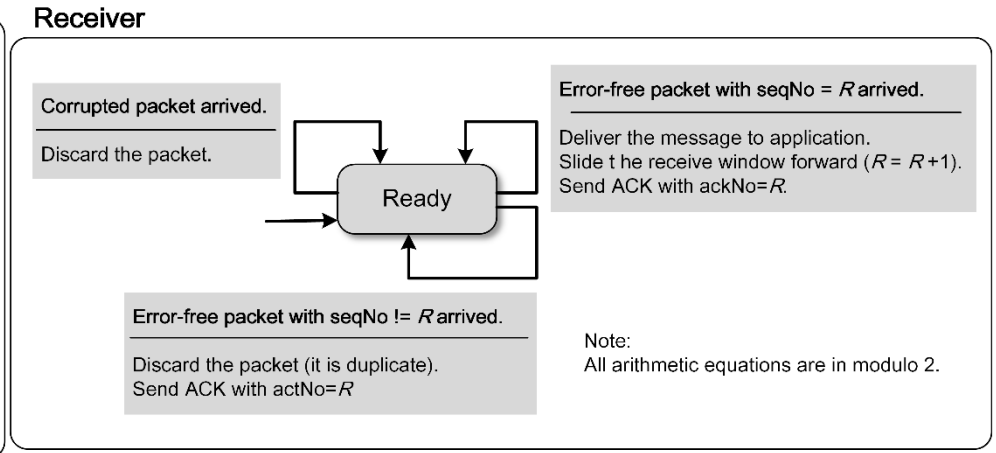
1. 확인응답이 송신측에 도착하면 송신측은 $x+1$의 순서 번호를 갖는 다음 패킷을 전송함
2. 확인응답이 훼손되거나 손실되면, 송신측은 타임아웃 후에 순서 번호 $x$의 패킷을 재전송함. 이때 재전송된 패킷은 순서 번호 $x$를 갖는 중복 패킷임

이 때, 1번상황에서 $x$번과 $x+1$번 패킷에 대한 확인응답을 받는다면, 그 다음 패킷은 $x+2$이 아닌 $x$여도 구분이 가능하다.

# 전송 계층 개요

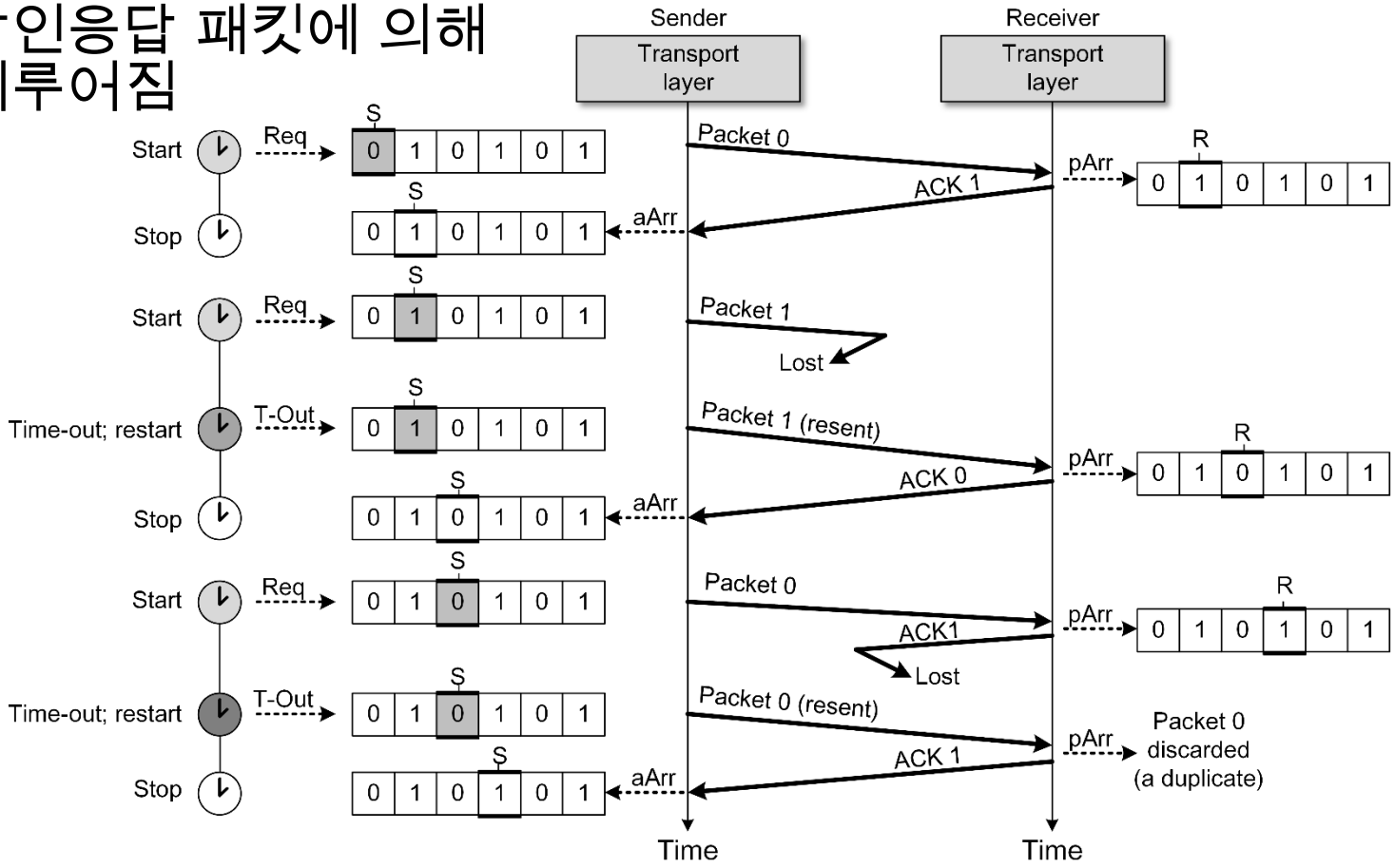
- 전송 계층 프로토콜 (5/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (4/7)
    - 확인응답 번호
      - 수신측에서 받기를 기대하는 다음 패킷의 순서 번호를 나타냄
      - seqNo가 $x$라면, ackNo는 $x+1$의 값을 가짐
    - FSM



# 전송 계층 개요

- 전송 계층 프로토콜 (6/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (5/7)
    - 정지 대기 프로토콜 예시
      - 타이머와 확인응답 패킷에 의해 재전송이 이루어짐



# 전송 계층 개요

- 전송 계층 프로토콜 (7/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (6/7)
    - 효율
      - 채널이 큰 대역폭과 긴 왕복 지연을 가지고 있을 경우 비효율적임
      - 대역폭 지연 곱(Product)을 최대로 사용해야 함
        - 대역폭과 왕복 지연의 곱으로, 수신측으로부터의 확인응답을 기다리면서 송신측에서 채널을 통해 전송할 수 있는 비트의 수를 의미함
    - 예제 13.5

전송 대기 시스템에서 선로의 대역폭이 1 Mbps이고 1비트가 왕복하는 데 20 ms이 걸린다고 가정해보자. 대역폭 지연 곱은 얼마인가?
시스템의 데이터 패킷이 1,000 비트의 길이라면, 선로의 이용율은 얼마인가?

- 대역폭 지연 곱: $(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000$비트
  - 시스템은 왕복 지연 동안 20,000비트를 전송할 수 있음
- 데이터 패킷의 길이에 따라 1,000비트만 전송 가능함
- 선로 이용율: $\frac{1{,}000}{20{,}000} \times 100 = 5\%$

∴ 해당 선로에서 정지 대기 프로토콜을 사용하는 것은 용량 낭비임

# 전송 계층 개요

- 전송 계층 프로토콜 (8/29)
  - 정지 대기(Stop-and-Wait) 프로토콜 (7/7)
    - 예제 13.5

예제 13.5에서 확인응답에 대한 고려 없이 최대 15개의 패킷을 전송할 수 있는 프로토콜을 이용하면 선로의 이용율은 얼마인가?

- 대역폭 지연 곱: $(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000$비트
  - 시스템은 왕복 지연 동안 20,000비트를 전송할 수 있음
- 데이터 패킷의 총 길이에 따라 $15 \times 1{,}000 = 15{,}0000$비트 전송 가능
- 선로 이용율: $\frac{15{,}000}{20{,}000} \times 100 = 75\%$

# 전송 계층 개요

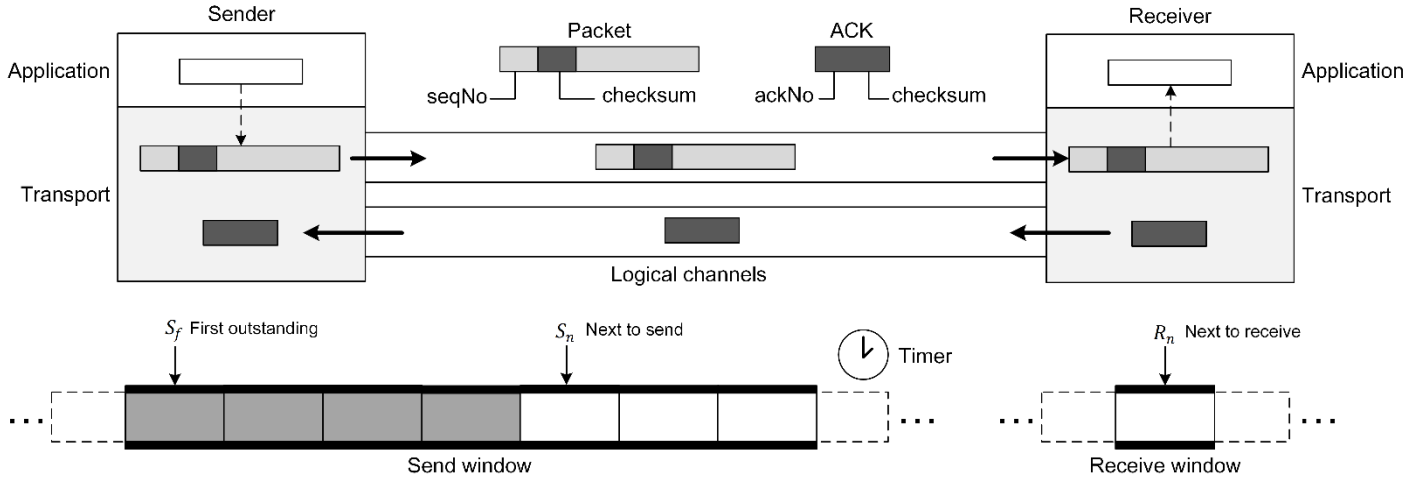
- 전송 계층 프로토콜 (9/29)
  - 파이프라인(Pipeline)
    - 개요
      - 이전 작업이 끝나기 전에 새로운 작업을 시작하는 기법
      - 네트워트 상에서는 송신자가 이전 패킷의 ACK를 받기 전에 추가 패킷을 전송할 때 사용할 수 있음
      - 대역폭이 클수록 더 큰 전송 효율을 낼 수 있음
    - 전송 계층 프로토콜에서의 파이프라인
      - 정지 대기 프로토콜은 파이프라인을 제공하지 않음
        - 확인응답을 수신하거나 또는 타임 아웃이 발생한 후에 재전송이 진행됨
      - N-프레임-후퇴(GBN, Go-Back-N) 프로토콜, 선택적 반복(Selective Repeat) 프로토콜에서는 파이프라인을 제공함

# 전송 계층 개요

- 전송 계층 프로토콜 (10/29)
  - N-프레임-후퇴 프로토콜 (1/11)
    - 개요
      - 슬라이딩 윈도우를 기반으로 오류 제어를 수행하는 연결 지향 프로토콜
      - 송신측은 확인응답을 수신하기 전에 여러 개(N)의 패킷을 전송할 수 있음
      - 수신측은 하나의 패킷만 버퍼에 저장할 수 있음
      - 오류 발생 시 해당 패킷 이후 모든 패킷을 재전송

# 전송 계층 개요

- 전송 계층 프로토콜 (11/29)
  - N-프레임-후퇴 프로토콜 (2/11)
    - 동작
      1. 송신자는 윈도우 크기(N) 내에서 연속적으로 패킷을 전송함
      2. 수신자는 오류 없이 수신한 마지막 패킷 번호를 기준으로 누적 ACK를 전송함
      3. 만약 $i$번째 패킷에 오류가 발생하면 수신자는 $i-1$까지만 확인응답을 보내고, 확인응답을 멈춤
      4. 송신자는 타임아웃 발생 시 $i$번째 패킷 이후 모든 패킷을 재전송함

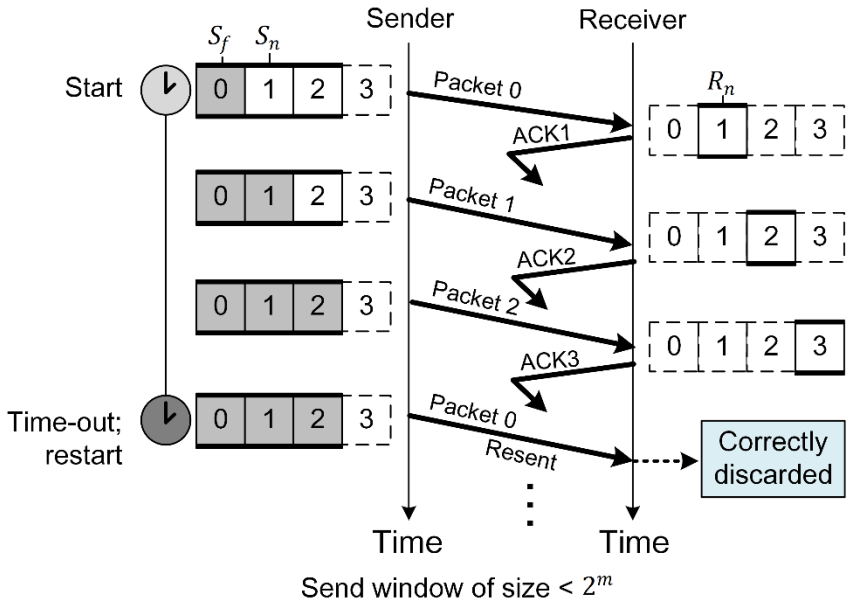


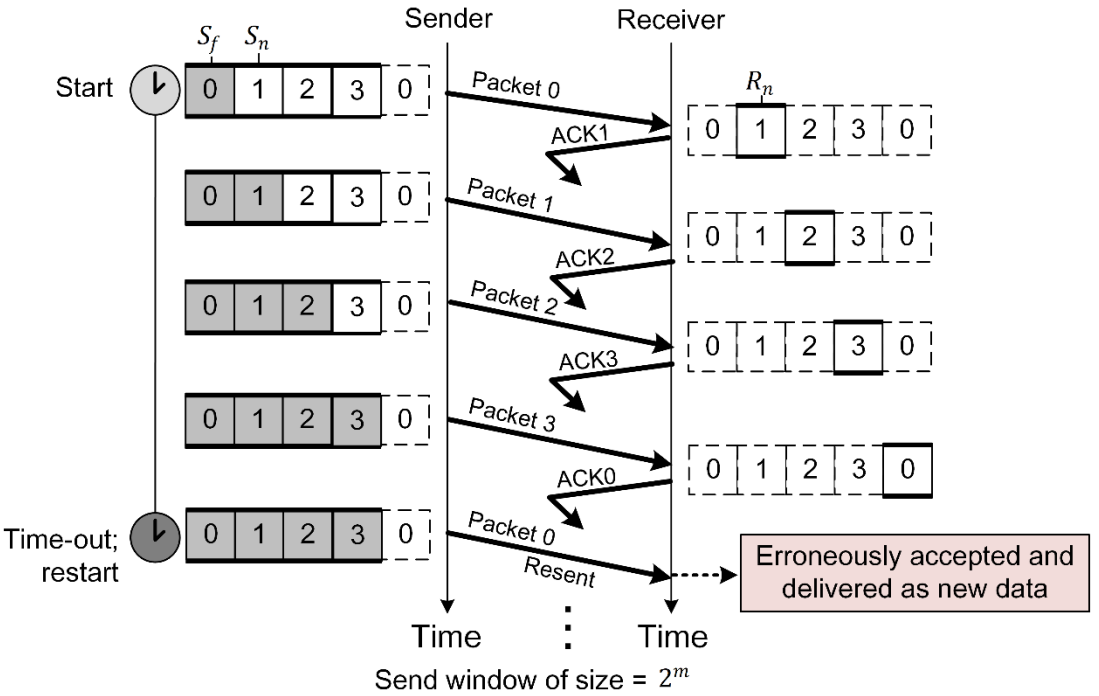
# 전송 계층 개요

- 전송 계층 프로토콜 (12/29)
  - N-프레임-후퇴 프로토콜 (3/11)
    - 순서 번호
      - 순서 번호 필드의 비트 수 $m$에 따라 $modulo\ 2^m$ 까지의 순서 번호를 가짐
    - 확인응답 번호
      - 확인응답은 누적 값으로, 정상적으로 수신된 패킷 중 가장 마지막 순서 번호를 기반으로 확인응답 번호를 나타냄
      - 수신하기를 기대하는 다음 패킷의 순서 번호를 나타냄

# 전송 계층 개요

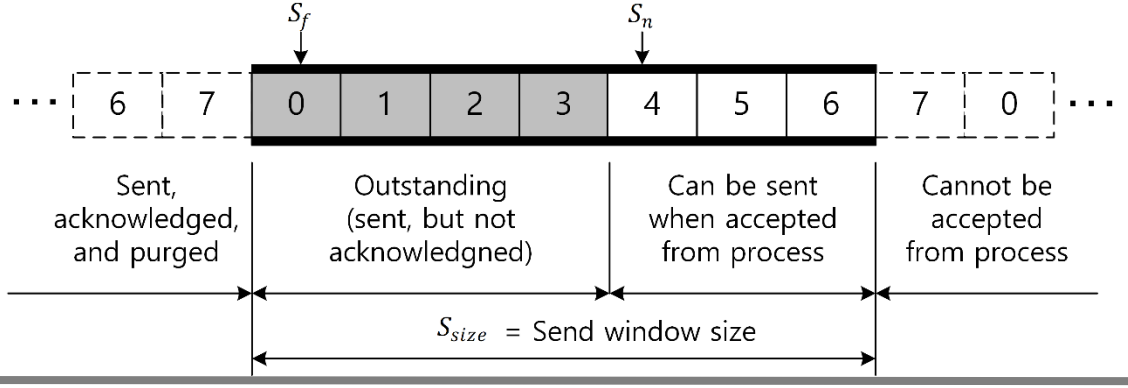
- 전송 계층 프로토콜 (13/29)
  - N-프레임-후퇴 프로토콜 (4/11)
    - 송신 윈도우
      - 개요
        - 전송 중이거나 전송될 데이터 패킷의 순서 번호를 포함하는 가상의 상자
        - 윈도우의 최대 크기는 $2^m - 1$로 고정되어 있음



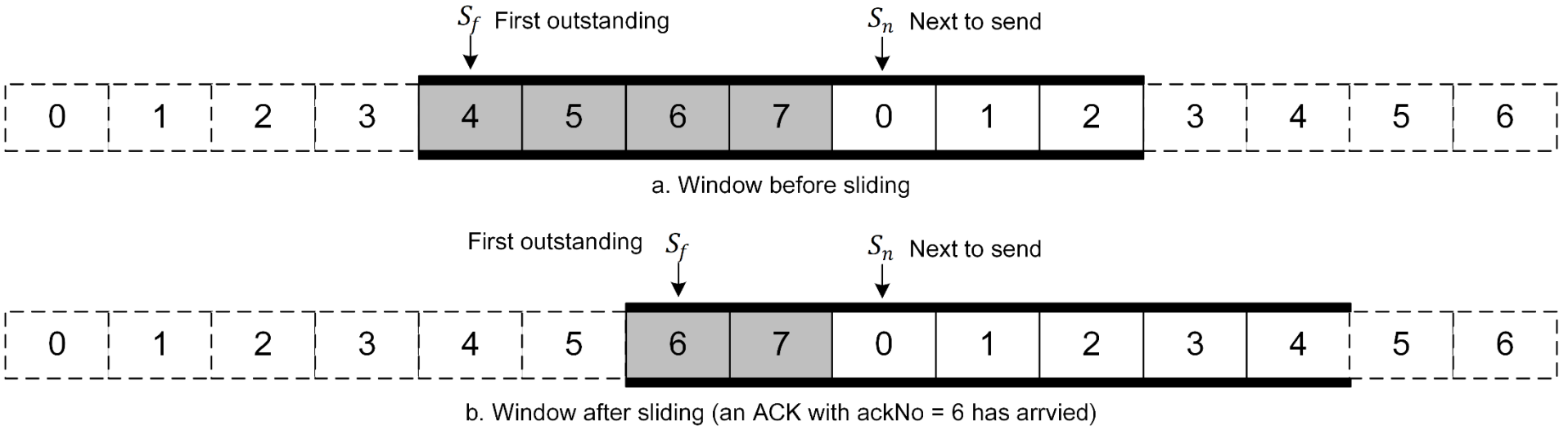
# 전송 계층 개요

- 전송 계층 프로토콜 (14/29)
  - N-프레임-후퇴 프로토콜 (5/11)
    - 송신 윈도우
      - 영역
        - 이미 확인응답된 패킷에 속하는 순서 번호
          - 패킷의 전송이 이미 완료 되었다고 간주하며, 사본을 보관하지 않음
        - 전송은 되었으나 확인응답이 되지 않은 패킷에 속하는 순서 번호
          - 미해결(Outstanding) 패킷 이라고 부름
          - 송신측은 수신측에서 해당 패킷들이 수신되었는지, 손실되었는지 확인을 기다림
        - 전송할 수 있으나 데이터가 응용 계층에서 수신되지 않은 패킷의 순서 번호
        - 윈도우가 이동하기 전까지 패킷을 전송할 수 없는 순서 번호



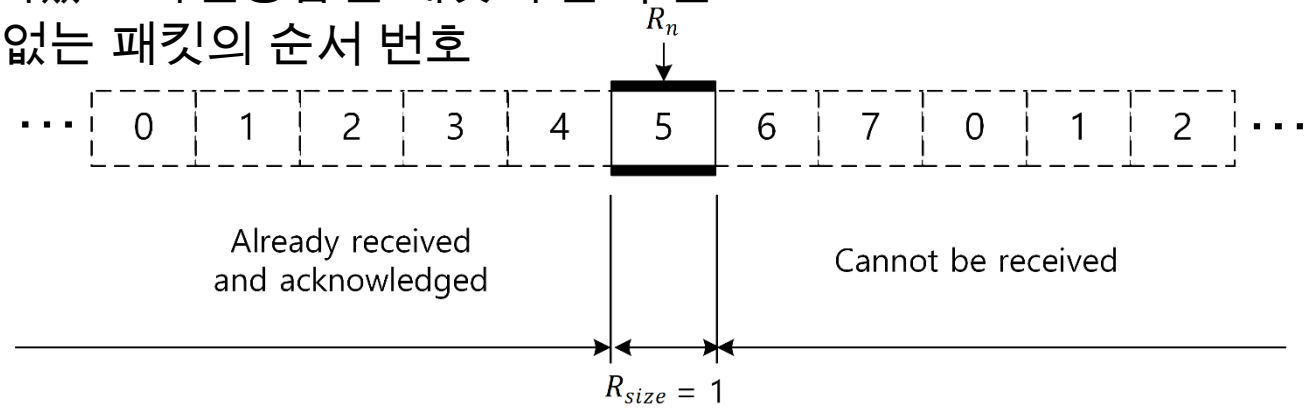
# 전송 계층 개요

- 전송 계층 프로토콜 (15/29)
  - N-프레임-후퇴 프로토콜 (6/11)
    - 송신 윈도우
      - e.g., ackNo=6인 확인응답을 수신한 경우 송신 윈도우의 이동 방법



# 전송 계층 개요

- 전송 계층 프로토콜 (16/29)
  - N-프레임-후퇴 프로토콜 (7/11)
    - 수신 윈도우
      - 개요
        - 수신측에서 허용할 수 있는 데이터 패킷의 순서 번호를 포함하는 가상의 상자
        - 윈도우의 크기는 프로토콜에 따라 달라지며, GBN 프로토콜에서는 1임
        - 올바른 패킷이 수신되면 $R_n = (R_n + 1)\ modulo\ 2^m$으로 이동함
      - 영역
        - $R_n$: 수신되며 확인응답할 패킷의 순서 번호
        - 폐기될 순서번호
          - 이미 수신되었고 확인응답된 패킷의 순서 번호
          - 수신될 수 없는 패킷의 순서 번호

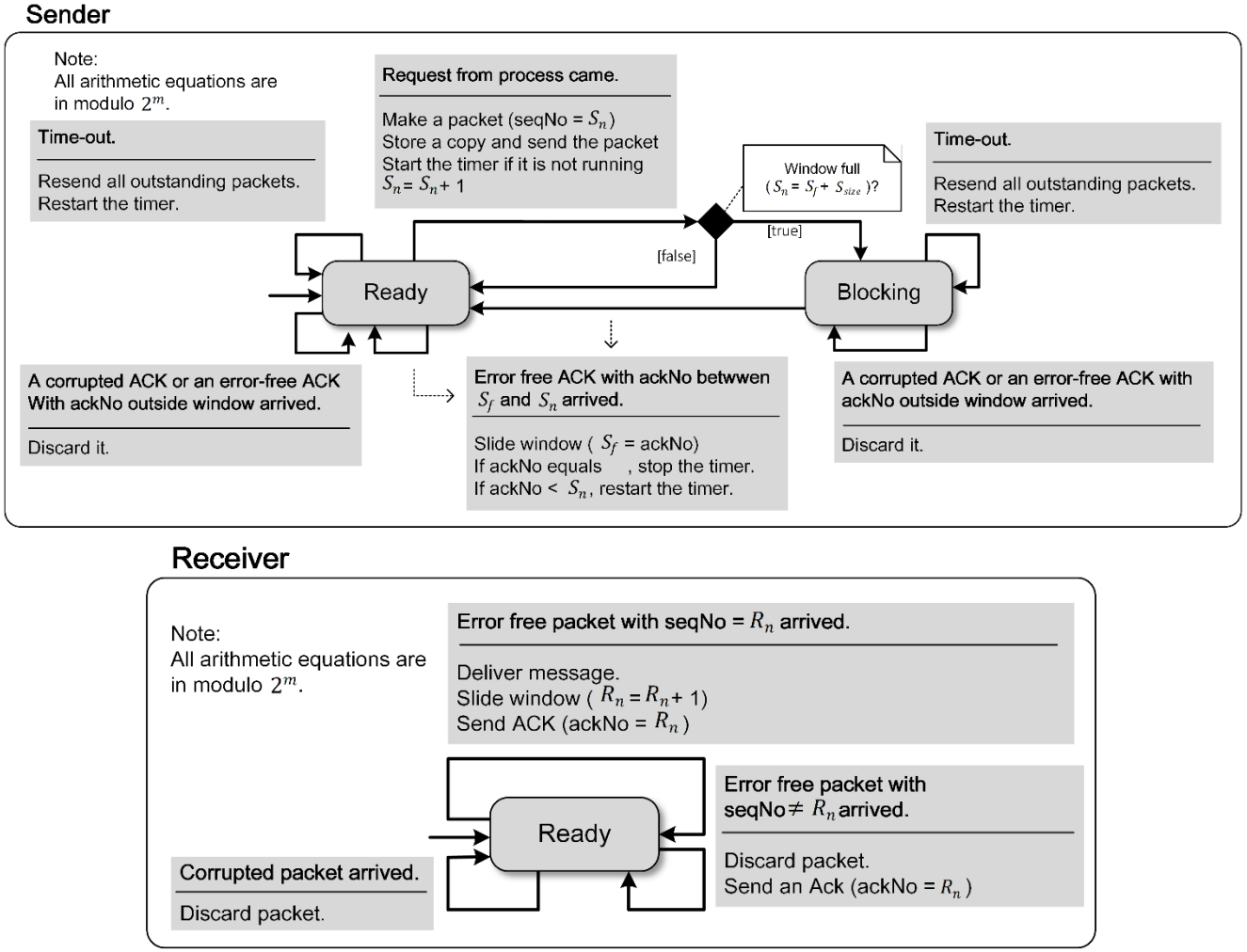


# 전송 계층 개요

- 전송 계층 프로토콜 (17/29)
  - N-프레임-후퇴 프로토콜 (8/11)
    - 타이머
      - 가장 먼저 보낸 첫 번째 미해결 패킷에만 타이머를 둠
        - $i$번째 패킷에 대한 확인응답이 없으면, $i$번째 뒤의 모든 패킷을 재전송하므로 첫 번째 미해결 패킷의 타이머만으로 해결할 수 있음
    - 패킷 재전송
      - 타이머가 만료되면, 송신측은 모든 미해결 패킷들을 재전송함
      - 타임아웃 발생 시 윈도우 크기인 N만큼 후퇴하며 재전송이 발생함

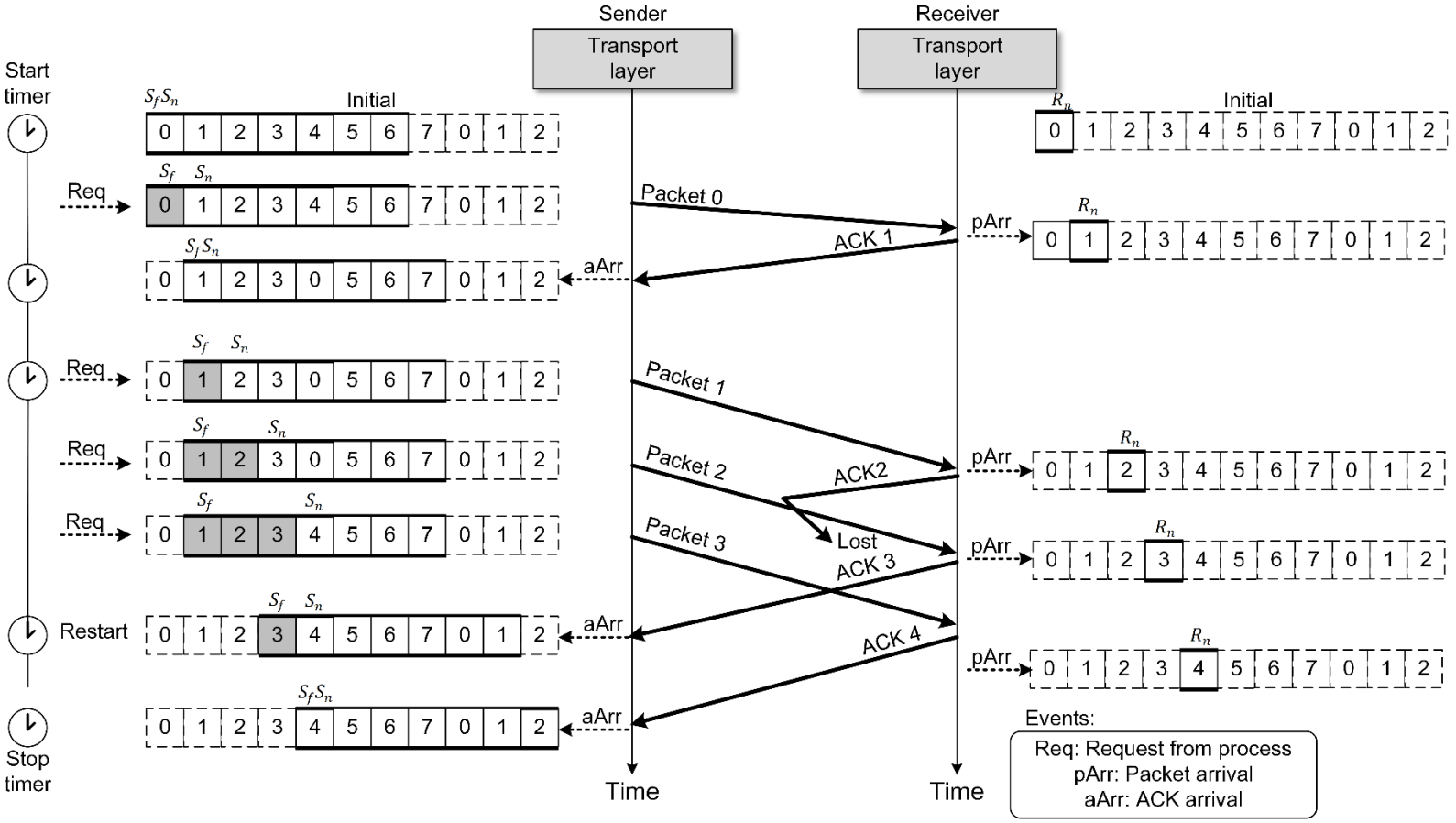
# 전송 계층 개요

- 전송 계층 프로토콜 (18/29)
  - N-프레임-후퇴 프로토콜 (9/11)
    - FSM



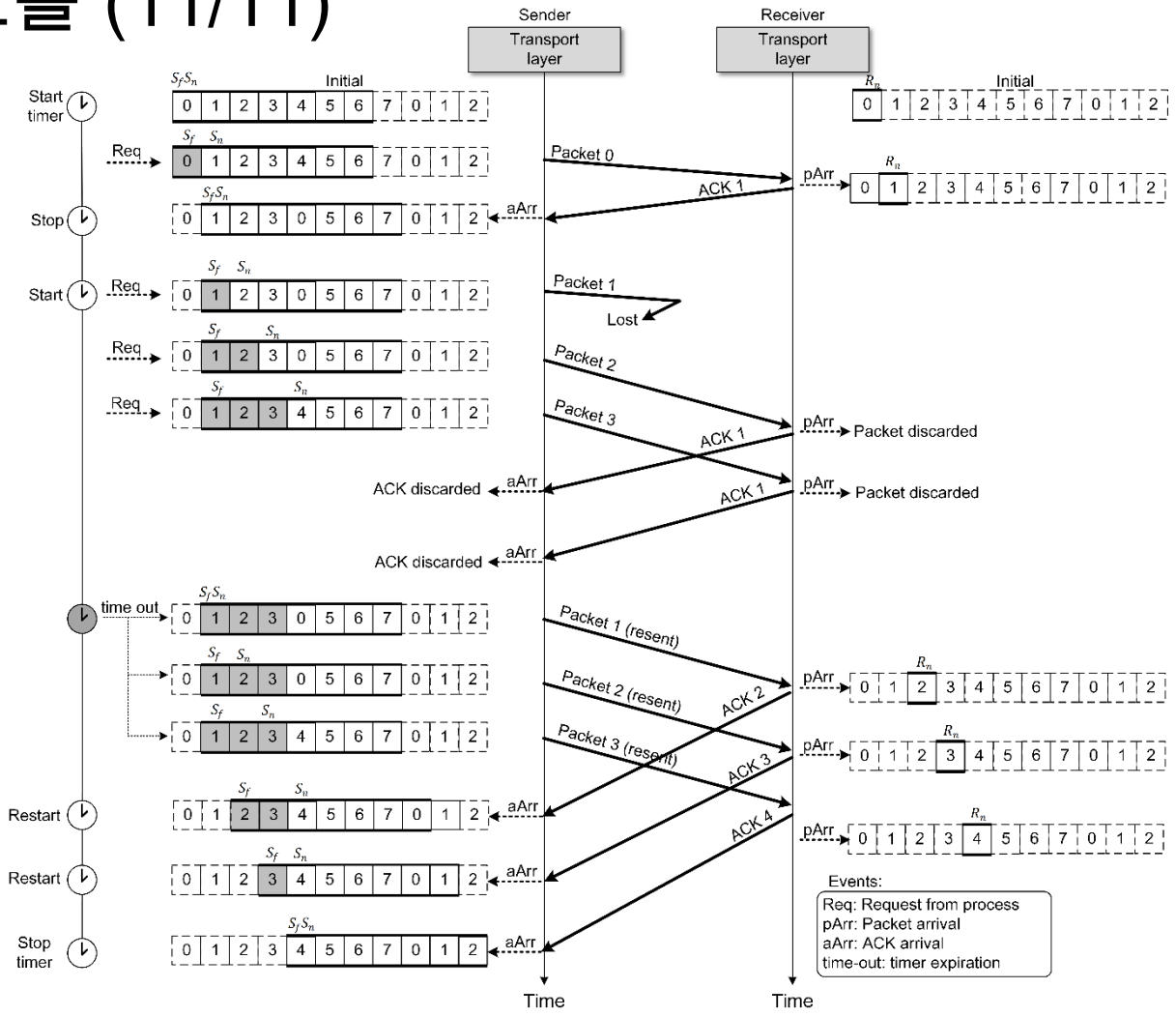
# 전송 계층 개요

- 전송 계층 프로토콜 (19/29)
  - N-프레임-후퇴 프로토콜 (10/11)
    - ACK의 일부가 지연되거나 손실된 예시



# 전송 계층 개요

- 전송 계층 프로토콜 (20/29)
  - N-프레임-후퇴 프로토콜 (11/11)
    - 패킷이 손실된 예시

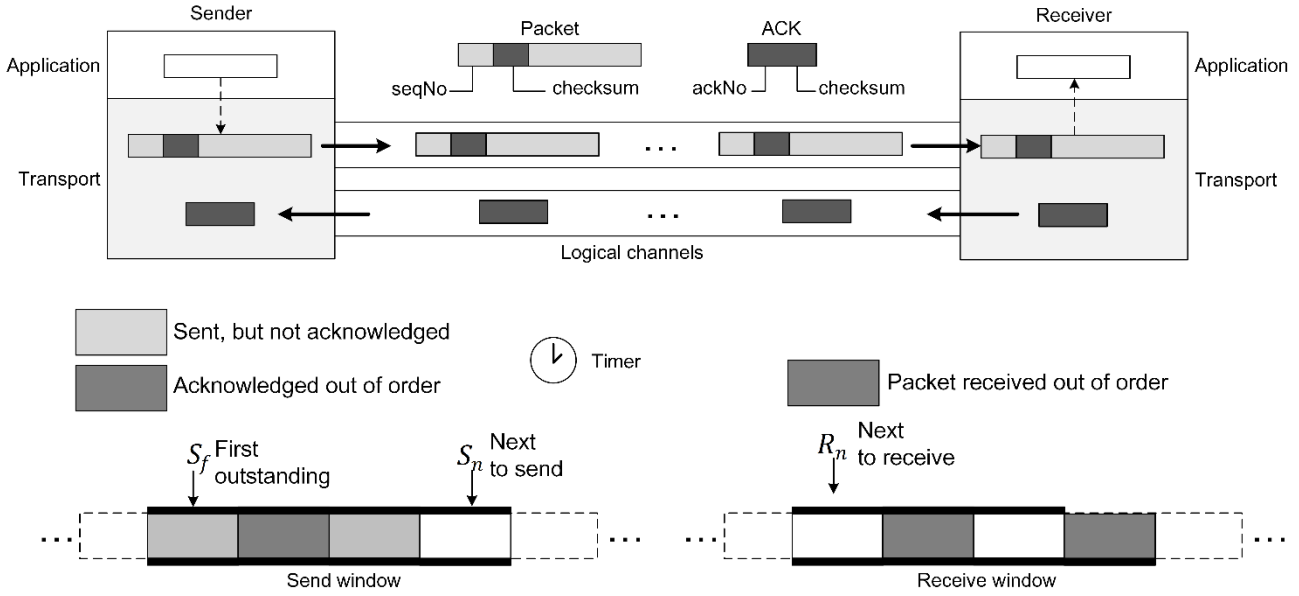


# 전송 계층 개요

- 전송 계층 프로토콜 (21/29)
  - 선택적 반복 프로토콜 (1/8)
    - 개요
      - 슬라이딩 윈도우를 기반으로 오류 제어를 수행하는 연결 지향 프로토콜
      - 송신측은 확인응답을 수신하기 전에 여러 개(N)의 패킷을 전송할 수 있음
      - 수신측은 여러 개(N)의 패킷을 버퍼에 저장할 수 있음
      - 오류 발생 시 손실된 패킷만 선택적으로 재전송

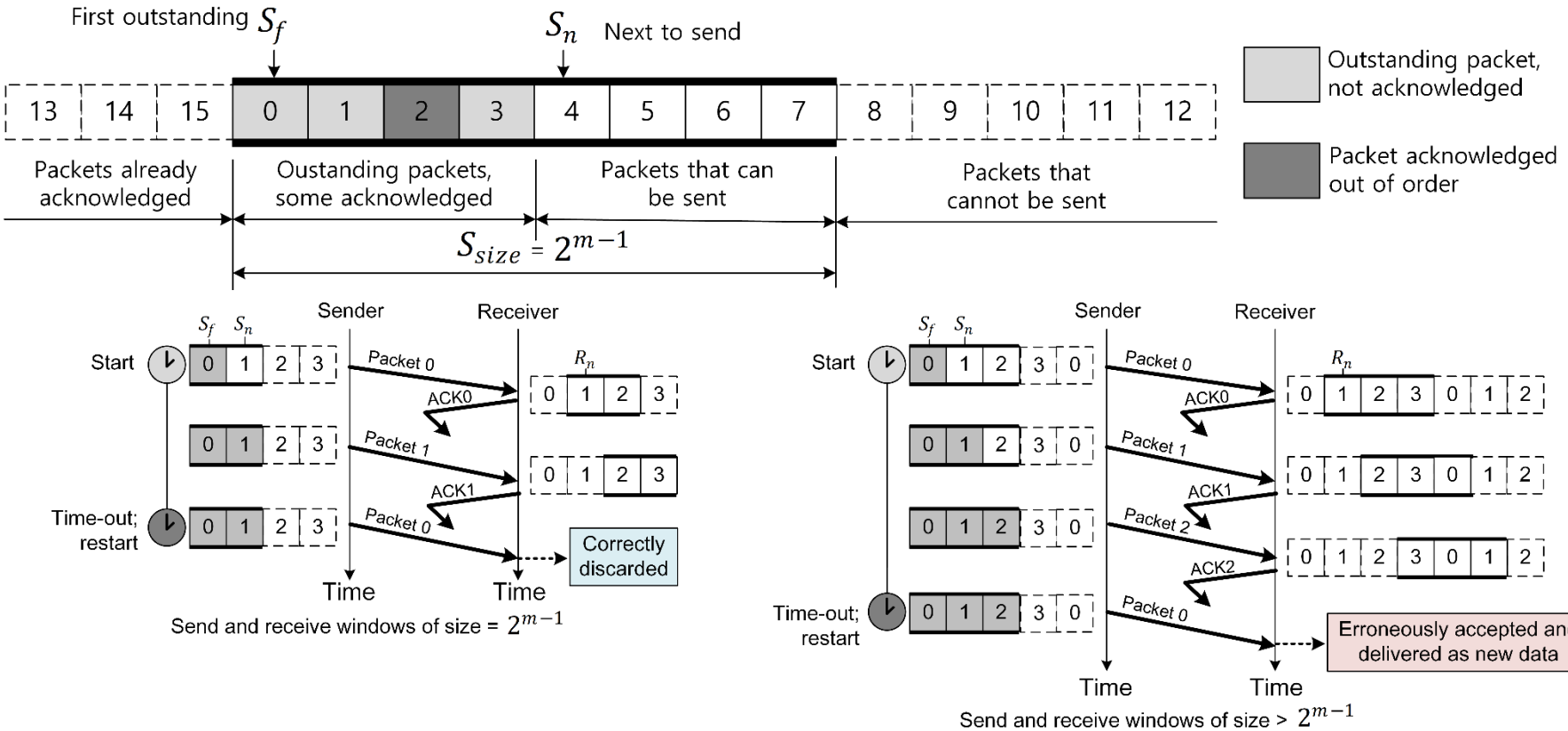
# 전송 계층 개요

- 전송 계층 프로토콜 (22/29)
  - 선택적 반복 프로토콜 (2/8)
    - 동작
      1. 송신자는 윈도우 크기(N) 내에서 여러 개의 패킷을 전송함
      2. 수신자는 패킷 수신 시 개별 ACK를 전송 함
      3. 송신자는 ACK가 누락된 패킷만 재전송함
      4. 수신자는 올바른 순서로 버퍼에 패킷을 저장 함



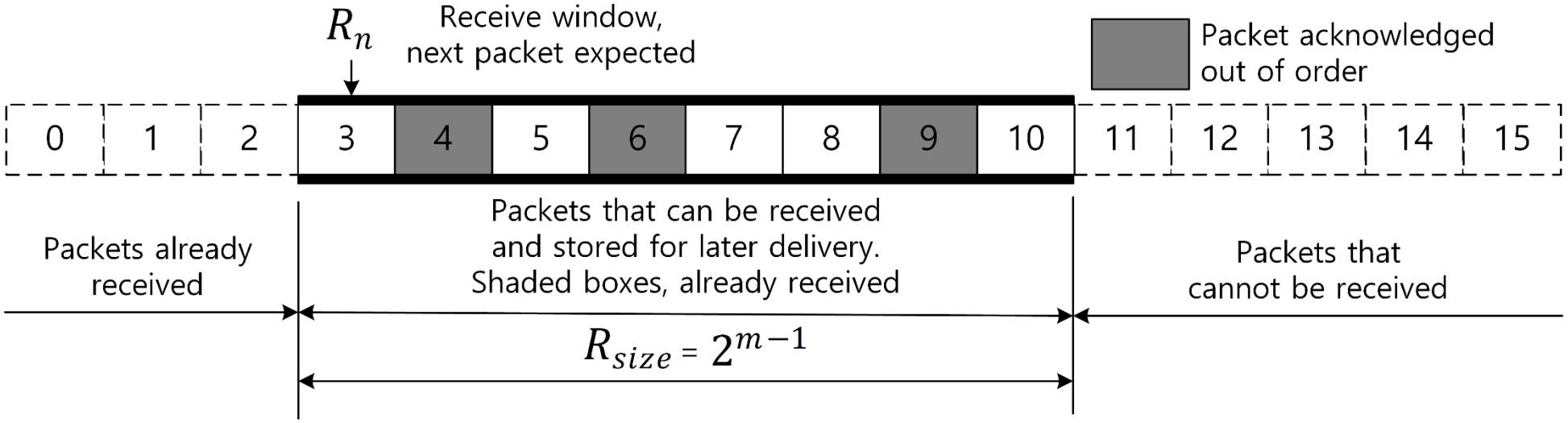
# 전송 계층 개요

- 전송 계층 프로토콜 (23/29)
  - 선택적 반복 프로토콜 (3/8)
    - 송신 윈도우
      - 송신 윈도우의 크기는 $2^{m-1}$로 고정되어있음



# 전송 계층 개요

- 전송 계층 프로토콜 (24/29)
  - 선택적 반복 프로토콜 (4/8)
    - 수신 윈도우
      - 송신 윈도우와 동일한 크기($2^{m-1}$)를 가짐
      - 재전송으로 순서에 어긋나게 도착한 패킷들은 버퍼에 저장됨
      - 신뢰성 있는 프로토콜이므로 순서를 맞추기 위해 응용 계층으로 전달하기 전에 먼저 전송되었으나 수신되지않은 패킷들을 기다려야 함



# 전송 계층 개요

- 전송 계층 프로토콜 (25/29)
  - 선택적 반복 프로토콜 (5/8)
    - 타이머
      - 이론상으로는 미해결 패킷마다 하나씩의 타이머를 사용함
      - 선택적 반복 프로토콜을 구현하는 대부분의 전송 계층 프로토콜에서는 하나의 타이머만 사용함
    - 확인 응답
      - ackNo는 오류 없이 수신된 하나의 단일 패킷의 순서 번호를 나타냄
      - 다른 패킷들에 대한 피드백 없이 독립적으로 작용함

# 전송 계층 개요

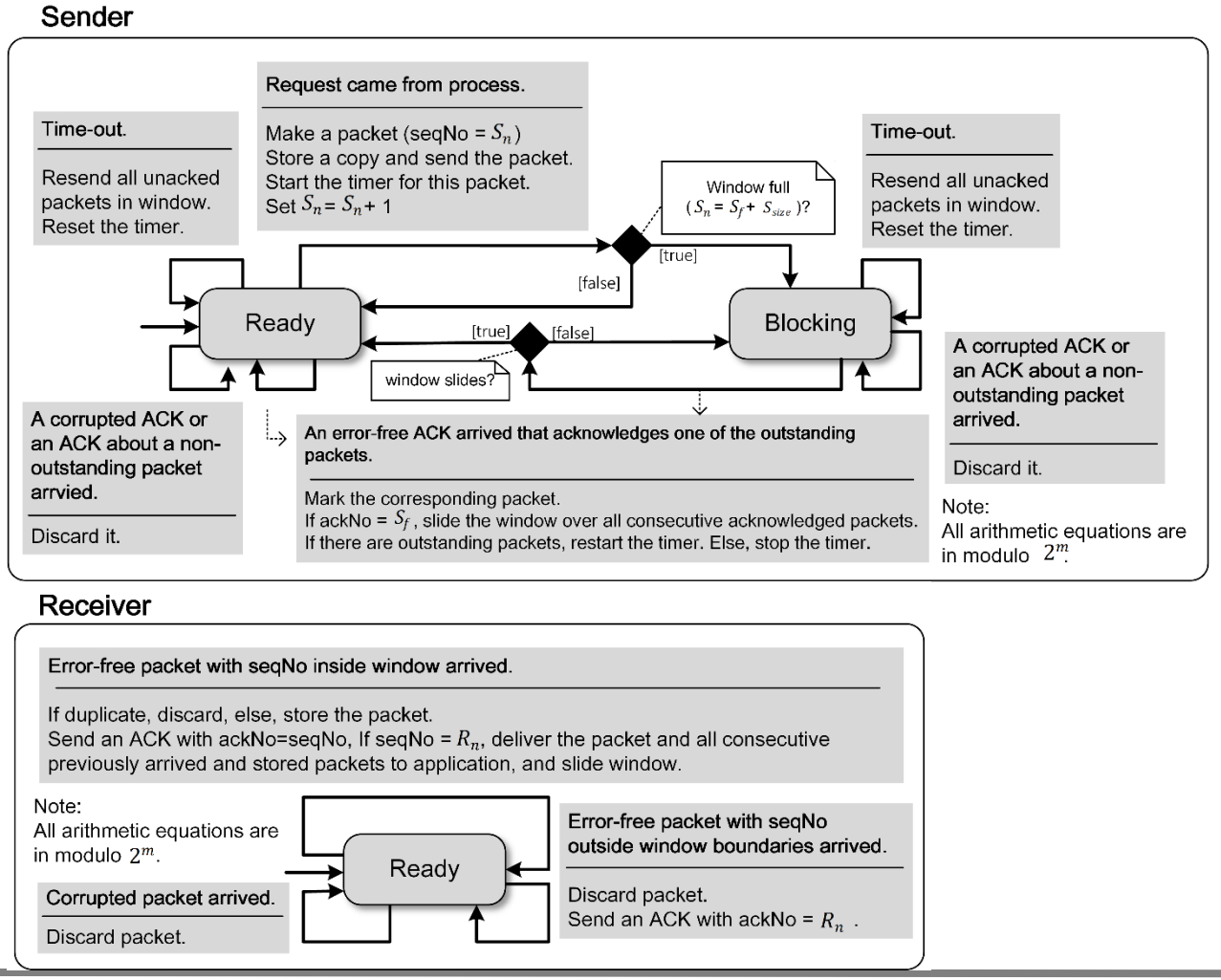
- 전송 계층 프로토콜 (26/29)
  - 선택적 반복 프로토콜 (6/8)
    - 예제 13.9

송신측에서 패킷 0, 1, 2, 3, 4, 5의 6개의 패킷을 전송한다고 가정해보자.
송신측이 ackNo=3의 ACK를 수신하였다. 시스템이 GBN 또는 SR을 사용하는 경우에 ACK 수신은 무엇을 의미하는가?

- GBN을 이용하는 시스템
  - 패킷 0, 1, 2가 훼손되지 않고 잘 수신되었음
- SR을 이용하는 시스템
  - 패킷 3이 훼손되지 않고 잘 수신되었음
  - 다른 패킷들에 대한 정보는 없음

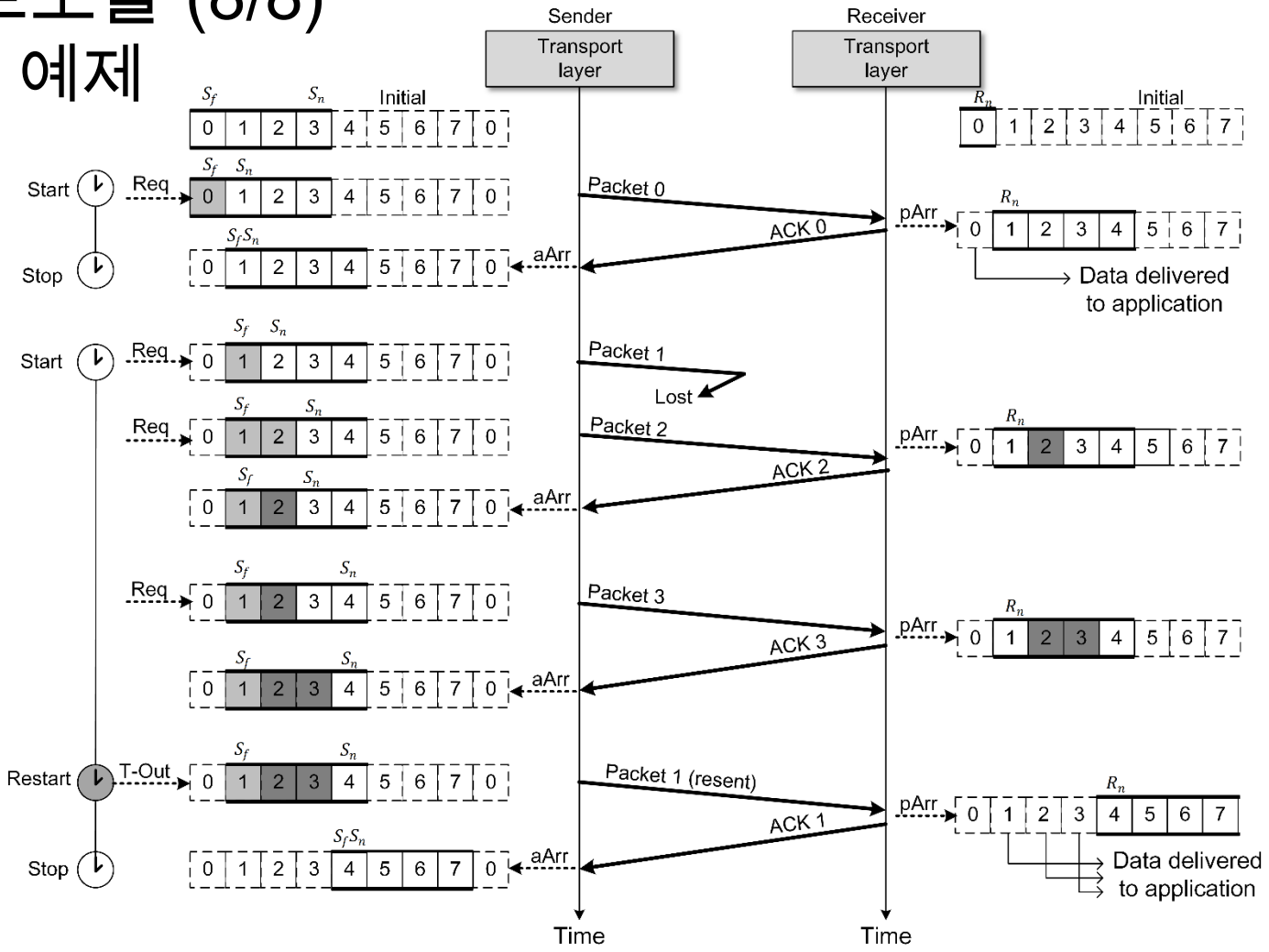
# 전송 계층 개요

- 전송 계층 프로토콜 (27/29)
  - 선택적 반복 프로토콜 (7/8)
    - FSM



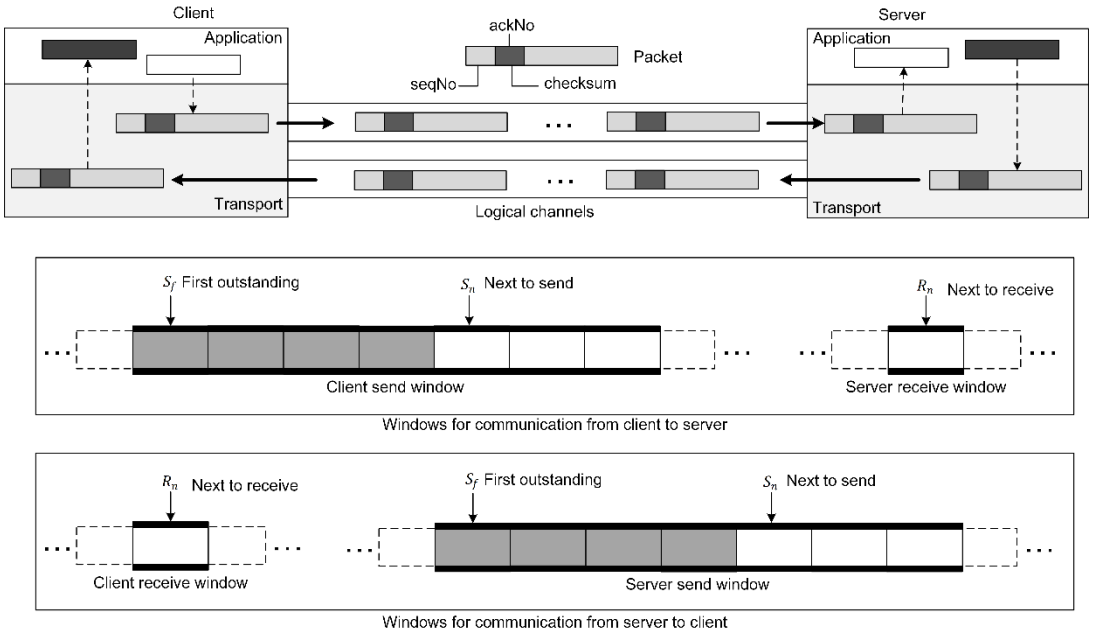
# 전송 계층 개요

- 전송 계층 프로토콜 (28/29)
  - 선택적 반복 프로토콜 (8/8)
    - 패킷이 손실된 예제



# 전송 계층 개요

- 전송 계층 프로토콜 (29/29)
  - 양방향 프로토콜: 피기배킹
    - 개요
      - 일반적인 데이터 전송은 단방향임
      - 실제 통신은 클라이언트와 서버간 양방향 데이터 전송이 이루어짐
      - 양방향 통신의 효율을 향상시키기 위해 사용함
    - 동작
      - 데이터 전송 시, 양방향 모두 패킷에 데이터와 이전에 수신한 데이터에 대한 ACK 정보를 실어서 보냄

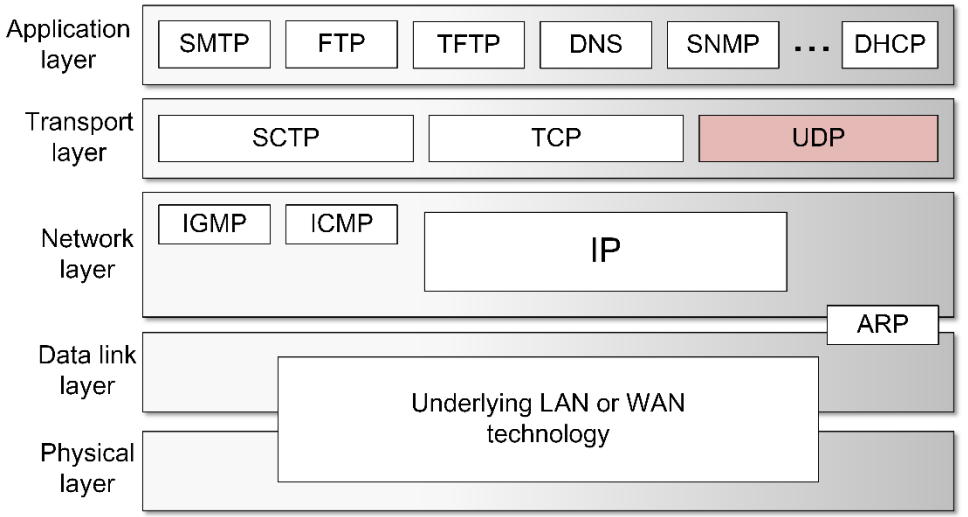


# 목 차

- 보충
- 전송 계층 개요
- UDP (User Datagram Protocol)

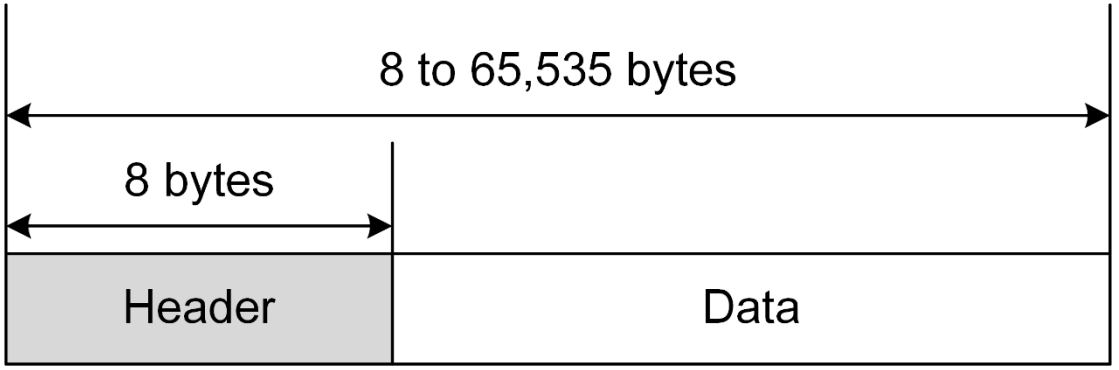
# UDP

- 개요
  - 정의
    - 연결 설정 없이 데이터를 빠르게 전송하는데 사용되는 비연결성 통신 프로토콜
  - 역할
    - 포트 번호를 사용하여 프로세스 대 프로세스 통신 생성
    - 전송 계층에서 기본적인 오류 제어 메커니즘 제공
      - 수신된 패킷에서 오류를 탐지하면 폐기함
  - 특징
    - 신뢰성을 제공하지 않음
    - 최소한의 오버헤드만 사용함



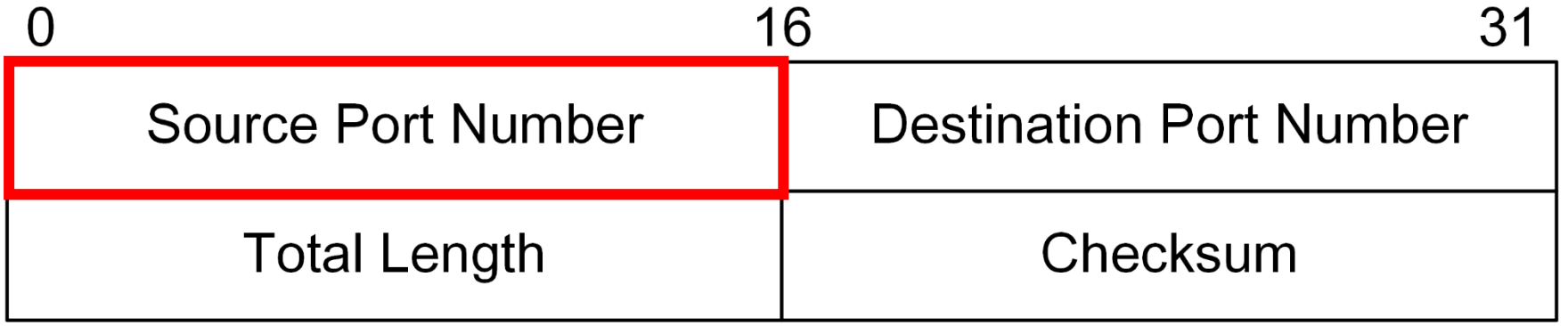
# UDP

- 사용자 데이터그램(User Datagram) (1/6)
  - 개요
    - UDP에서 사용하는 데이터 단위
    - 8바이트의 고정 헤더 크기를 가짐
    - 헤더를 포함한 전체는 8~65,535바이트의 크기를 가짐



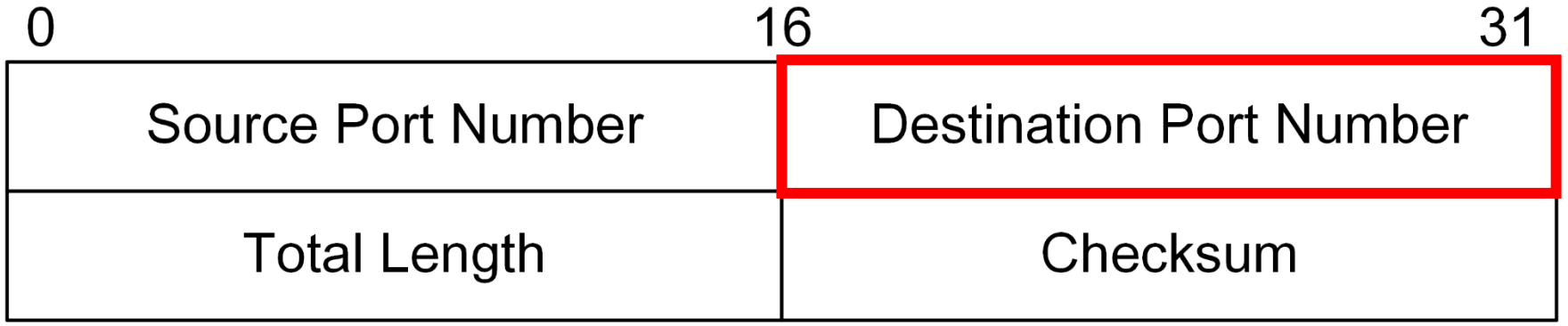
# UDP

- 사용자 데이터그램(User Datagram) (2/6)
  - 헤더 구조 (1/3)
    - 발신지 포트 번호(Source Port Number)
      - 발신지 호스트 상에서 수행되는 프로세스가 사용하는 포트 번호
      - 16비트의 크기를 가지므로 0~65,535의 범위 내에서 사용 가능
      - 발신지 호스트가 요청을 전송하는 클라이언트인 경우, 임시 포트 번호를 가짐
      - 발신지 호스트가 응답을 전송하는 서버라면 잘 알려진 포트번호를 가짐



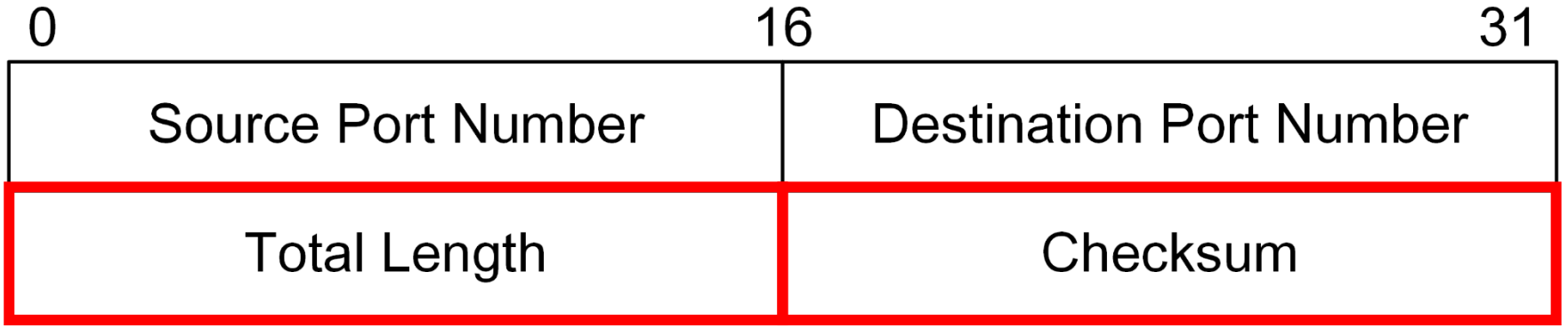
# UDP

- 사용자 데이터그램(User Datagram) (3/6)
  - 헤더 구조 (2/3)
    - 목적지 포트 번호(Destination Port Number)
      - 목적지 호스트 상에서 수행되는 프로세스가 사용하는 포트 번호
      - 16비트의 크기를 가지므로 0~65,535의 범위 내에서 사용 가능
      - 목적지 호스트가 클라이언트에 의해 요청이 보내지는 서버라면 잘 알려진 포트번호를 가짐
      - 목적지 호스트가 응답을 받는 클라이언트인 경우, 임시 포트 번호를 가짐



# UDP

- 사용자 데이터그램(User Datagram) (4/6)
  - 헤더 구조 (3/3)
    - 길이(Total Length)
      - 헤더와 데이터를 합한 사용자 데이터그램의 전체 길이
      - 16비트의 크기를 가지므로 0~65,535 사이의 길이 정의 가능
        - IP 데이터그램 내에 UDP 사용자 데이터그램이 포함되므로 실제의 UDP 길이는 65,535 보다 작은 크기를 가짐
    - 체크섬(Checksum)
      - 헤더와 데이터를 모두 포함한 사용자 데이터그램 전체에 대해 오류를 탐지하기 위해 사용



# UDP

- 사용자 데이터그램(User Datagram) (5/6)
  - 예제 14.1

다음은 16진수의 형식으로 UDP 헤더를 덤프한 것이다.

**CB84 000D 001C 001C**

아래의 질문에 답하라.

a. 발신지 포트 번호는 얼마인가
   - 처음의 네 개의 16진수 0xCB84이며, 십진수로 표현하면 52,100

b. 목적지 포트 번호는 얼마인가?
   - 두 번째 네 개의 16진수 0x000D이며, 십진수로 표현하면 13

c. 사용자 데이터그램의 총 길이는 얼마인가?
   - 세 번째 네 개의 16진수 0x001C이며, 십진수로 표현하면 28바이트

# UDP

- 사용자 데이터그램(User Datagram) (6/6)
  - 예제 14.1

다음은 16진수의 형식으로 UDP 헤더를 덤프한 것이다.

CB84 000D 001C 001C

아래의 질문에 답하라.

d. 데이터의 길이는 얼마인가?
   - 총 길이가 28바이트이고, 헤더의 길이가 8바이트이므로 28-8=20바이트

e. 데이터의 전송 방향이 클라이언트로부터 서버 쪽으로인가, 아니면 반대의 방향인가?
   - 목적지 포트가 13(잘 알려진 포트)이므로, 클라이언트로부터 서버로 패킷이 전송 됨

# UDP

- UDP 서비스 (1/14)
  - 프로세스 대 프로세스 통신
    - IP 주소와 포트 번호로 구성된 소켓을 이용하여 프로세스 대 프로세스 통신을 제공함

▼ UDP 에서 사용되는 잘 알려진 포트

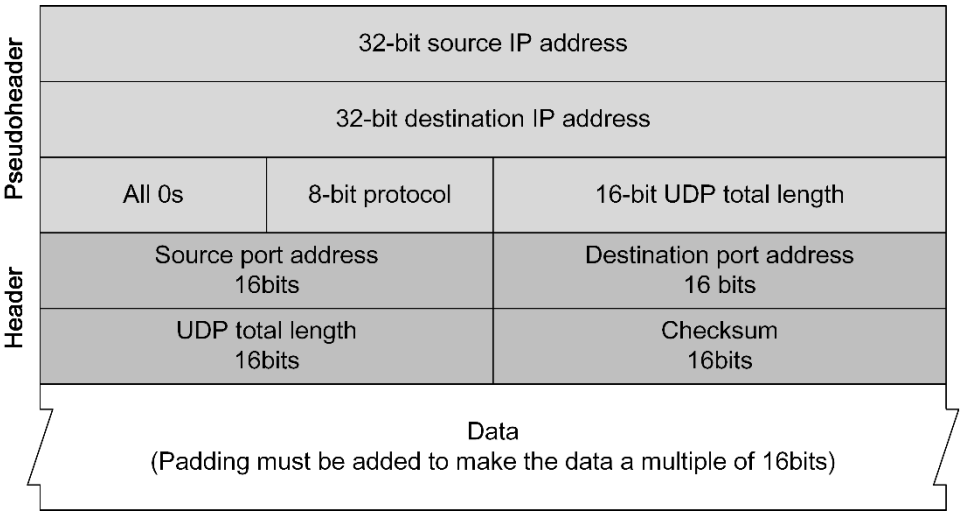
| Port | Protocol | Description | Port | Protocol | Description |
|---|---|---|---|---|---|
| 7 | Echo | Echoes a received datagram back to the sender | 67 | Bootps | Server port to download bootstrap information |
| 9 | Discard | Discards any datagram that is received | 68 | Bootpc | Client port to download bootstrap information |
| 11 | Users | Active users | 69 | TFTP | Trivial File Transfer Protocol |
| 13 | Daytime | Returns the date and the time | 111 | RPC | Remote Procedure Call |
| 17 | Quote | Returns a quote of the day | 123 | NTP | Network Time Protocol |
| 19 | Chargen | Returns a string of characters | 161 | SNMP | Simple Network Management Protocol |
| 53 | Domain | Domain Name Service(DNS) | 162 | SNMP | Simple Network Management Protocol (trap) |

# UDP

- UDP 서비스 (2/14)
  - 비연결형 서비스
    - 연결 설정/해제 과정이 없음
    - 여러 개의 데이터그램이 같은 목적지로 가더라도 각각의 데이터그램은 독립적으로 전송됨
      - 보낸 순서와 다르게 도착할 수 있음
      - 서로 다른 경로로 전송될 수 있음
    - 긴 데이터 스트림에 대해 분할/재조립이 처리되지 않음
      - 충분히 작은 크기로 애플리케이션이 직접 조절해서 전달해야 함
      - 65,535 – UDP 헤더의 8바이트 – IP헤더의 20바이트 = 65,507 이하
  - 흐름 제어
    - 흐름 제어 기능이 존재하지 않음
      - 윈도우 메커니즘이 존재하지 않음
    - 수신측에서는 수신되는 메시지로 인해 오버플로우가 발생할 수 있음

# UDP

- UDP 서비스 (3/14)
  - 혼잡 제어
    - 비연결형 프로토콜 이므로 혼잡 제어를 제공하지 않음
  - 오류 제어
    - 체크섬이 유일한 오류 제어 메커니즘
    - 메시지의 손실이나 중복을 송신자가 알 수 없음
    - 체크섬으로 오류를 감지할 경우 폐기함
      - 의사 헤더(Pseudoheader), UDP 헤더, 응용 계층으로부터 온 데이터의 세 부분을 포함하여 체크섬 계산
    - 송신자가 체크섬을 계산하지 않을 수 있음
      - 체크섬 필드를 0으로 설정하여 전송함



- UDP 서비스 (4/14)
  - 오류 제어
    - 예제 14.2

UDP 사용자 데이터그램의 체크섬을 계산하라

| | | | |
|---|---|---|---|
| 153.18.8.105 | | | |
| 171.2.14.10 | | | |
| All 0s | 17 | 15 | |
| 1087 | | 13 | |
| 15 | | Checksum (All 0s) | |
| T | E | S | T |
| I | N | G | Pad |

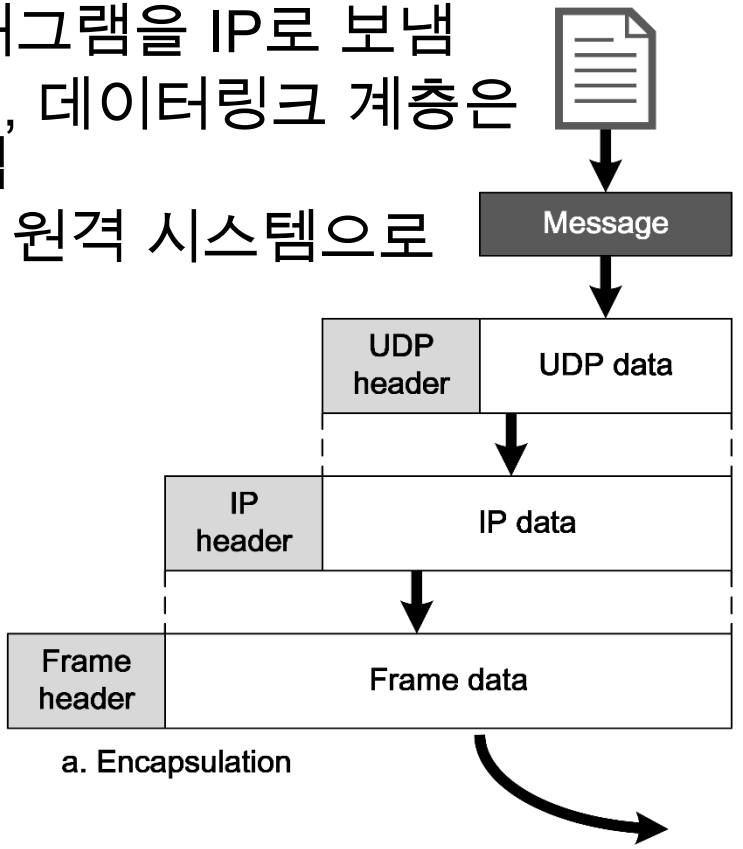
```
10011001 00010010 ──→ 153. 18
00001000 01101001 ──→ 8. 105
10101011 00000010 ──→ 171. 2
00001000 00001010 ──→ 14. 10
00000000 00010001 ──→ 0 / 17
00000000 00001111 ──→ 15
00000100 00111111 ──→ 1087
00000000 00001101 ──→ 13
00000000 00001111 ──→ 15
00000000 00000000 ──→ 0 (checksum)
01010100 01000101 ──→ T / E
01010011 01010100 ──→ S / T
01001001 01001110 ──→ I / N
01000111 00000000 ──→ G  0(pad)
-----------------------------
10010110 11101011 ──→ Sum
01101001 00010100 ──→ Checksum
```

- UDP 서비스 (5/14)
  - 오류 제어
    - 예제 14.3

다음의 가상 상황에서 체크섬으로 어떤 값이 전송되는가?
a. 송신자는 체크섬을 포함하지 않기로 결정한다.
b. 송신자는 체크섬을 포함하기로 결정했지만, 체크섬의 값이 모두 1이다.
c. 송신자는 체크섬을 포함하기로 결정했지만, 체크섬의 갑이 모두 0이다.

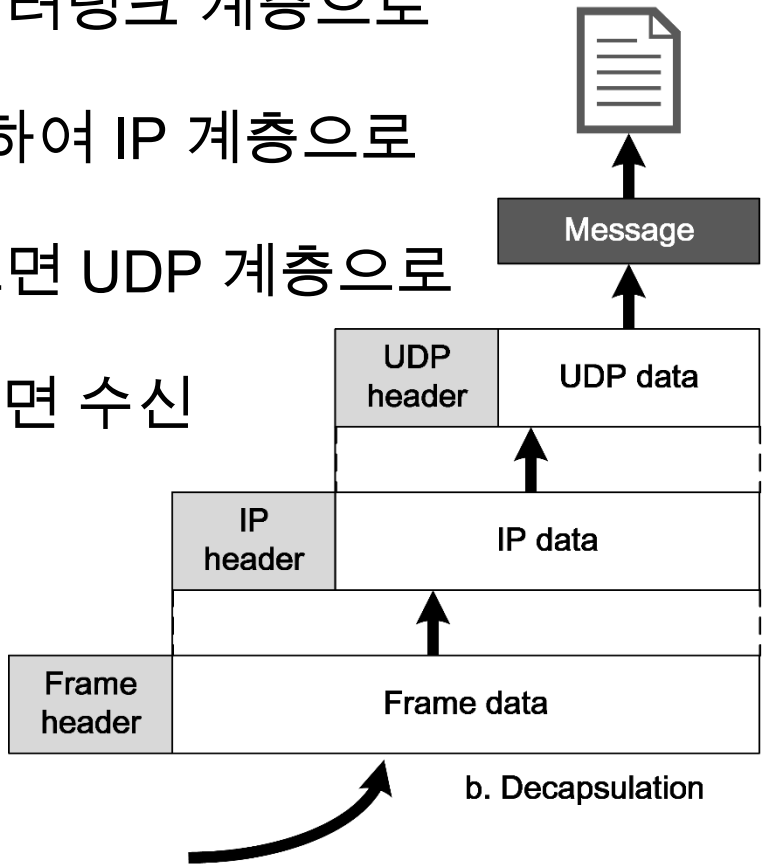
a. 체크섬이 계산되지 않았다는 것을 알리기 위하여 체크섬 필드의 값은 모두 0으로 설정됨
b. 송신자가 합에 보수를 취한 결과가 모두 0이면, 전송하기 전에 결과 값에 다시 보수를 취하여 모두 1인 값을 보냄
   - a의 경우와 혼돈을 피하기 위함
c. 합의 계산에 포함되는 모든 항목의 값이 0인 상황은 불가능함
   - 1의 보수 덧셈을 수행해서 자리올림이 발생할 경우, 1을 더하므로 체크섬 필드의 최소 값은 1임

- UDP 서비스 (6/14)
  - 캡슐화와 역 캡슐화 (1/2)
    - 캡슐화
      - 송신 프로세스가 생성한 메시지에 UDP 헤더 추가
      - UDP는 소켓 주소와 함께 사용자 데이터그램을 IP로 보냄
      - IP는 IP 헤더를 붙여 네트워크 계층으로, 데이터링크 계층은 프레임 헤더를 붙여 물리 계층으로 보냄
      - 물리 계층은 비트를 신호로 부호화하여 원격 시스템으로 전달함



# UDP

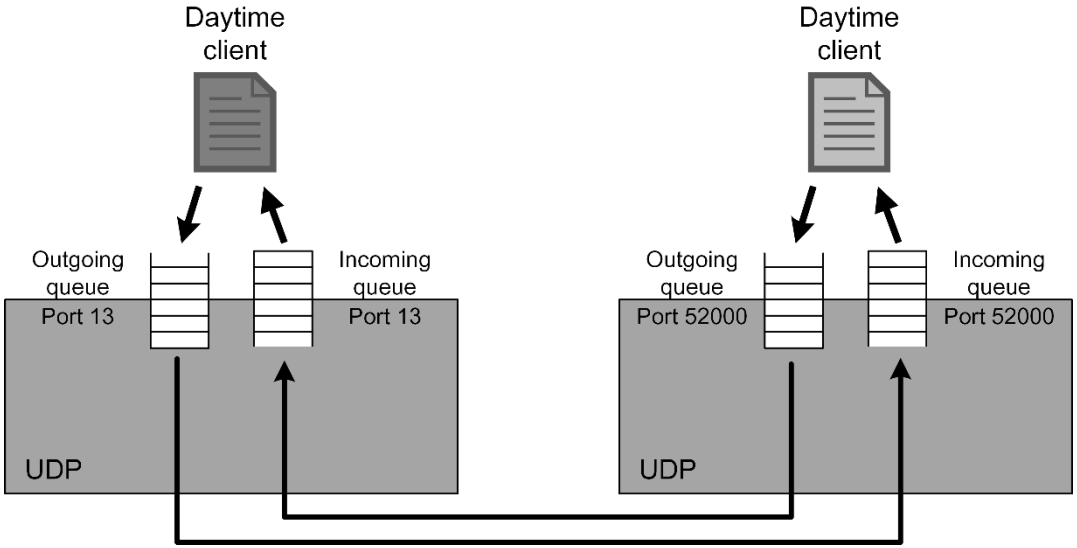
- UDP 서비스 (7/14)
  - 캡슐화와 역 캡슐화 (2/2)
    - 역캡슐화
      - 수신측 물리계층이 비트스트림을 데이터링크 계층으로 전달함
      - 데이터링크 계층은 헤더 검사 후 제거하여 IP 계층으로 전달함
      - IP 계층은 IP 헤더 검사 후 오류가 없으면 UDP 계층으로 전달함
      - UDP 계층은 헤더 검사 후 오류가 없으면 수신 프로세스에 메시지를 전달함



- UDP 서비스 (8/14)
  - 큐잉(Queuing) (1/3)
    - 개요
      - 여러 프로세스가 동시에 UDP를 사용하는 경우, 각 프로세스는 포트 번호로 구분 됨
      - 송수신 시 포트별 큐(Queue)를 이용해 메시지를 관리함
    - 특징
      - 포트 번호 단위로 운영됨
      - 수신 포트가 열려 있지 않거나 큐가 가득차면 메시지를 폐기함

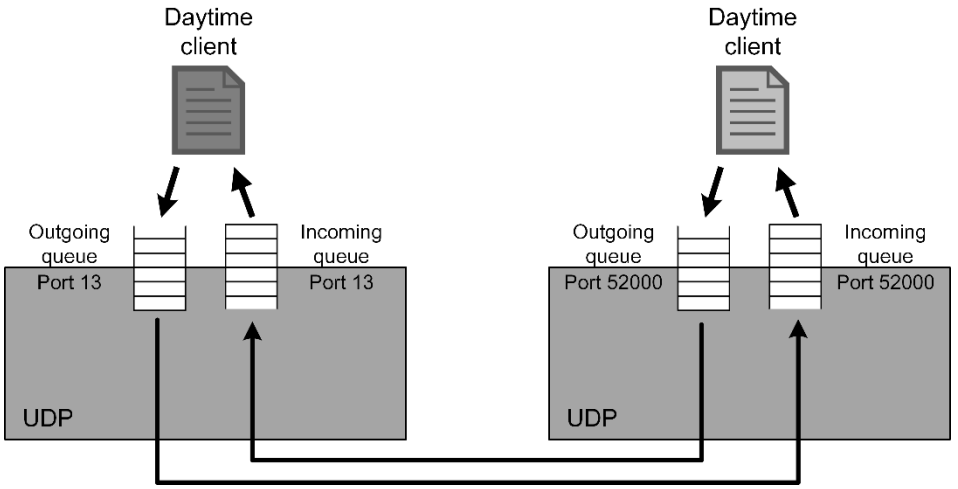
# UDP

- UDP 서비스 (9/14)
  - 큐잉(Queuing) (2/3)
    - 클라이언트측 동작
      - Outgoing Queue
        - 송신 메시지를 포트 번호에 맞게 큐에 저장 후 전송
      - Incoming Queue
        - 수신 메시지를 해당 포트 큐에 저장
        - 이후 프로세스가 가져감
      - 지정된 포트가 열려있지 않은 경우
        - ICMP Port Unreachable 메시지 전송



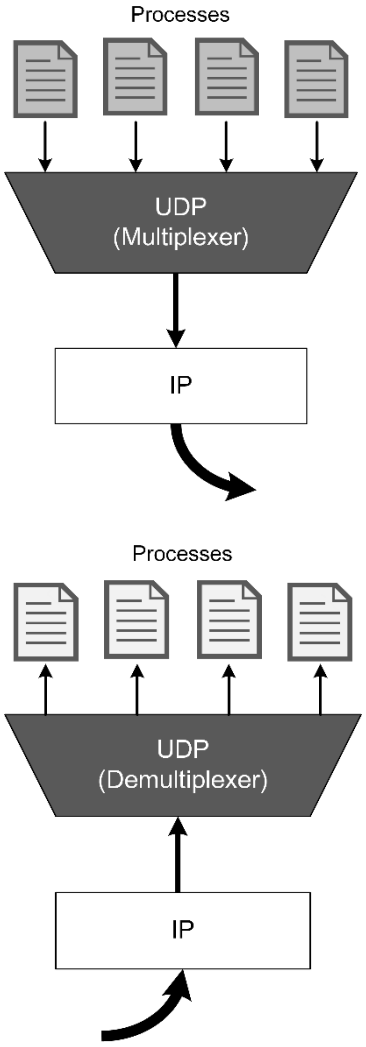
# UDP

- UDP 서비스 (10/14)
  - 큐잉(Queuing) (3/3)
    - 서버측 동작
      - 실행 시 특정 포트를 열고 해당 포트에 대한 입력 큐 (Incoming Queue) 생성
      - 메시지가 수신되면 포트 번호를 확인
        - 해당 포트의 입력 큐가 존재하면 해당 큐에 저장
        - 해당 포트의 입력 큐가 없으면 ICMP Port Unreachable 메시지 전송
      - 출력 큐에 많은 메시지가 쌓이면 오버플로우로 초과된 데이터그램 폐기
      - 응답 메시지를 보낼 때 서버 프로세스가 해당 포트의 출력 큐 (Outgoing Queue)를 통해 UDP 헤더를 추가해 전송



# UDP

- UDP 서비스 (11/14)
  - 다중화와 역다중화
    - 다중화
      - 송신측에서 UDP는 하나지만, 각 송신 프로세스마다 고유한 포트 번호를 사용하는 다-대-일 관계
      - 각 프로세스는 포트 번호로 구분
      - 각 프로세스의 메시지에 UDP 헤더를 추가하여 IP 계층에 전달
    - 역다중화
      - 수신측에서 UDP는 하나지만, 여러 프로세스가 동시에 UDP를 사용하는 일-대-다 관계
      - 수신한 데이터그램의 목적지 포트 번호를 확인
      - 해당 포트 번호에 매핑된 정확한 수신 프로세스로 메시지 분배



# UDP

- UDP 응용 (12/14)
  - UDP 특징
    - 비연결형 서비스
      - 연결 설정/해제 없음
      - 단순 패킷 교환 방식
    - 독립성
      - 각 데이터 그램은 서로 독립적으로 처리
    - 장점
      - 지연 감소
      - 오버헤드 작음
      - 단순 구조
      - 혼잡이 없어서 혼잡 제어 미제공
    - 단점
      - 신뢰성 부족
        - 순서 보장, 재전송, 오류제어 미제공

# UDP

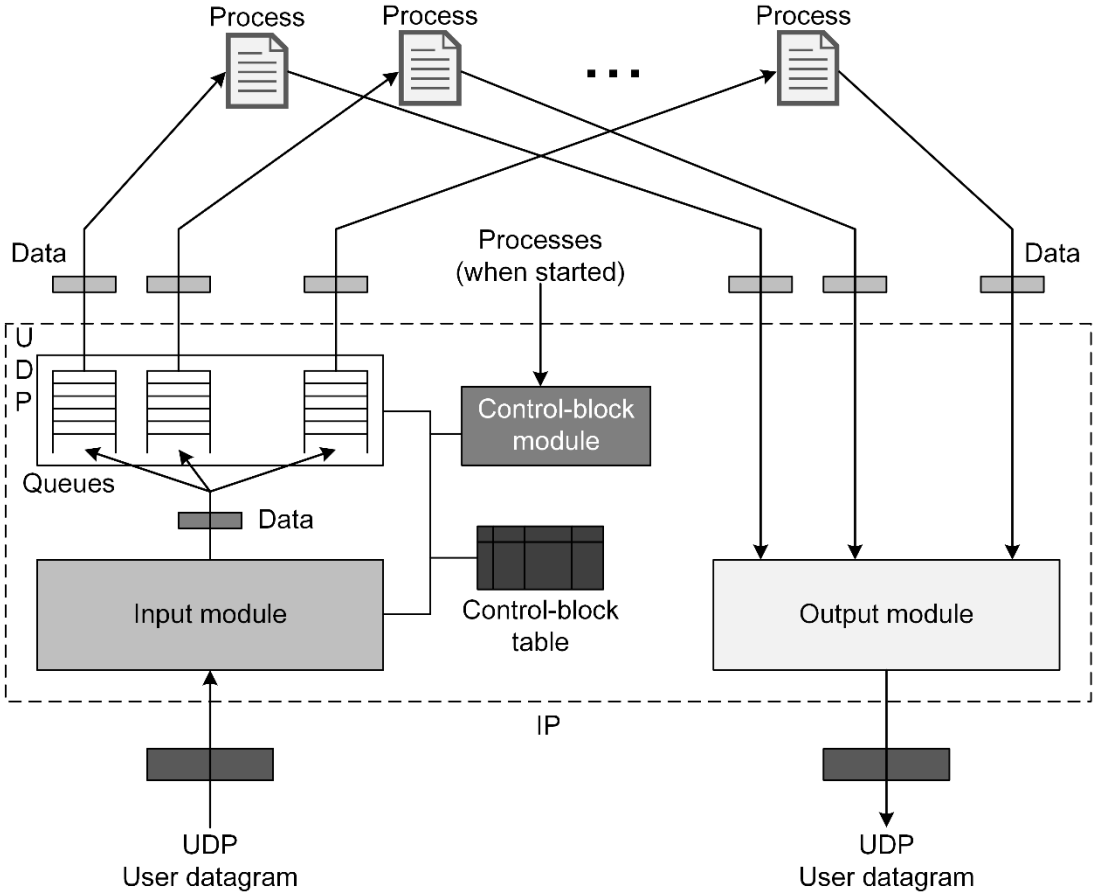
- UDP 응용 (13/14)
  - UDP 응용 사례 (1/2)
    - TFTP (Trivial File Transfer Protocol)
      - 간단한 파일 전송 프로토콜
      - UDP의 69번 포트 사용
      - 작은 크기의 파일 전송
      - 단순 요청/응답 구조
    - 멀티캐스팅(Multicasting)
      - 하나의 송신자가 여러 수신자에게 동시에 전송하는 방식
      - 연결 설정이 없어서 다수의 수신자에게 동시에 전송 가능

# UDP

- UDP 응용 (14/14)
  - UDP 응용 사례 (2/2)
    - SNMP (Simple Network Management Protocol)
      - 네트워크 장비 관리용 프로토콜
      - UDP의 161, 162번 포트 사용
      - 상태 확인, 오류보고, 설정관리에 사용
        - 빠른 전송이 필요하고 신뢰성 요구가 낮음
        - 관리 메시지의 크기가 작음
    - RIP (Routing Information Protocol)
      - 라우터 간 경로 정보를 교환하는 라우팅 프로토콜
      - UDP의 520번 포트 사용
      - 주기적으로 라우팅 테이블 업데이트에 사용
        - 30초마다 갱신하므로 신뢰성 요구가 낮음
        - 메시지 크기가 작음
        - 연결 유지없이 모든 라우터가 주기적으로 전달만 함

# UDP

- UDP 패키지
  - 정의
    - UDP가 UDP 패킷을 송신하고 수신하는 과정을 간략히 표현한 것
  - 구성요소 (1/6)
    - 입력 큐(Input Queues)
    - 제어 블록 테이블 (Control-Block Table)
    - 제어 블록 모듈 (Control-Block Module)
    - 입력 모듈
    - 출력 모듈



# UDP

- UDP 패키지 – 구성요소 (2/6)
  - 입력 큐
    - 각각의 프로세스 당 하나씩 배정되는 입력 큐의 집합을 사용
  - 제어 블록 테이블
    - 열린 포트를 추적하기 위한 테이블
    - 각 엔트리는 FREE이거나 IN-USE 두 값 중 하나를 가짐
    - 상태(state), 프로세스 ID, 포트 번호, 상응 큐 번호 등 최소 4개 이상의 필드를 가짐

# UDP

- UDP 패키지 – 구성요소 (3/6)
  - 제어 블록 모듈
    - 개요
      - 제어 블록 테이블의 관리를 담당함
      - 프로세스가 시작될 때 운영체제로부터 포트번호를 요청함
        - 프로세스는 서버에는 잘 알려진 포트번호 배당
        - 프로세스는 클라이언트에는 임시 포트 번호 배당
      - 프로세스는 프로세스 ID와 포트 번호를 제어 블록 모듈에 보내어 엔트리를 테이블에 생성함
      - 큐 번호의 필드는 0의 값을 가짐
    - 의사코드

```
UDP_Control_Block_Module (process ID, Port number)
{
      Search the table for a FREE entry
      if (not found)
            Delete one entry using a predefined strategy
      Create a new entry with the state IN − USE
      Enter the process ID and the port number
      Return
}   // End module
```

- UDP 패키지 – 구성요소 (4/6)
  - 입력 모듈 (1/2)
    - 개요
      - IP로부터 사용자 데이터그램을 받음
      - 제어 블록 테이블을 탐색하여 이 사용자 데이터그램과 같은 포트번호를 갖는 엔트리를 찾음
        - 해당 엔트리가 있다면 엔트리 내의 정보를 사용하여 데이터를 큐에 집어넣음
        - 해당 엔트리가 없다면 ICMP 메시지를 생성함

- UDP 패키지 – 구성요소 (5/6)
  - 입력 모듈 (2/2)
    - 의사코드

```
UDP_Input_Module (User Datagram)
{
      Look for the entry in the control_block table
      if (found)
      {
            Check to see if a queue is allocated
            If (queue is not allocated)
                  allocate a queue
            else
                  enqueue the data
      } //end if
       else
       {
              Ask ICMP to send an “unreachable port” message
              Discard the user datagram
       } //end else

      Return
}    // End module
```

- UDP 패키지 – 구성요소 (6/6)
  - 출력 모듈
    - 개요
      - 사용자 데이터그램을 생성하고 보냄
    - 의사코드

```
UDP_Input_Module (User Datagram)
{
      Create a user datagram
      Send the user datagram
      Return
}   // End module
```

# UDP

- UDP 패키지 (1/4)
  - 예제 14.8

처음 동작은 목적지 포트 번호가 52,012인 사용자 데이터그램이 도착하는 것이다. 입력 모듈은 이 포트 번호를 탐색하여 찾아낸다. 이 포트에는 큐 번호 38이 배당되어 있으므로 포트가 전에 이미 사용되어있음을 알 수 있다.
입력 모듈은 데이터를 38번 큐에 보낸다. 제어 블록 테이블은 변하지 않는다.

| State | Process ID | Port Number | Queue Number |
|---|---|---|---|
| IN-USE | 2,345 | 52,010 | 34 |
| IN-USE | 3,422 | 52,011 | |
| FREE | | | |
| IN-USE | 4,652 | 52,012 | 38 |
| FREE | | | |

# UDP

- UDP 패키지 (2/4)
  - 예제 14.9

몇 초 후 프로세스가 시작된다. 이 프로세스는 운영체제에 포트 번호를 요청하고 52,014를 할당받는다. 프로세스는 자신의 ID(4,978)와 포트 번호를 제어 블록 모듈에 보내어 테이블 내에 엔트리를 생성한다. 모듈은 첫 번째의 FREE 엔트리를 취하여 수신된 정보를 입력한다. 이 목적지를 향한 사용자 데이터그램이 도착하지 않았으므로 이 시점에서 모듈은 큐를 배당하지는 않는다.

| State | Process ID | Port Number | Queue Number |
|---|---|---|---|
| IN-USE | 2,345 | 52,010 | 34 |
| IN-USE | 3,422 | 52,011 | |
| FREE | | | |
| IN-USE | 4,652 | 52,012 | 38 |
| FREE | | | |



| State | Process ID | Port Number | Queue Number |
|---|---|---|---|
| IN-USE | 2,345 | 52,010 | 34 |
| IN-USE | 3,422 | 52,011 | |
| IN-USE | 4,978 | 52,014 | |
| IN-USE | 4,652 | 52,012 | 38 |
| FREE | | | |

# UDP

- UDP 패키지 (3/4)
  - 예제 14.10

52,011 포트에 사용자 데이터그램이 도착한다. 입력 모듈은 테이블을 검사하여 이 목적지를 위한 큐가 아직 배당되지 않았음을 알게 된다. 이것은 이 패킷이 이 목적지로 온 첫 번째 사용자 데이터그램이기 때문이다. 모듈은 43번 큐를 생성한다.

| State | Process ID | Port Number | Queue Number |
|---|---|---|---|
| IN-USE | 2,345 | 52,010 | 34 |
| IN-USE | 3,422 | 52,011 | |
| IN-USE | 4,978 | 52,014 | |
| IN-USE | 4,652 | 52,012 | 38 |
| FREE | | | |



| State | Process ID | Port Number | Queue Number |
|---|---|---|---|
| IN-USE | 2,345 | 52,010 | 34 |
| IN-USE | 3,422 | 52,011 | 43 |
| IN-USE | 4,978 | 52,014 | |
| IN-USE | 4,652 | 52,012 | 38 |
| FREE | | | |

# UDP

- UDP 패키지 (4/4)
  - 예제 14.11

몇 초 후 52,222 포트에 사용자 데이터그램이 도착한다. 입력 모듈이 테이블을 검사한 결과 이 목적지를 위한 엔트리를 발견하지 못한다. 사용자 데이터그램은 폐기되고 발신지에 “unreachable port” 메시지를 보낼 것을 ICMP에 요청한다.

| State | Process ID | Port Number | Queue Number |
|---|---|---|---|
| IN-USE | 2,345 | 52,010 | 34 |
| IN-USE | 3,422 | 52,011 | 43 |
| IN-USE | 4,978 | 52,014 | |
| IN-USE | 4,652 | 52,012 | 38 |
| FREE | | | |

# Thanks!

송 민 희(minhee@pel.sejong.ac.kr)