

# TCP/IP 프로토콜

-13장 전송 계층 개요, 14장 UDP-

서진우([jinwoo@pel.sejong.ac.kr](mailto:jinwoo@pel.sejong.ac.kr))

세종대학교 프로토콜공학연구실

# 목 차

---

- 전송 계층 개요
- UDP (User Datagram Protocol)

# 목 차

---

- 전송 계층 개요
- UDP (User Datagram Protocol)

# 전송 계층 개요

---

- 전송 계층(Transportation Layer)

- 정의

- OSI 4계층으로, 송신 프로세스와 수신 프로세스 간의 종단 간 데이터 전송을 담당하는 계층

- 역할

- 네트워크 계층에서 전달받은 데이터를 수신 프로세스에 전달
- 포트 번호를 이용하여 송수신 프로세스를 식별
- 데이터 분할 및 재조립을 통해 상위 계층에 데이터 전달

- 특징

- 종단 간 프로세스 통신 제공
- 연결형 및 비연결형 통신 지원
- 신뢰성 있는 데이터 전송 및 흐름 제어 등의 기능 제공

# 전송 계층 개요

- 전송 계층 서비스 (1/21)

\*프로세스(Process)

전송 계층의 서비스를 이용하는 응용 계층 객체(동작하는 프로그램).

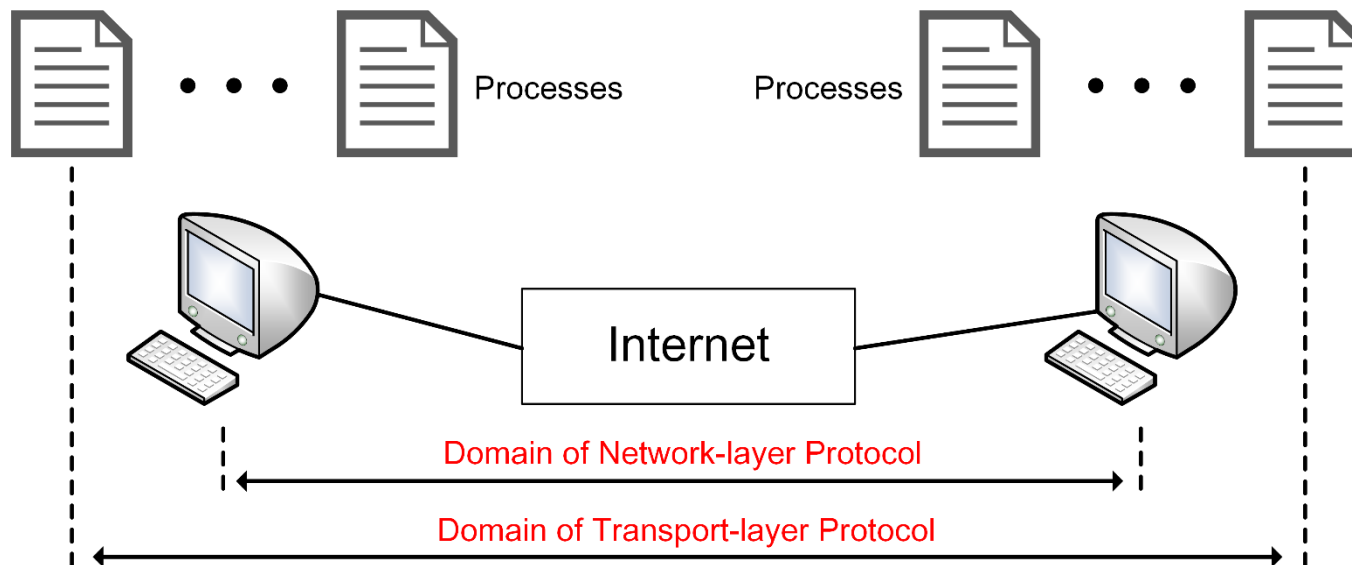
- 프로세스 대 프로세스 통신

- 네트워크 계층은 호스트 대 호스트 통신

- 네트워크 계층 프로토콜은 메시지를 단지 수신자 컴퓨터에 전달

- 메시지는 올바르게 프로세스에서 처리되어야 할 필요

- 전송 계층 프로토콜에서 수행

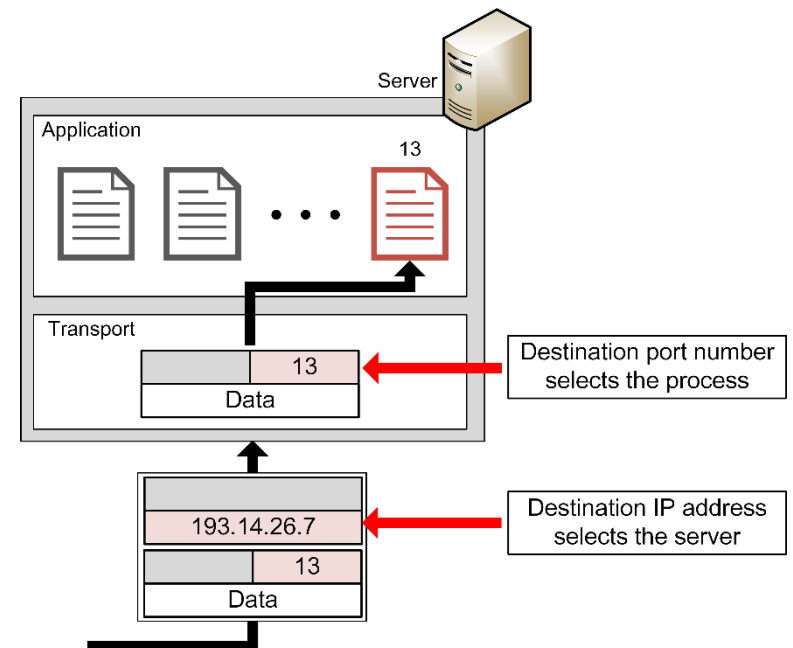
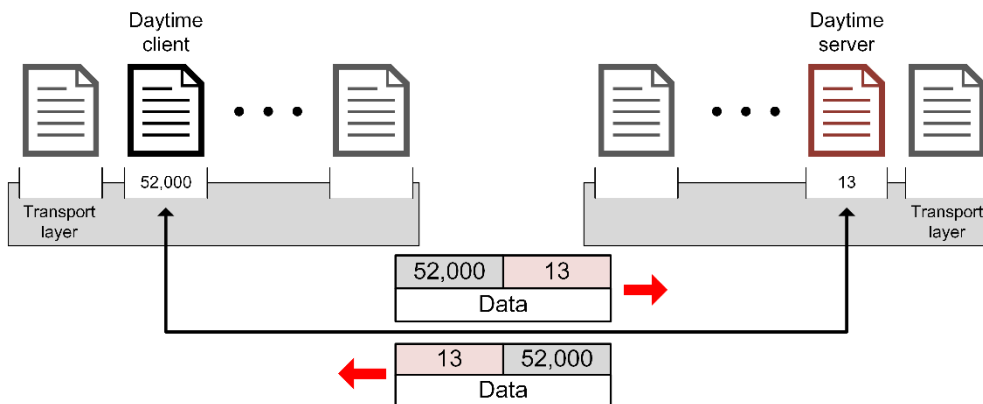


# 전송 계층 개요

- 전송 계층 서비스 (2/21)

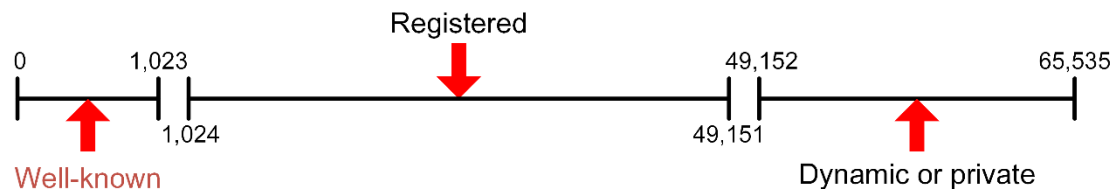
- 주소 체계: 포트 번호 (1/3)

- 프로세스를 식별하기 위한 주소 체계
- 호스트 내부에서 프로세스 식별을 위해 사용
  - IP 주소만으로는 호스트까지만 식별
- ICANN (Internet Corporation for Assigned Names and Numbers)에서 범위 관리
  - 3개의 구간으로 나누어 관리



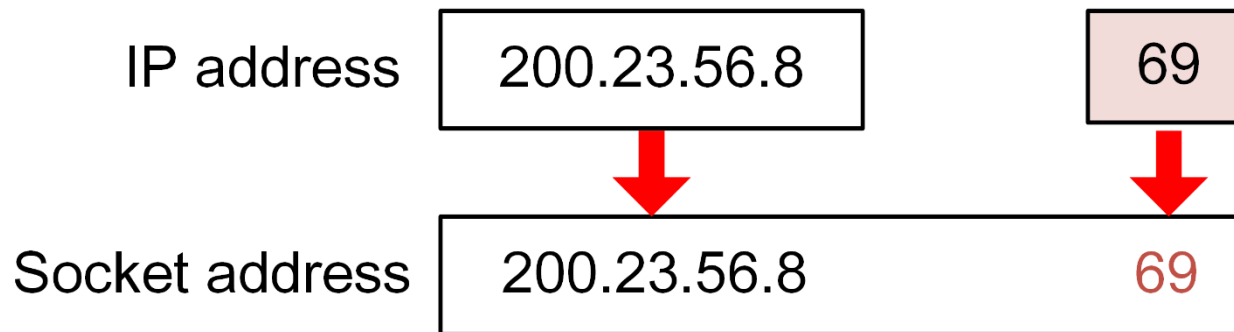
# 전송 계층 개요

- 전송 계층 서비스 (3/21)
- 주소 체계: 포트 번호 (2/3)
  - ICANN 범위
    - 잘 알려진 포트(Well-known Port)
      - 0~1,023 범위
      - ICANN에 의해 통제
    - 등록된 포트(Registered Port)
      - 1,024~49,151 범위
      - ICANN에 의해 할당이나 통제를 받지 않고, 중복 방지를 위해 ICANN에서 등록만 진행
    - 동적 포트(Dynamic Port)
      - 49,152~65,535 범위
      - ICANN에 의해 통제를 받지 않고 소켓 주소가 아님
      - 임시 또는 개인 포트 번호로 사용할 수 있는 범위



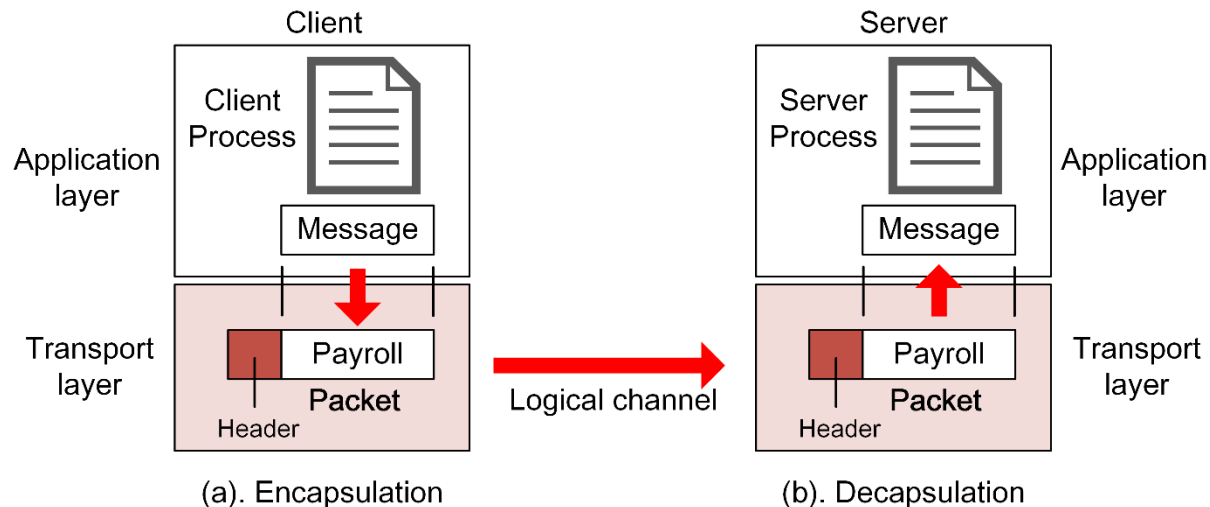
# 전송 계층 개요

- 전송 계층 서비스 (4/21)
- 주소 체계: 포트 번호 (3/3)
  - 소켓 주소(Socket Address)
    - IP 주소와 포트 번호의 조합
    - 클라이언트 소켓 주소는 클라이언트 프로세스를 유일하게 정의
    - 서버 소켓 주소는 서버 프로세스를 유일하게 정의
    - IP 주소는 네트워크 계층 패킷 헤더에, 포트 번호는 전송 계층 패킷 헤더에 저장되어 전달



# 전송 계층 개요

- 전송 계층 서비스 (5/21)
  - 캡슐화(Encapsulation)
    - 송신 측 전송 계층이 응용 계층의 메시지에 전송 계층 헤더를 추가하여 패킷을 생성
  - 역캡슐화(Decapsulation)
    - 수신 측 전송 계층이 패킷의 헤더를 제거하고 메시지를 응용 계층의 프로세스에 전달



# 전송 계층 개요

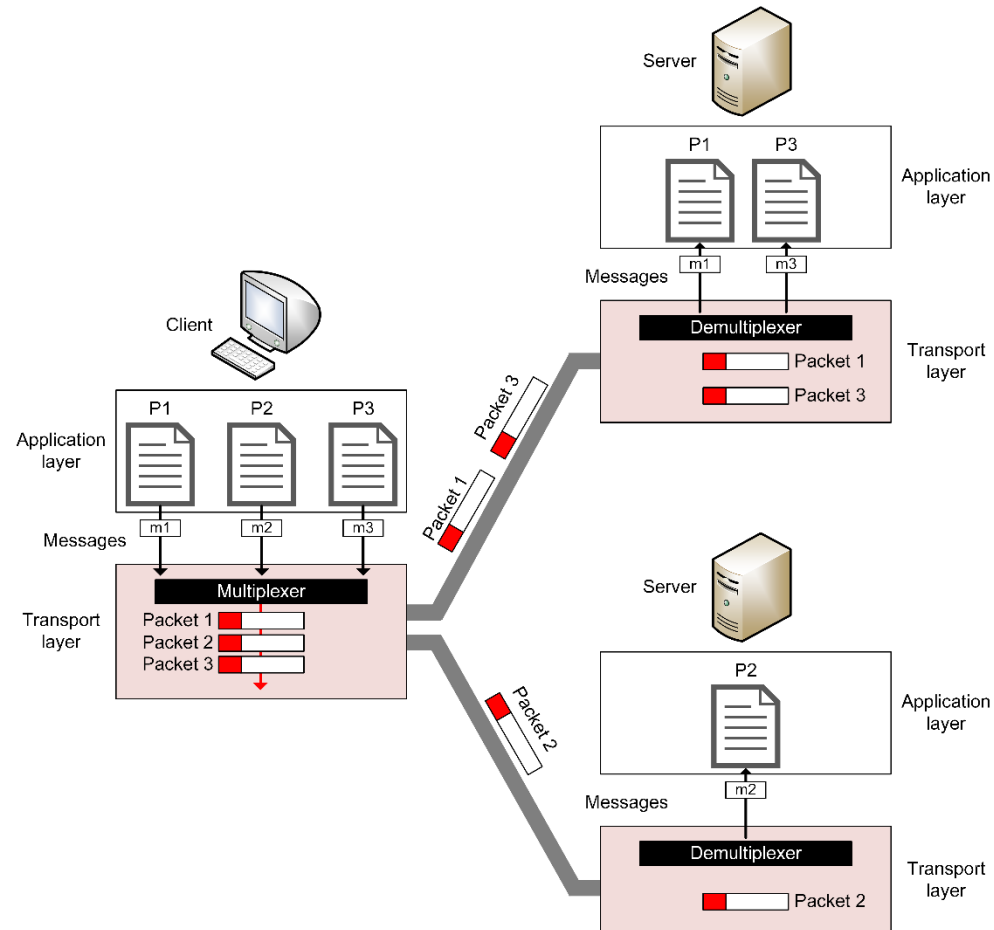
## • 전송 계층 서비스 (6/21)

### • 다중화(Multiplexing)

- 여러 응용 프로세스에서 전달받은 메시지를 전송 계층에서 처리하여 전송
- 여러 송신 프로세스의 데이터를 하나의 전송 계층이 수용

### • 역다중화(Demultiplexing)

- 수신한 패킷의 포트 번호 등을 이용하여 해당 프로세스에 데이터를 전달
- 하나의 전송 계층이 여러 수신 프로세스에 데이터를 분배



# 전송 계층 개요

---

- 전송 계층 서비스 (7/21)

\*버퍼(Buffer)  
패킷을 저장할 수 있는 메모리 영역.

- 흐름 제어 (1/3)

- 정의

- 수신 측의 처리 능력을 초과하는 데이터 전송 방지를 위해 송신 측과 수신 측 간의 처리 속도 차이를 조절하는 기능

- 방식

- 밀기(Pushing)와 끌기(Pulling)
    - 버퍼(Buffer)를 이용한 송수신 데이터를 임시 저장

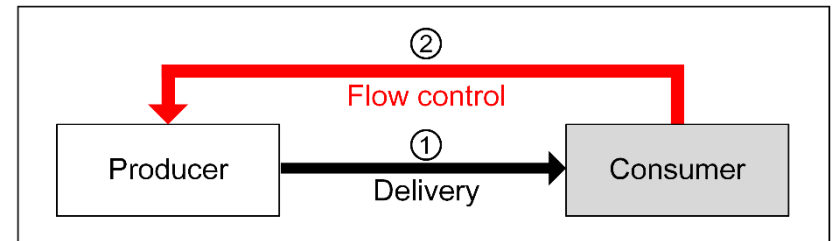
# 전송 계층 개요

- 전송 계층 서비스 (8/21)

- 흐름 제어 (2/3)

- 밀기(Pushing)

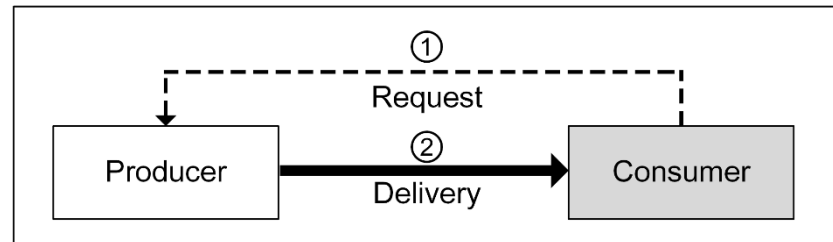
- 송신 측이 수신 측의 요청 없이 데이터 전송 후, 수신 측은 자신의 상태에 따라 전송 중단 및 재개를 요청하는 방식



(a). Pushing

- 끌기(Pulling)

- 수신 측이 요청할 경우에만 송신 측이 데이터를 전송하는 방식



(b). Pulling

- 버퍼(Buffer) 이용

- 수신 측의 버퍼가 가득 찰 경우, 수신 측에서 패킷의 전송을 멈추도록 송신 측에 패킷의 윈도우 크기 필드에 작성하여 알림
    - 수신 측의 버퍼에 빈 공간이 생길 경우, 수신 측에서 다시 전송하도록 송신 측 패킷의 윈도우 크기 필드에 작성하여 알림

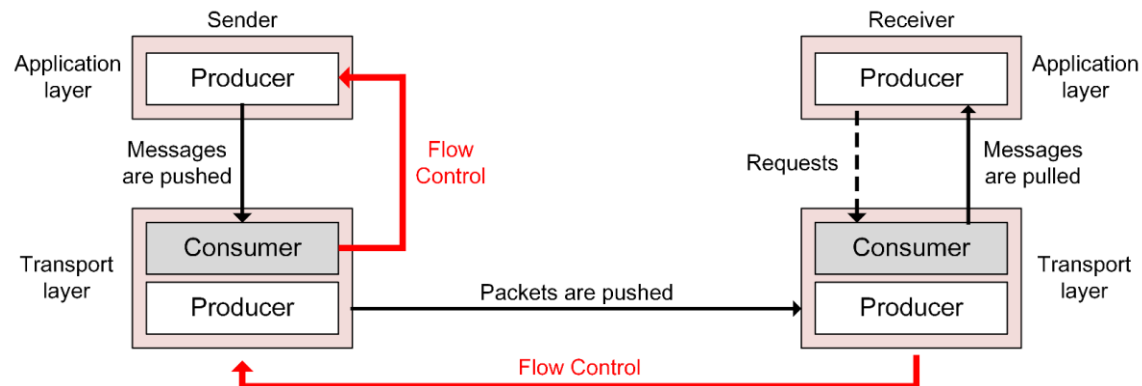
# 전송 계층 개요

- 전송 계층 서비스 (9/21)

- 흐름 제어 (3/3)

- 전송 계층에서의 흐름 제어

- 송신 전송 계층은 메시지를 패킷으로 캡슐화한 후 밀기 방식을 이용하여 수신 전송 계층의 버퍼로 전송
    - 수신 전송 계층은 수신한 패킷을 버퍼에 저장한 후 역캡슐화하여 응용 계층으로 전달
    - 수신 버퍼의 빈 공간에 따라 윈도우 크기를 조절하여 송신 측의 전송량을 제어
    - 전송 계층은 응용 계층 프로세스가 메시지를 요청하기 전까지 버퍼에 저장



# 전송 계층 개요

- 전송 계층 서비스 (10/21)

- 오류 제어 (1/2)

- 정의

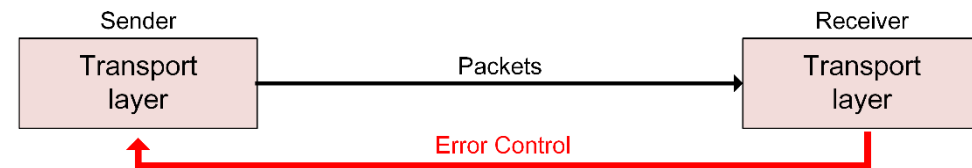
- 신뢰성 있는 데이터 전송을 위해 전송 중에 발생할 수 있는 오류 감지 및 복구하는 기능

- 기능

- 훼손된 패킷 감지 및 폐기
    - 손실되거나 폐기된 패킷을 송신 측에 재전송 요청
    - 중복 수신된 패킷은 확인 후 폐기
    - 순서가 역전되어 도착한 패킷은 수신 버퍼에 임시 저장하고, 누락된 패킷이 도착하면 순서에 맞게 재정렬

- 방식

- 순서 번호(Sequence Number)
    - 확인 응답(Acknowledgement)



# 전송 계층 개요

---

- 전송 계층 서비스 (11/21)
- 오류 제어 (2/2)
  - 순서 번호(Sequence Number)
    - 패킷의 재전송 여부, 중복 여부, 순서를 확인하는 용도
    - 패킷의 부여된 일련 번호
    - 순서 번호를  $m$  비트로 표현 시,  $\text{mod } 2^m$  값을 가짐
      - $0 \sim (2^m - 1)$ 의 범위를 가짐
  - 확인응답(Acknowledgement)
    - 수신 측에서 오류 없이 잘 수신된 패킷들에 대해 확인응답 전송 가능
      - 훼손되거나 중복 수신된 패킷을 폐기
      - 순서가 역전된 패킷은 폐기되거나 손실된 패킷이 도착하기 전까지 버퍼에 저장
    - 송신 측에서 타이머를 사용하는 경우, 패킷 손실 감지 가능
      - 패킷 전송 시 타이머 구동하고, 타이머가 만료되기 전까지 ACK가 도착하지 않으면 패킷 재전송

# 전송 계층 개요

---

- 전송 계층 서비스 (12/21)

- 흐름과 오류 제어의 결합 (1/3)

- 송신 측과 수신 측에 각각 번호를 매긴 버퍼를 통해 흐름 제어와 오류 제어를 결합 가능

- 송신 측 동작 과정

1. 패킷의 순서 번호로서 버퍼 내의 비어있는 영역의 제일 앞 부분의 번호인  $x$  사용
2. 패킷 전송 시, 패킷의 복사본은 메모리 영역  $x$ 에 저장
3. 상대방으로부터의 확인응답을 기다림
4. 확인응답 도착 시, 패킷은 제거되며 메모리 영역은 비워짐

- 수신 측 동작 과정

1. 순서 번호  $y$ 를 갖는 패킷 도착 시, 응용 계층에서 수신할 준비가 되기 전까지 수신된 패킷을 메모리 영역  $y$ 에 저장
2. 패킷  $y$ 의 도착을 알리기 위하여 확인응답 전송

# 전송 계층 개요

---

- 전송 계층 서비스 (13/21)
  - 흐름과 오류 제어의 결합 (2/3)
    - 슬라이딩 윈도우(Sliding Window)
      - 순서 번호를 원형으로 표현 시, 버퍼를 표현하는 일련의 조각 집합
      - 각 조각들은 언제나 원형의 일부분을 점유
  - 동작 방식
    - 송신 측에서 패킷 전송 시, 관련 조각 마크
      - 버퍼가 다 차서 더 이상의 메시지를 응용 계층으로부터 수신할 수 없다는 것을 의미
    - 확인응답 수신 시, 해당 조각의 마크 해제
    - 윈도우의 시작부터 몇 개의 연속적인 조각의 마크 해제 시, 윈도우의 끝에서 좀 더 많은 빈 조각을 허용하기 위해 윈도우를 관련 순서 범위를 미끄러지듯 돌림

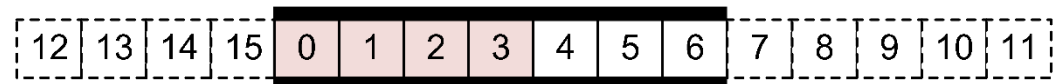
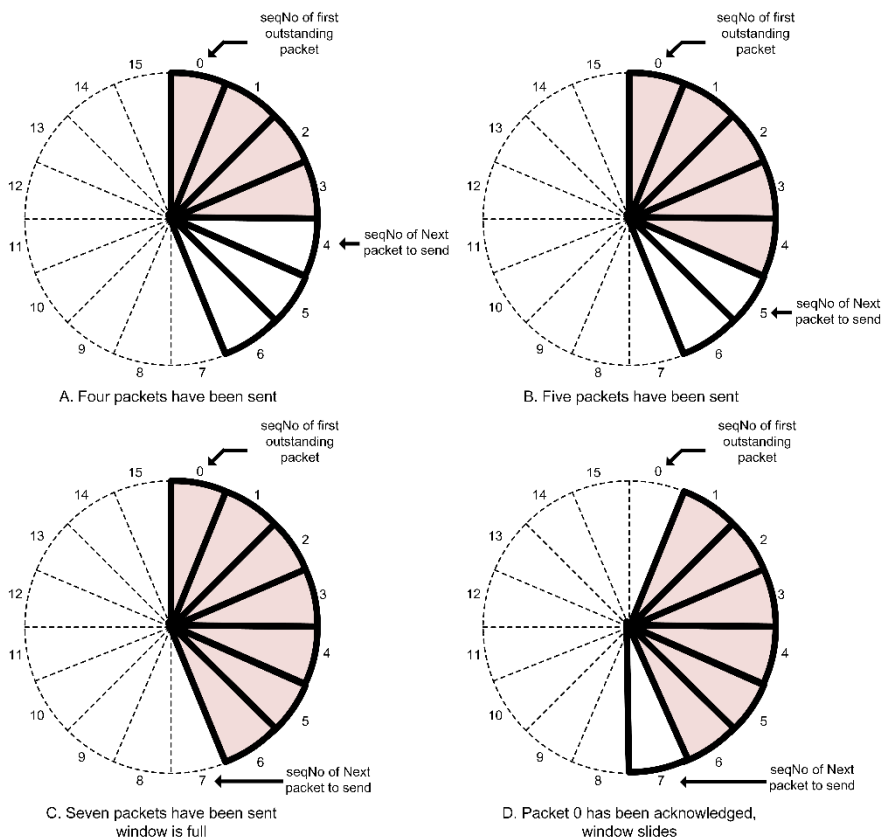
# 전송 계층 개요

- 전송 계층 서비스 (14/21)

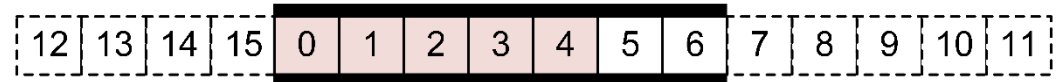
- 흐름과 오류 제어의 결합 (3/3)

- 슬라이딩 윈도우(Sliding Window)

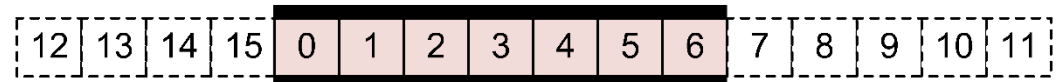
- 예시 - 순서 번호가  $\text{mod } 16$  ( $m = 4$ )이며 윈도우의 크기가 7인 경우



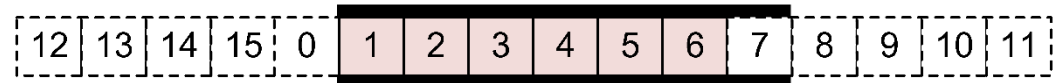
A. Four packets have been sent



B. Five packets have been sent



C. Seven packets have been sent  
window is full



D. Packet 0 has been acknowledged,  
window slides

# 전송 계층 개요

## • 전송 계층 서비스 (15/21)

\*로드(Load)  
네트워크로 전송되는 패킷의 양.

### • 혼잡 제어 (1/3)

#### • 정의

- 네트워크로 유입되는 로드(Load)가 네트워크가 처리할 수 있는 양을 초과하는 상황을 방지하거나 완화하는 기능

#### • 혼잡 발생 이유

- 라우터나 스위치가 패킷을 저장하기 위한 버퍼인 큐가 존재
  - 패킷은 적절한 출력 큐에 저장되며 자신의 순서가 되어 전송 전까지 기다림
  - 큐의 용량은 유한하며 라우터가 저장할 수 있는 것보다 더 많은 패킷이 라우터에 들어올 경우 발생

#### • 방식

- 개방 루프 혼잡 제어(Open-loop Congestion Control)
  - 혼잡 발생 이전에 정책을 적용하여 혼잡 예방
- 폐 루프 혼잡 제어(Closed-loop Congestion Control)
  - 혼잡 발생 이후 네트워크 상태를 확인하여 혼잡을 악화

# 전송 계층 개요

---

- 전송 계층 서비스 (16/21)
  - 혼잡 제어 (2/3)
    - 개방 루프 혼잡 제어(Open-loop Congestion Control)
      - 혼잡 발생 이전에 정책을 적용하여 혼잡 예방
    - 재전송 정책
      - 재전송은 혼잡을 가중시키기 때문에 적절한 재전송 정책 필요
      - 재전송으로 인한 트래픽 증가를 줄이기 위해 타이머와 확인 응답을 관리하며 불필요한 재전송을 억제
    - 윈도우 정책
      - 송신 윈도우 크기를 지나치게 크게 설정할 시 네트워크 혼잡이 발생
      - 윈도우 크기를 조절하여 네트워크 혼잡을 줄임
    - 확인응답 정책
      - 수신 측에서 수신한 패킷에 대한 확인 응답을 전송하지 않을 경우, 송신 측에서의 전송 속도가 느려져 혼잡 발생
      - 수신 측에서 전송할 패킷이 있거나 특정한 타이머가 만료되는 경우에만 확인응답을 전송하게 하여 네트워크의 로드를 줄임

# 전송 계층 개요

- 전송 계층 서비스 (17/21)

\*ECN(Explicit Congestion Notification)

IP 패킷과 TCP 헤더에서 네트워크 혼잡을 미리 알리고 제어하기 위해 사용하는 2개의 비트 필드.

- 혼잡 제어 (3/3)

- 폐 루프 혼잡 제어(Closed-loop Congestion Control)

- 혼잡 발생 이후 네트워크 상태를 피드백 받아 실시간으로 확인하고 전송량을 동적으로 조절

- 암묵적 피드백(Implicit Feedback)

- 네트워크로부터 직접적인 통보가 아닌 송신 측이 패킷 손실이나 왕복 시간의 지연을 보고 네트워크 혼잡을 스스로 감지

- 명시적 피드백(Explicit Feedback)

- 라우터나 수신 측이 혼잡 발생 신호를 송신 측에 직접 보내어 알림
      - e.g., IP 헤더의 ECN 비트를 사용하는 TCP ECN

- 수신 측 중심 피드백(Receiver-driven Feedback)

- 수신 측이 자신의 버퍼 상태나 수신 윈도우 크기 등을 고려하여 송신 측이 보낼 수 있는 데이터 양을 직접 지정

# 전송 계층 개요

## • 전송 계층 서비스 (18/21)

### • 비연결형과 연결 지향 서비스 (1/4)

#### • 비연결형(Connectionless) 서비스

##### • 정의

- 연결 설정 없이 메시지를 청크(Chunk) 단위로 분할하여 독립적인 패킷으로 전송하는 서비스

##### • 특징

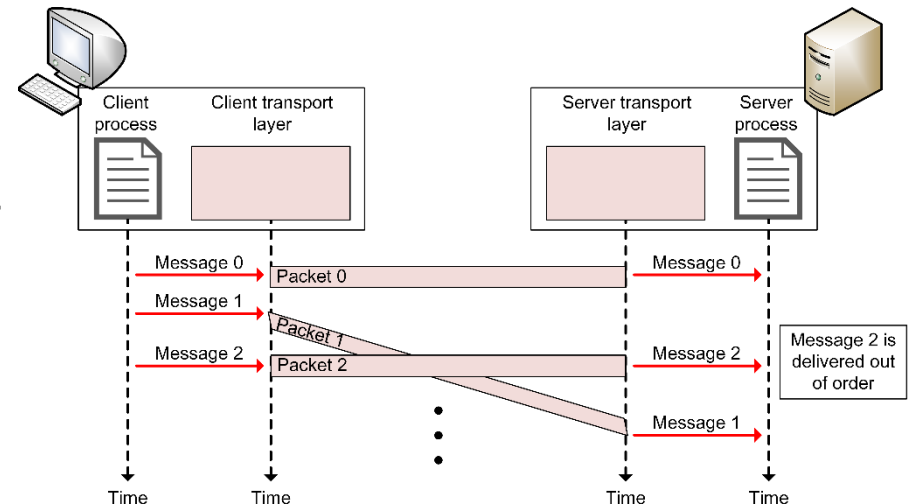
- 각 패킷은 서로 독립적으로 처리
- 패킷이 수신 측에 도착하는 순서가 달라질 수 있음
- 패킷 손실에 대한 확인 및 복구가 어려움

##### • 제어 기능

- 흐름 제어, 오류 제어, 혼잡 제어가 구현될 필요가 없음

\*청크(Chunk)

전송 계층에서 처리할 수 있는 크기로 분할된 메시지의 데이터 조각.



# 전송 계층 개요

- 전송 계층 서비스 (19/21)

- 비연결형과 연결 지향 서비스 (2/4)

- 연결 지향(Connection-oriented) 서비스

- 정의

- 데이터 전송 전에 연결을 설정하고, 데이터 전송 후 연결을 해제하는 서비스

- 과정

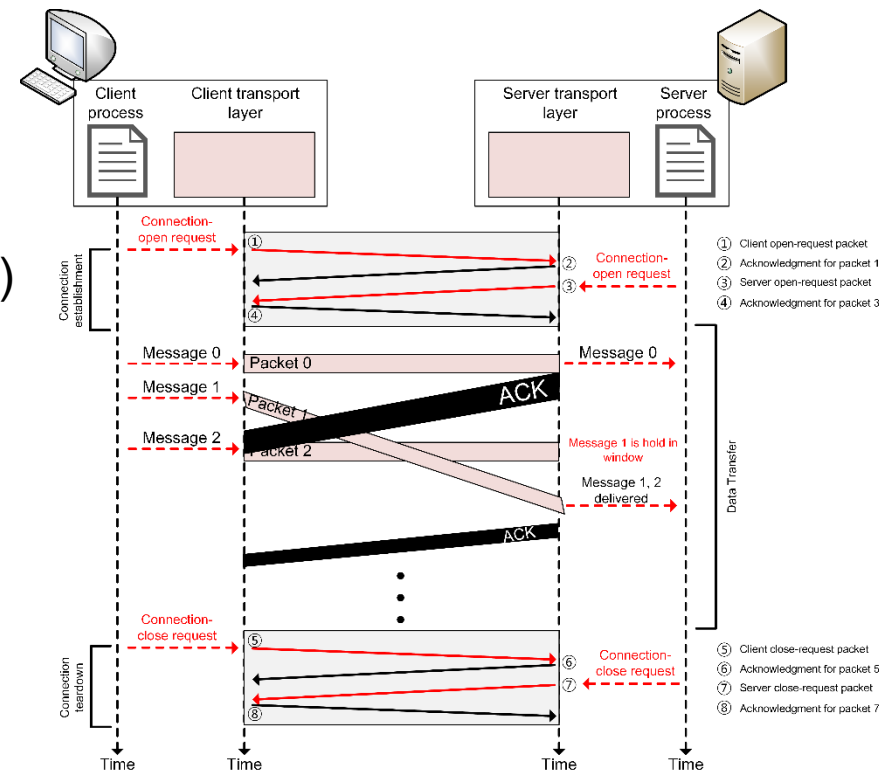
1. 연결 설정(Connection Establishment)
2. 데이터 전송(Data Transfer)
3. 연결 해제(Connection Teardown)

- 특징

- 연결지향 서비스는 두 종단 시스템 간의 연결을 기반으로 데이터 교환
      - 패킷 간 의존성이 존재

- 제어 기능

- 흐름 제어, 오류 제어, 혼잡 제어를 구현 가능



# 전송 계층 개요

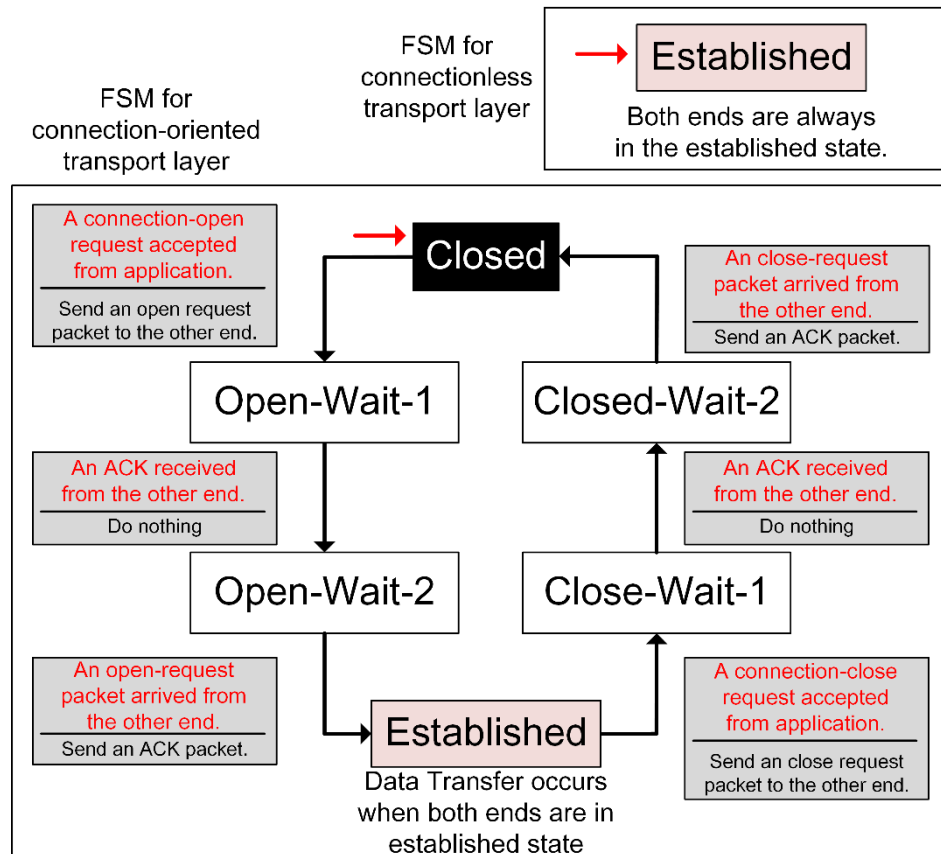
---

- 전송 계층 서비스 (20/21)
- 비연결형과 연결 지향 서비스 (3/4)
  - 유한 상태 기기(FSM, Finite State Machine)
    - 정의
      - 정해진 개수의 상태와 상태 간 전이 규칙을 이용하여 이벤트에 따른 동작을 표현하는 수학적 모델
    - 구성
      - 상태(State)
        - 현재 시스템이 어떤 상황에 있는 지를 나타냄
      - 이벤트(Event)
        - 상태 전이를 발생시키는 입력 또는 조건
      - 액션(Action)
        - 이벤트가 발생했을 때 수행하는 동작
      - 상태 전이(Transition)
        - 이벤트와 액션에 따라 현재 상태에서 다른 상태로 이동하는 과정
      - 초기 상태(Initial State)
        - FSM이 동작을 시작하는 상태

# 전송 계층 개요

- 전송 계층 서비스 (21/21)
- 비연결형과 연결 지향 서비스 (4/4)
  - FSM으로 표현된 비연결형과 연결 지향 서비스

Note: The colored arrow shows the starting state.



# 전송 계층 개요

- 전송 계층 프로토콜 (1/34)

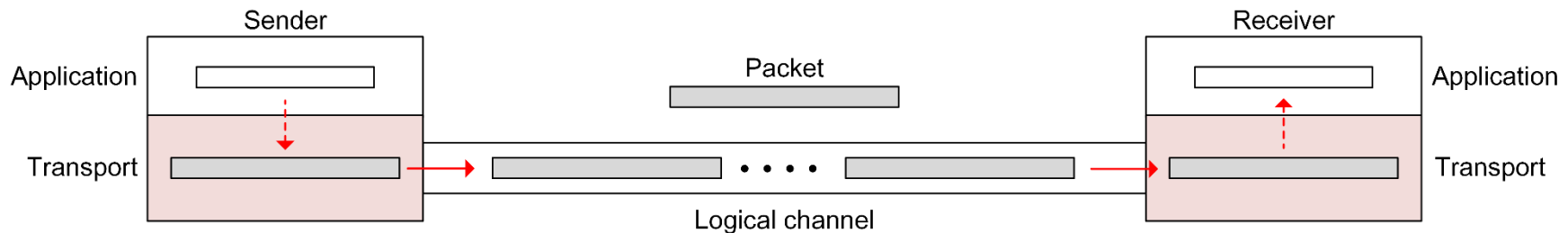
- 단순(Simple) 프로토콜 (1/2)

- 개요

- 흐름/오류 제어가 없는 비연결형 프로토콜

- 동작

1. 자신의 응용 계층으로부터 전송 계층은 메시지를 수신
2. 전송 계층은 메시지를 패킷으로 만든 후 패킷 전송
3. 수신 측 전송 계층은 자신의 네트워크 계층을 통하여 들어온 패킷 수신
4. 수신 측 전송 계층은 패킷으로부터 메시지 추출 후, 자신의 응용 계층으로 전달

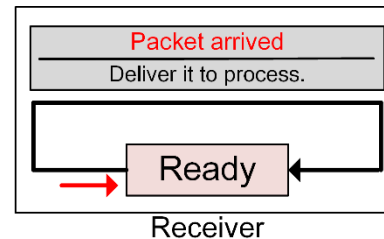
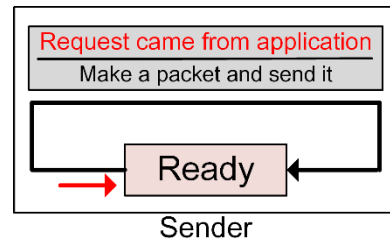


# 전송 계층 개요

- 전송 계층 프로토콜 (2/34)

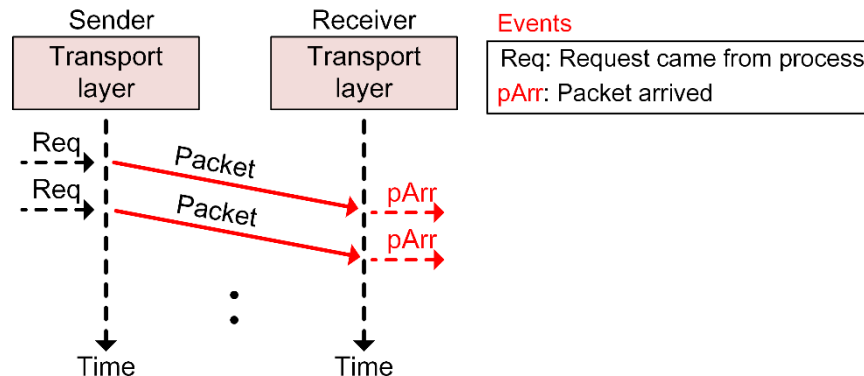
- 단순(Simple) 프로토콜 (2/2)

- FSM



- 예시

- 송신 측이 수신 측을 고려하지 않은 경우



- 한계점

- 오류 제어와 흐름 제어가 없어 패킷 손실 및 수신 처리 능력을 고려하지 못함

# 전송 계층 개요

- 전송 계층 프로토콜 (3/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (1/11)

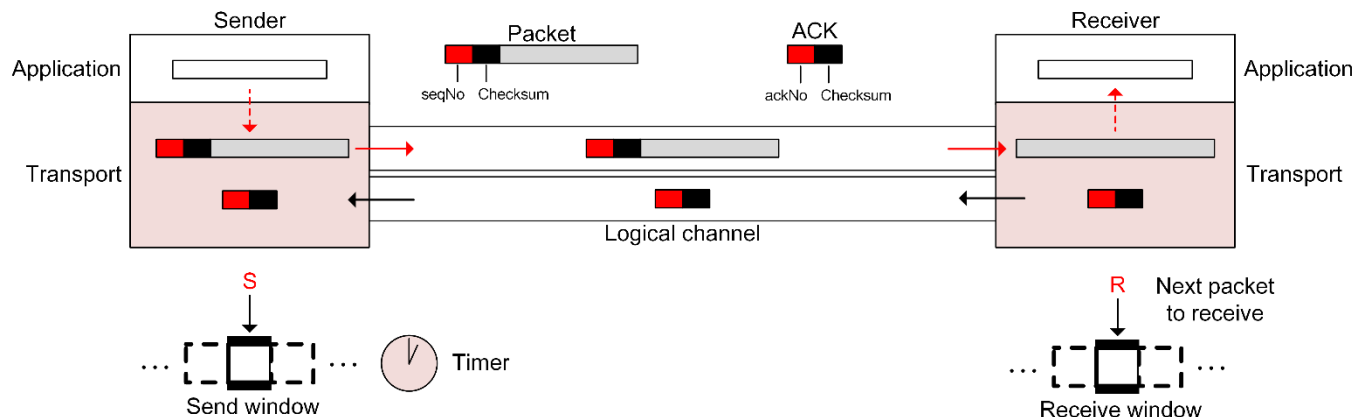
- 흐름/오류 제어를 모두 사용하는 연결 지향 프로토콜

- 동작 (1/2)

- 송/수신 측 모두 크기가 1인 슬라이딩 윈도우 사용
- 한 번에 하나의 패킷과 하나의 확인응답 전송 가능

1. 송신 측은 한 번에 하나의 패킷 전송 후, 확인응답이 들어오기 전까지 다음 패킷 전송하지 않고 기다림

- 패킷이 훼손되었는지 검사하기 위해 각각의 데이터 패킷에 검사합 추가



# 전송 계층 개요

- 전송 계층 프로토콜 (4/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (2/11)

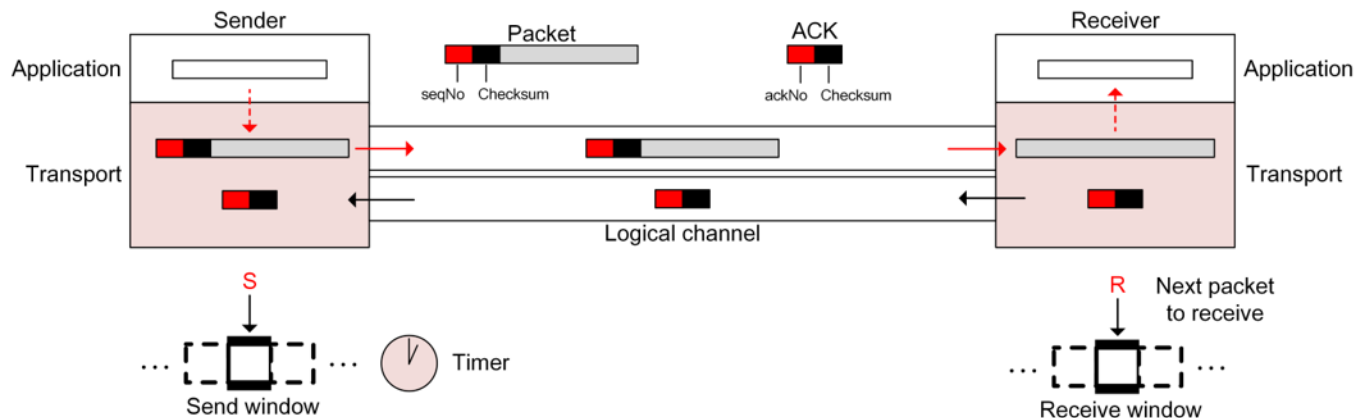
- 동작 (2/2)

- 2. 수신 측은 패킷을 검사합을 통해 패킷 검사

- 검사합이 틀릴 경우, 훼손되었다고 판단 후 폐기

- 3. 송신 측은 패킷을 전송할 때마다 타이머 구동

- 타이머가 만료되기 전에 확인응답 도착 시, 타이머는 정지되며 다음 패킷 전송
      - 타이머 만료 시, 패킷이 손실 혹은 훼손되었다고 판단 후 패킷 재전송
      - 이를 위해, 송신 측은 패킷의 사본을 가지고 있어야 함



# 전송 계층 개요

---

- 전송 계층 프로토콜 (5/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (3/11)

- 순서 번호(seqNo, Sequence Number)

- 패킷을 중복 수신하지 않도록 하기 위해 사용
    - 패킷 헤더에 패킷의 순서 번호 필드 추가

- *modulo 2* 연산을 통해 1비트의 필드만 사용

- 순서 범위 고려 필요

- $x$ 를 순서 번호로 사용 시,  $x + 1$ 만 필요하며  $x + 2$ 는 필요없음
    - 패킷이 정상적으로 수신 측에 도착 시, 수신 측은 확인응답 전송
      - 확인응답이 송신 측에 도착 시, 송신 측은  $x + 1$ 의 순서 번호를 갖는 패킷 전송
    - 패킷이 정상적으로 수신 측에 도착하였지만 확인응답이 훼손/손실 될 시, 송신 측은 타임-아웃 후 패킷 재전송
      - 이럴 경우, 재전송된 패킷은 중복 패킷
      - 수신 측은  $x + 1$  순서 번호를 갖는 패킷의 수신을 기대하였지만,  $x$  순서 번호를 갖는 패킷 수신
    - 패킷이 훼손되어 수신 측에 도착하지 않을 시, 송신 측은 타임-아웃 후 패킷 재전송
      - 재전송된 패킷을 수신한 수신 측은 확인응답 전송

# 전송 계층 개요

- 전송 계층 프로토콜 (6/34)

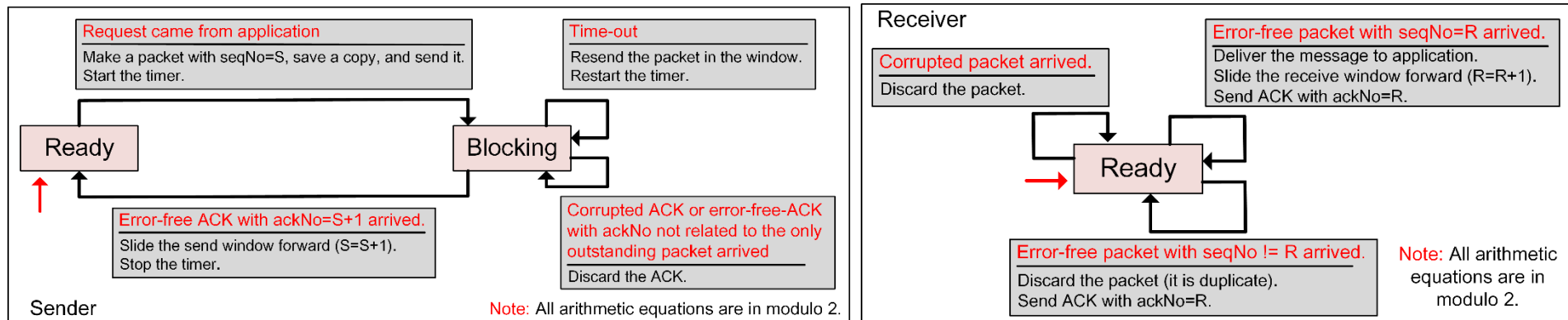
- 정지 대기(Stop-and-Wait) 프로토콜 (4/11)

- 확인응답(ackNo, Acknowledgement Number)

- 패킷을 중복 수신하지 않도록 하기 위하여 사용
    - 수신 측에서 받기를 기대하는 다음 패킷의 순서 번호를 나타냄
      - seqNo가  $x$ 라면, ackNo는  $x + 1$ 의 값을 가짐

- FSM (1/4)

- 연결 지향 프로토콜이므로, 양 끝단은 데이터 패킷 교환 전 Established 상태



# 전송 계층 개요

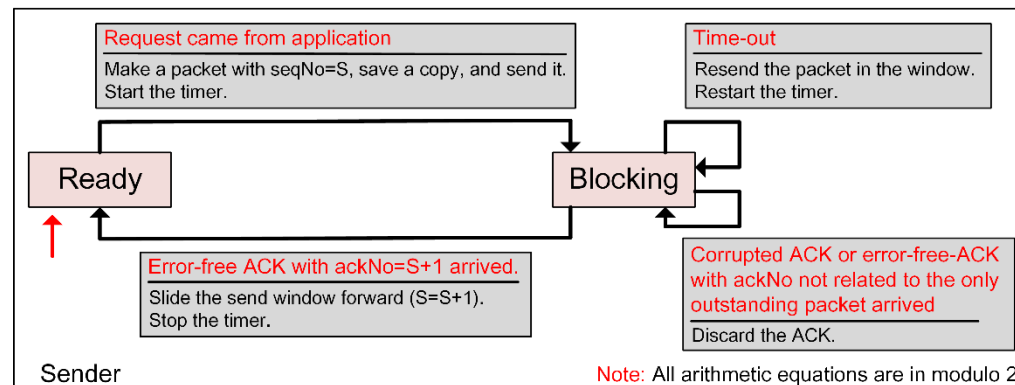
- 전송 계층 프로토콜 (7/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (5/11)

- FSM (2/4)

- 송신 측

- 초기에 Ready 상태에 있으며, 이후 Ready 상태와 Blocking 상태로 이동 가능
      - 변수  $S$ 는 0으로 초기화
      - Ready 상태
        - 송신 측은 단지 하나의 이벤트가 발생하기만을 기다림
        - 응용 계층으로부터 요청이 들어올 경우, 송신 측은 순서 번호를  $S$ 로 설정한 패킷을 만들고 패킷의 사본을 저장한 후에 패킷 전송
        - 이 후 송신 측은 타이머 구동 후 Blocking 상태로 이동



# 전송 계층 개요

- 전송 계층 프로토콜 (8/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (6/11)

- FSM (3/4)

- 송신 측

- Blocking 상태

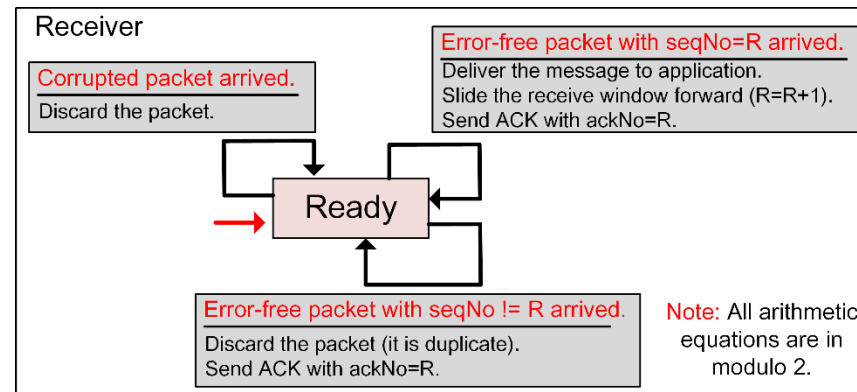
- 세 가지 경우가 발생 가능

- 1. 다음에 전송할 패킷에 관련된 확인응답 번호를 가지는 오류 없는 ACK가 도착하면 타이머 중단

- 윈도우는  $S = (S + 1) \text{ modulo } 2$ 로 이동 후 송신 측은 Ready 상태로 이동

- 2. 수신된 ACK가 훼손됐거나 오류는 없지만 확인응답 번호가 올바르지 않을 경우 ACK 폐기

- 3. 타임-아웃 발생 시, 송신 측은 패킷을 재전송하고 타이머 재구동



# 전송 계층 개요

- 전송 계층 프로토콜 (9/34)

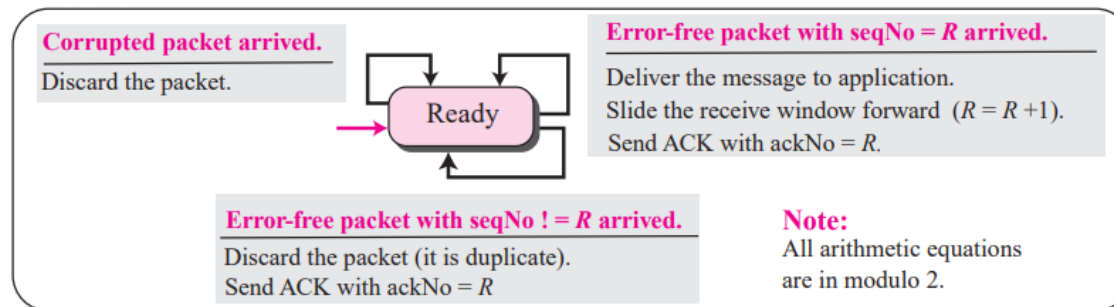
- 정지 대기(Stop-and-Wait) 프로토콜 (7/11)

- FSM (4/4)

- 수신 측

- 수신 측은 항상 Ready 상태에 있으며 변수  $R$ 는 0으로 초기화
      - 다음 세 가지 경우 발생 가능
        1. 순서 번호가  $R$ 로 설정된 오류 없는 패킷 도착 시, 패킷 내의 메시지는 응용 계층으로 전달
          - 윈도우는  $R = (R + 1) \text{ modulo } 2$ 로 이동 후 확인응답 번호  $R$ 로 설정된 ACK 전송
        2. 순서 번호가  $R$ 이 아닌 오류 없는 패킷이 도착 시, 패킷 폐기 후 확인응답 번호  $R$ 로 설정된 ACK 전송
        3. 훼손되어 수신된 패킷은 폐기

Receiver



# 전송 계층 개요

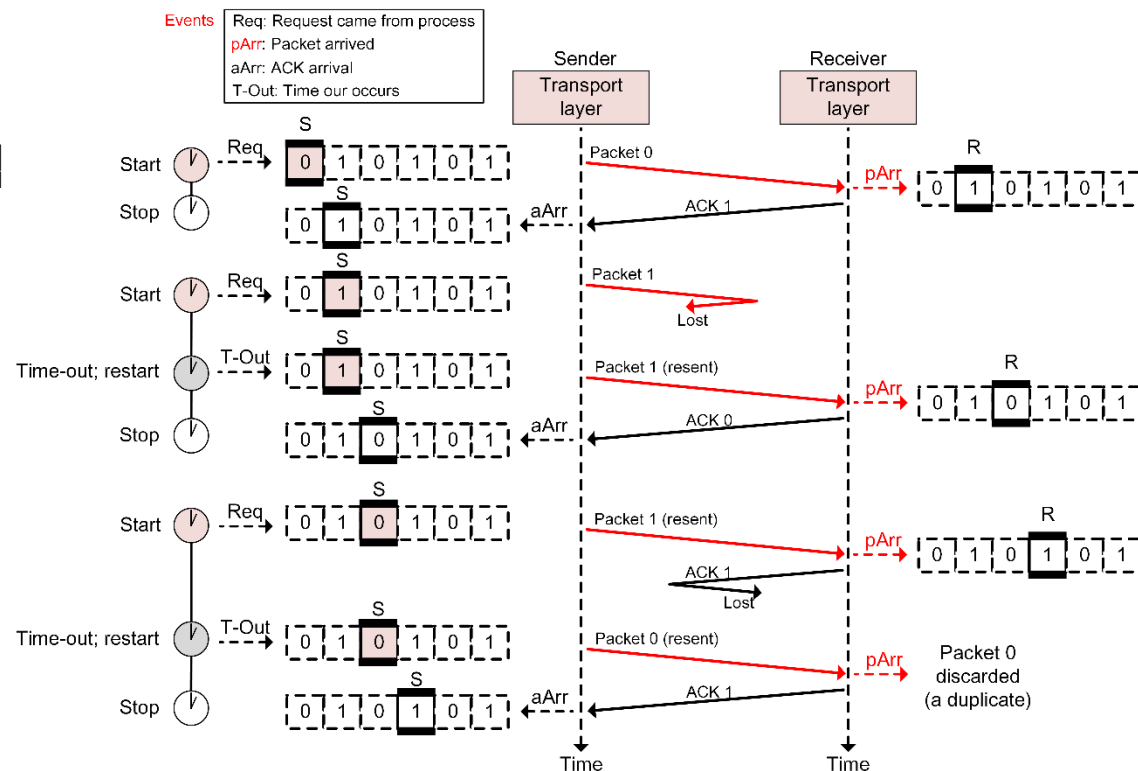
- 전송 계층 프로토콜 (10/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (8/11)

- 예시

- 타이머와 확인응답 패킷에 의해 재전송이 이루어지는 경우

- 패킷 0이 전송되고 확인응답 되어짐
- 패킷 1은 손실되고 타임-아웃 후에 재전송됨
- 재전송된 패킷 1은 확인응답되고 타이머는 중단
- 패킷 0은 전송되고 확인응답되었으나 ACK 손실
- 송신 측은 패킷 또는 ACK가 손실되었는지를 알 수 없기 때문에 타임-아웃 후에 패킷 0을 재전송하였고, 해당 패킷은 확인응답됨



# 전송 계층 개요

- 전송 계층 프로토콜 (11/34)

- 정지 대기(Stop-and-Wait) 프로토콜 (9/11)

- 효율

- 채널이 큰 대역폭과 긴 왕복 지연을 가지고 있을 경우 비효율적
    - 대역폭 지연 곱(Product)을 최대한으로 사용해야 함
      - 대역폭과 왕복 지연의 곱으로, 수신 측으로부터의 확인응답을 기다리면서 송신 측에서 채널을 통해 전송할 수 있는 비트의 수를 의미
    - 예제 13.1

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 이용률은 얼마인가?

- 풀이

- 대역폭 지연 곱:  $(1 \times 10^6) \times (20 \times 10^{-3}) = 20,000$ 비트
      - 시스템은 왕복 지연 동안 20,000비트를 전송 가능을 의미
    - 데이터 패킷의 길이에 따라 1,000비트만 전송 가능
    - 선로 이용률:  $\frac{1,000}{20,000} \times 100 = 5\%$ 
      - 해당 선로에서 정지 대기 프로토콜을 사용하는 것은 용량 낭비

# 전송 계층 개요

- 전송 계층 프로토콜 (12/34)
  - 정지 대기(Stop-and-Wait) 프로토콜 (10/11)
    - 효율
    - 예제 13.2

예제 13.1에서 확인응답에 대한 고려 없이 최대 15개의 패킷을 전송할 수 있는 프로토콜을 이용하면 선로의 이용률은 얼마인가?

- 풀이
  - 대역폭 지연 곱:  $(1 \times 10^6) \times (20 \times 10^{-3}) = 20,000$ 비트
    - 시스템은 왕복 지연 동안 20,000비트를 전송 가능을 의미
  - 데이터 패킷의 길이에 따라  $15 \times 1,000 = 15,000$ 비트 전송 가능
  - 선로 이용률:  $\frac{15,000}{20,000} \times 100 = 75\%$

# 전송 계층 개요

---

- 전송 계층 프로토콜 (13/34)
  - 정지 대기(Stop-and-Wait) 프로토콜 (11/11)
    - 파이프라인
      - 이전 업무가 완료되기 전에 새로운 업무를 시작하는 기법
  - 정지 대기 프로토콜에서의 파이프라인
    - 파이프라인 기능 미제공
      - 송신 측에서 패킷이 목적지에 도착하고 확인응답되기 전에는 다음 패킷 전송 불가능
    - N-프레임-후퇴(GBN, Go-Back-N) 프로토콜, 선택적 반복(Selective Repeat) 프로토콜에서는 파이프라인 제공
- 한계점
  - 패킷마다 응답을 기다려야 하므로 전송 효율이 낮음

# 전송 계층 개요

---

- 전송 계층 프로토콜 (14/34)
  - N-프레임-후퇴(Go-Back-N) 프로토콜 (1/12)
    - 개요
      - 슬라이딩 윈도우를 기반으로 오류 제어를 수행하는 연결 지향 프로토콜
  - 특징
    - 송신 측은 확인응답을 수신하기 전에 여러 개의 패킷을 전송 가능
    - 수신 측은 하나의 패킷만 버퍼에 저장 가능
    - 송신 측은 확인응답 도착 전에 전송된 패킷들에 대한 사본 소유
    - 오류 발생 시, 해당 패킷 이후 모든 패킷을 재전송
    - 여러 개의 데이터 패킷과 확인응답이 채널에 동시에 존재 가능

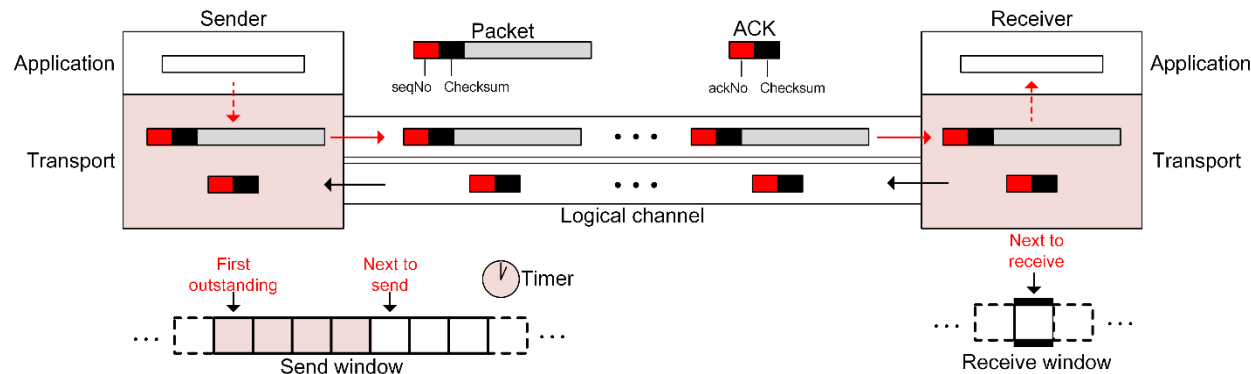
# 전송 계층 개요

- 전송 계층 프로토콜 (15/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (2/12)

- 동작

1. 송신자는 윈도우 크기(N) 내에서 연속적으로 패킷을 전송
2. 수신자는 오류 없이 수신한 마지막 패킷 번호를 기준으로 누적 ACK를 전송
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 재전송



# 전송 계층 개요

---

- 전송 계층 프로토콜 (16/34)
  - N-프레임-후퇴(Go-Back-N) 프로토콜 (3/12)
    - 순서 번호
      - 순서 번호는  $\text{modulo } 2^m$  이며,  $m$ 은 순서 번호 필드의 비트 수
  - 확인응답 번호
    - 누적 값이며, 수신되기를 기대하는 다음 패킷의 순서 번호를 의미
  - 예시 – 확인응답 번호가 7일 경우
    - 처음부터 6번까지의 순서 번호를 갖는 패킷이 모두 안전하게 도착
    - 수신 측은 순서 번호 7을 갖는 패킷의 수신을 기다리고 있다는 것을 의미

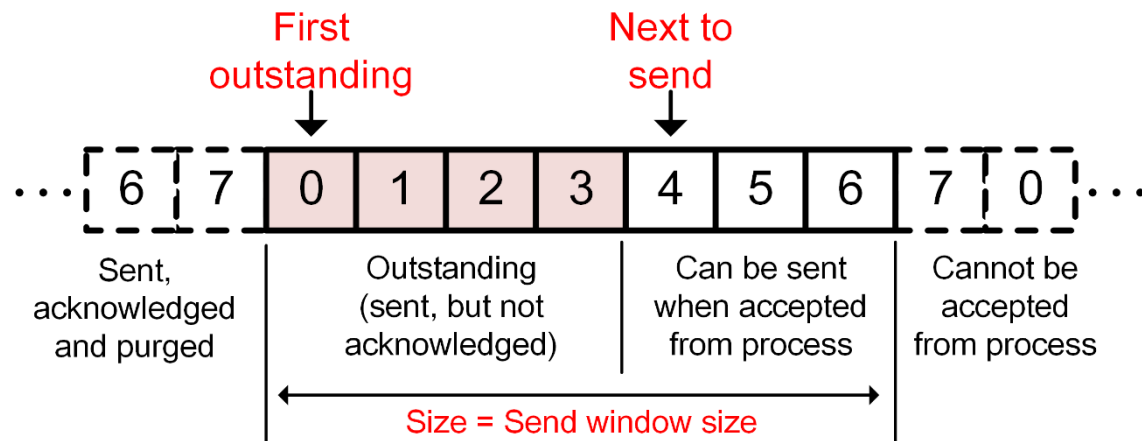
# 전송 계층 개요

- 전송 계층 프로토콜 (17/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (4/12)

- 송신 윈도우

- 전송 중이거나 전송될 데이터 패킷의 순서 번호를 포함하는 가상의 상자를 의미
      - 각 윈도우의 위치에서 어떤 순서 번호들은 이미 전송된 패킷을 나타내고, 또 다른 순서 번호들은 전송되고자 하는 패킷을 의미
    - 윈도우의 최대 크기는  $2^m - 1$ 로 고정



# 전송 계층 개요

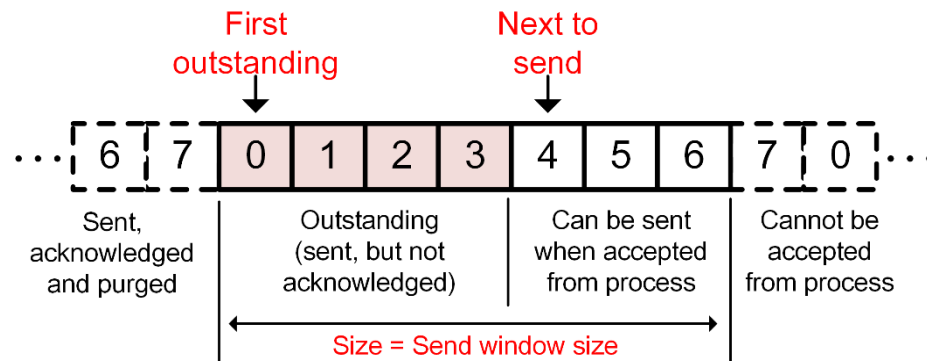
- 전송 계층 프로토콜 (18/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (5/12)

- 송신 윈도우

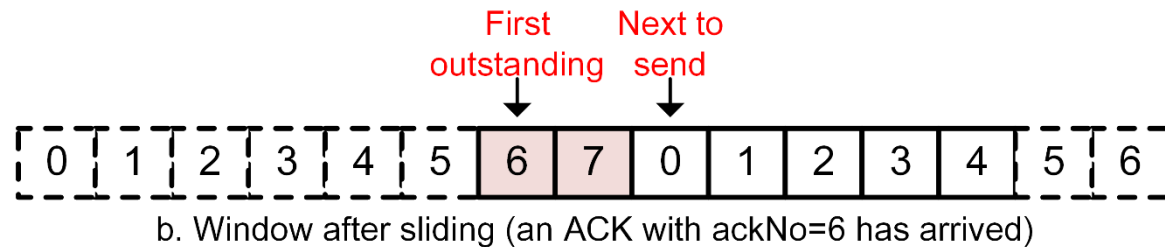
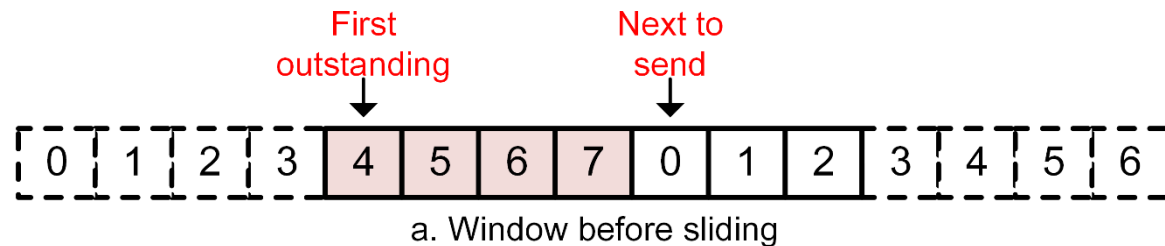
- 영역

1. 이미 확인응답된 패킷에 속하는 순서 번호
  - 패킷의 전송이 이미 완료 되었다고 간주하며, 사본을 보관하지 않음
2. 전송은 되었으나 확인응답이 되지 않은 패킷에 속하는 순서 번호
  - 미해결(Outstanding) 패킷 이라고 부름
  - 송신 측은 수신 측에서 해당 패킷들이 수신되었는지, 손실되었는지 확인을 기다림
3. 전송할 수 있으나 데이터가 응용 계층에서 수신되지 않은 패킷의 순서 번호
4. 윈도우가 이동하기 전까지 패킷을 전송할 수 없는 순서 번호



# 전송 계층 개요

- 전송 계층 프로토콜 (19/34)
  - N-프레임-후퇴(Go-Back-N) 프로토콜 (6/12)
    - 송신 윈도우
      - e.g., ackNo가 6인 확인응답이 도착할 경우 송신 윈도우 이동



# 전송 계층 개요

- 전송 계층 프로토콜 (20/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (7/12)

- 수신 윈도우

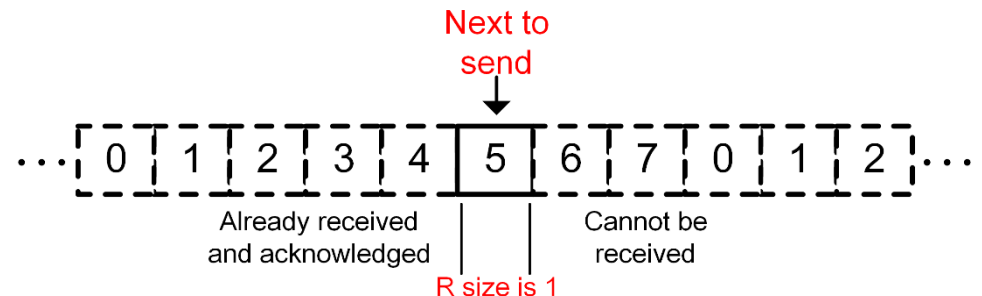
- 올바른 데이터 패킷 수신 및 확인응답 전송을 위해 사용
    - 수신 측에서 허용할 수 있는 데이터 패킷의 순서 번호를 포함하는 가상의 상자
    - 윈도우의 크기는 프로토콜에 따라 달라지며, GBN 프로토콜에서는 1
    - 올바른 패킷이 수신되면  $R_n = (R_n + 1) \text{ modulo } 2^m$  으로 이동
  - 영역

- $R_n$

- 수신되며 확인응답할 패킷의 순서 번호

- 그 외

- 이미 수신되었고 확인응답된 패킷의 순서 번호
        - 수신될 수 없는 패킷의 순서 번호



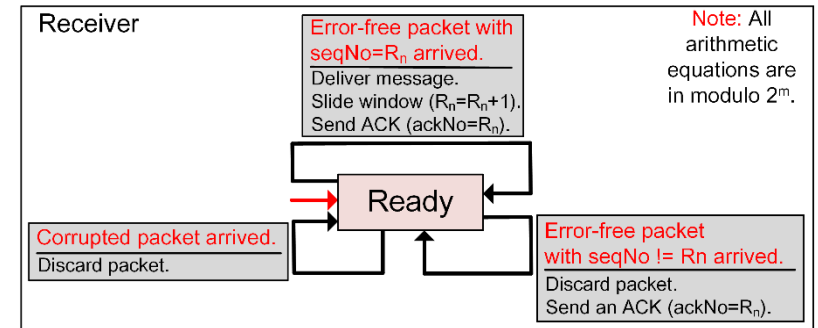
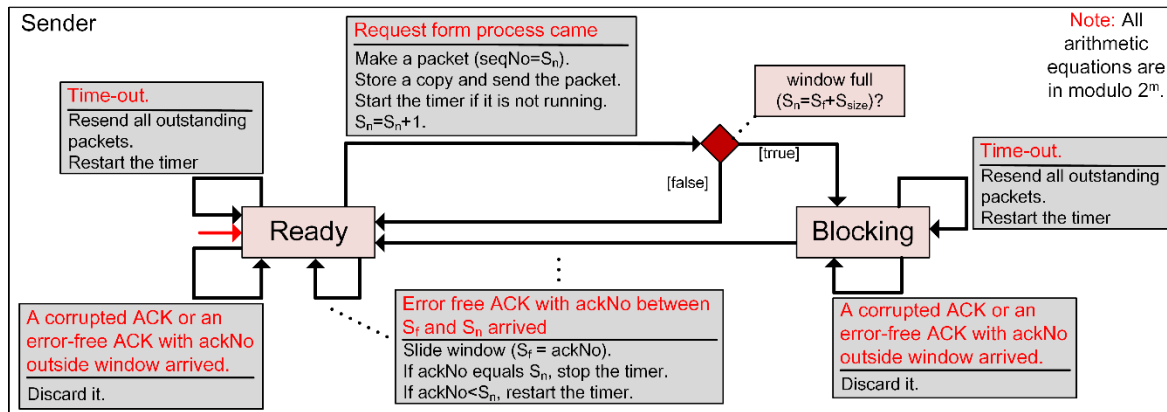
# 전송 계층 개요

---

- 전송 계층 프로토콜 (21/34)
- N-프레임-후퇴(Go-Back-N) 프로토콜 (8/12)
  - 타이머
    - 전송된 각각의 패킷에 대한 타이머가 존재하지만, 하나의 타이머 사용 가정
      - 첫 번째 미해결 패킷에만 타이머는 항상 먼저 만료
      - 타이머 만료 시, 모든 미해결 패킷 재전송
- 패킷 재전송
  - 타이머가 만료되면, 송신 측은 모든 미해결 패킷들을 재전송
  - 타임아웃 발생 시 윈도우 크기인  $N$ 만큼 후퇴하며 재전송이 발생

# 전송 계층 개요

- 전송 계층 프로토콜 (22/34)
  - N-프레임-후퇴(Go-Back-N) 프로토콜 (9/12)
    - FSM



- 한계점
  - 오류 발생 시 해당 패킷 이후의 정상적으로 수신된 패킷까지 재전송하여 비효율적

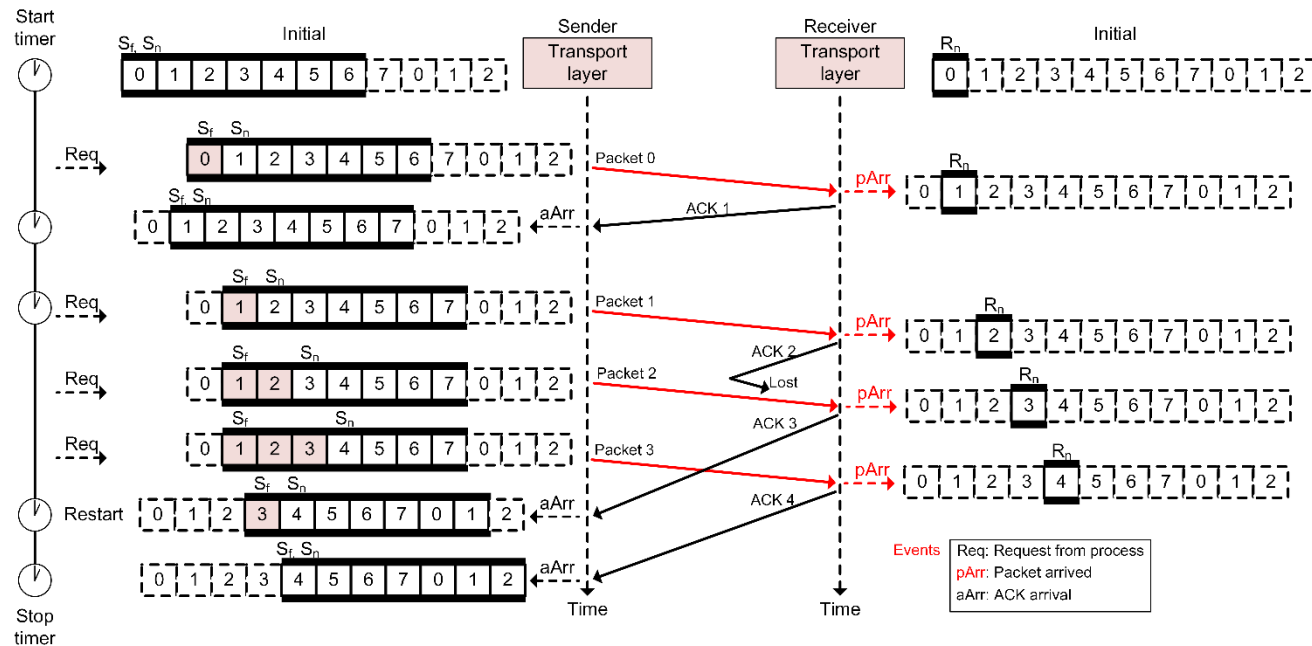
# 전송 계층 개요

- 전송 계층 프로토콜 (23/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (10/12)

- 예시 - ACK의 일부가 지연되거나 손실된 경우

- GBN에서는  
누적 값으로  
판단하기  
때문에 ACK 4가  
왔다는 것은  
Packet 3까지는  
잘 전송되었다고  
판단



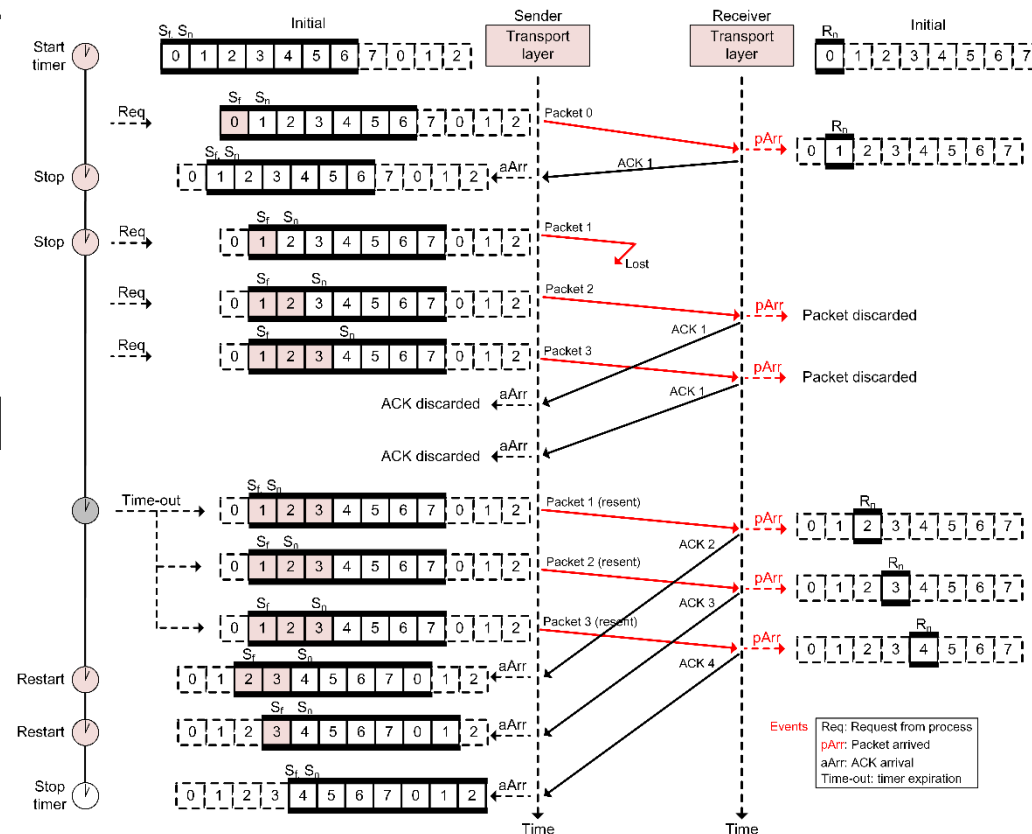
# 전송 계층 개요

- 전송 계층 프로토콜 (24/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (11/12)

- 예시 - 패킷이 손실된 경우 (1/2)

1. 송신자가 패킷 0~3을 전송
2. 패킷 1이 손실되고, 수신자는 패킷 2, 3을 수신하지만 순서가 맞지 않아 폐기
3. 수신자는 패킷 1을 기다리고 있음을 나타내기 위해 ACK1 전송
4. ACK1의 ackNo가 송신자의 송신 기준점과 같으므로 송신자는 해당 ACK를 폐기



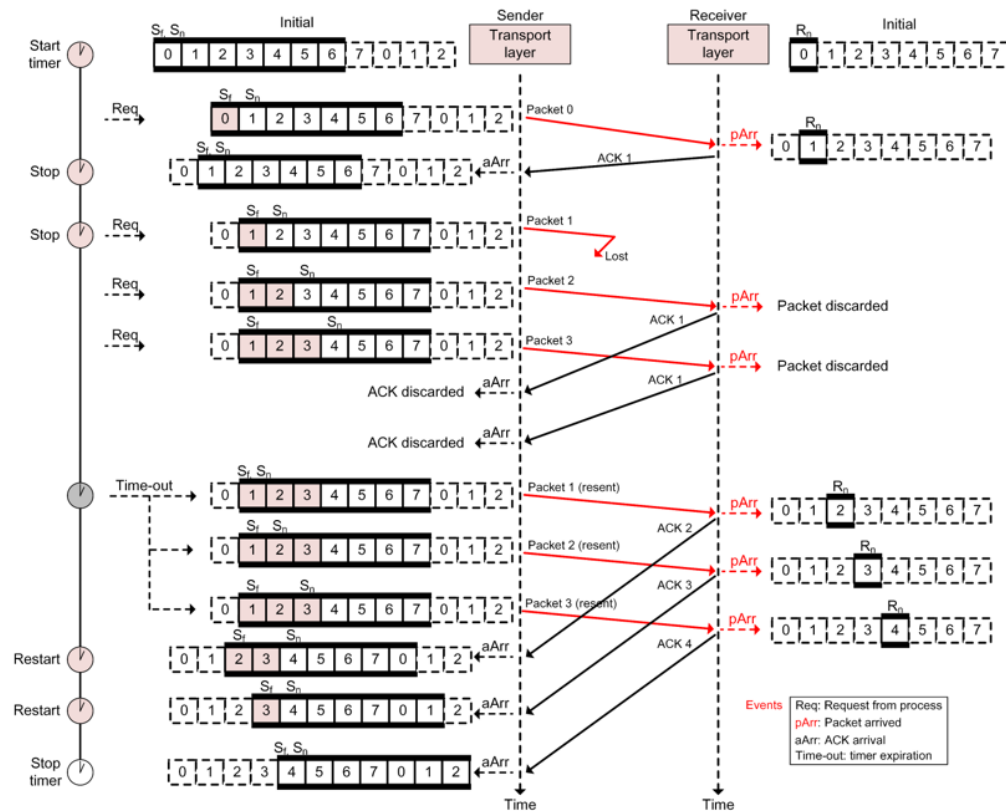
# 전송 계층 개요

- 전송 계층 프로토콜 (25/34)

- N-프레임-후퇴(Go-Back-N) 프로토콜 (12/12)

- 예시 - 패킷이 손실된 경우 (2/2)

5. 타임아웃 발생 시 송신자는 패킷 1, 2, 3을 재전송
6. 수신자는 재전송된 패킷을 정상적으로 수신하고 확인 응답(ACK)을 전송



# 전송 계층 개요

---

- 전송 계층 프로토콜 (26/34)
- 선택적 반복(Selective-Repeat) 프로토콜 (1/8)
  - 개요
    - 슬라이딩 윈도우를 기반으로 오류 제어를 수행하는 연결 지향 프로토콜
  - 특징
    - 송신 측은 확인응답을 수신하기 전에 여러 개의 패킷을 전송
    - 수신측은 여러 개의 패킷을 버퍼에 저장
    - 실제로 손실된 패킷만 선택적으로 재전송

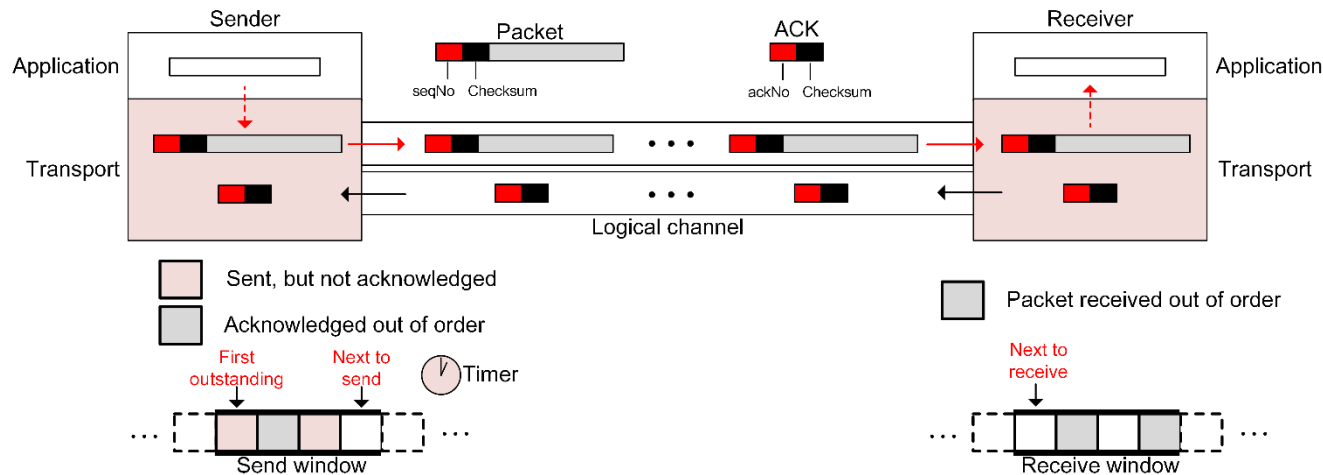
# 전송 계층 개요

- 전송 계층 프로토콜 (27/34)

- 선택적 반복(Selective-Repeat) 프로토콜 (2/8)

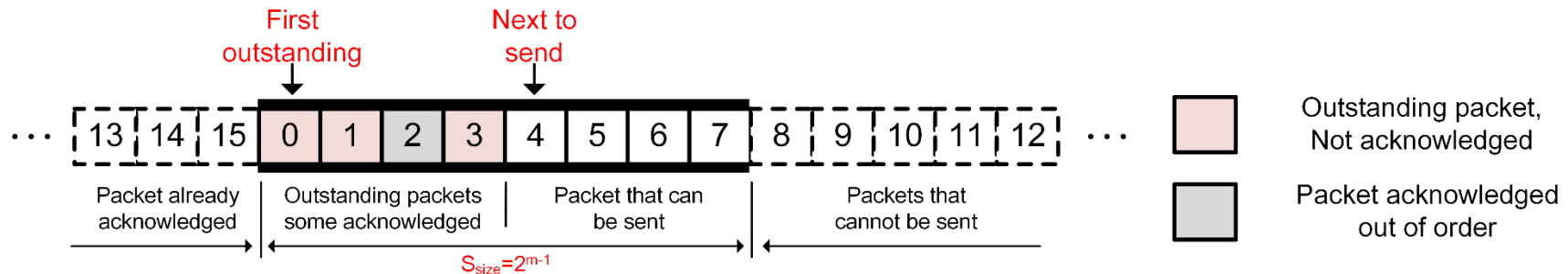
- 동작

1. 송신자는 윈도우 크기 내에서 여러 개의 패킷을 전송
2. 수신자는 패킷 수신 시 개별 ACK를 전송
3. 송신자는 ACK가 누락된 패킷만 재전송
4. 수신자는 올바른 순서로 버퍼에 패킷을 저장



# 전송 계층 개요

- 전송 계층 프로토콜 (28/34)
  - 선택적 반복(Selective-Repeat) 프로토콜 (3/8)
    - 송신 윈도우
      - 송신 윈도우의 크기는  $2^{m-1}$ 
        - e.g.,  $m = 4$ 일 경우, 순서 번호는 0부터 15까지이지만, 윈도우의 최대 크기는 8



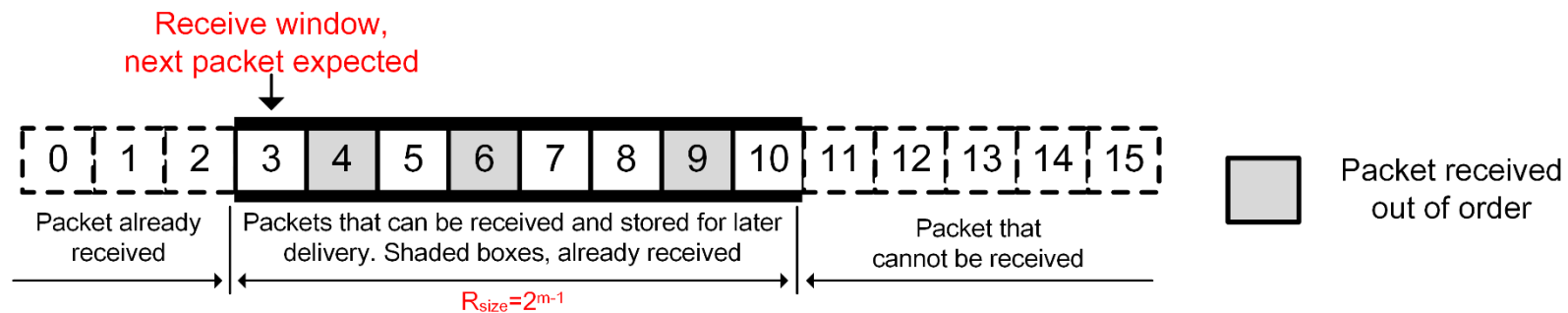
# 전송 계층 개요

- 전송 계층 프로토콜 (29/34)

- 선택적 반복(Selective-Repeat) 프로토콜 (4/8)

- 수신 윈도우

- 수신 윈도우의 크기는 송신 윈도우와 동일하게  $2^{m-1}$
    - 재전송으로 순서에 어긋나게 도착한 패킷들은 버퍼에 저장
    - 신뢰성 있는 프로토콜이므로 순서에 맞추기 위해 응용 계층으로 전달되기 전에 먼저 전송되었으나 수신되지 않은 패킷들을 기다려야 함



# 전송 계층 개요

---

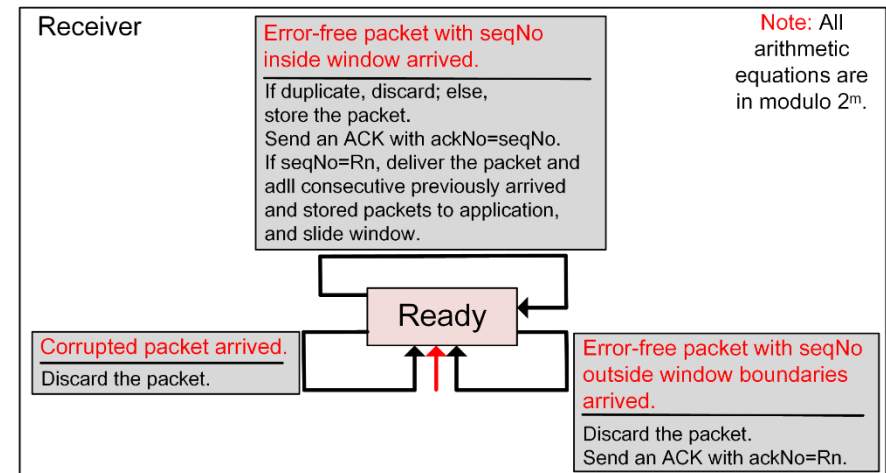
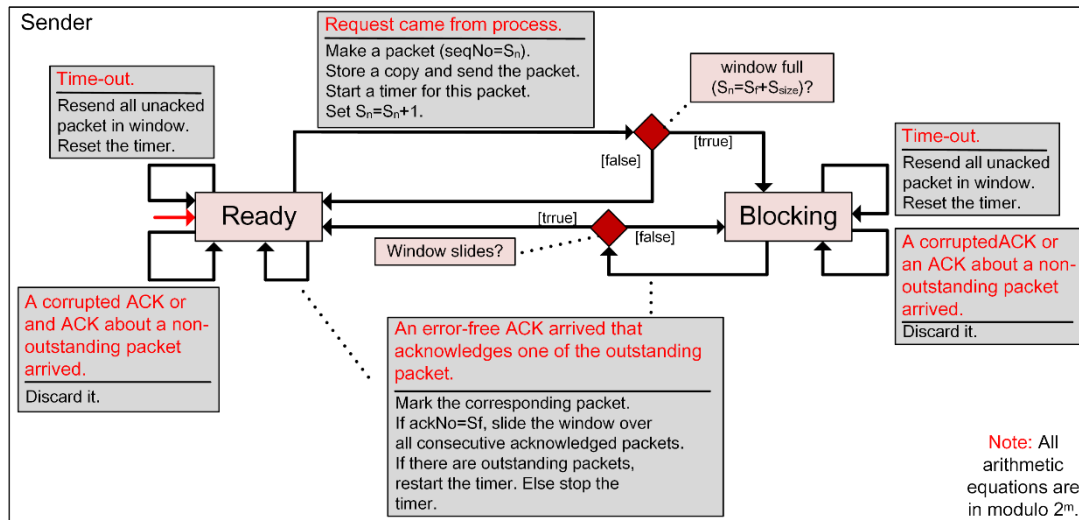
- 전송 계층 프로토콜 (30/34)
  - 선택적 반복(Selective-Repeat) 프로토콜 (5/8)
    - 타이머
      - 이론상으로는 미해결 패킷마다 하나씩의 타이머를 사용
      - SR 프로토콜을 구현하는 대부분의 전송 계층 프로토콜에서는 하나의 타이머만 사용
  - 확인응답
    - ackNo는 오류 없이 수신된 하나의 단일 패킷의 순서 번호를 의미
    - 다른 패킷들에 대한 피드백 없이 독립적으로 작동

# 전송 계층 개요

- 전송 계층 프로토콜 (31/34)

- 선택적 반복(Selective-Repeat) 프로토콜 (6/8)

- FSM

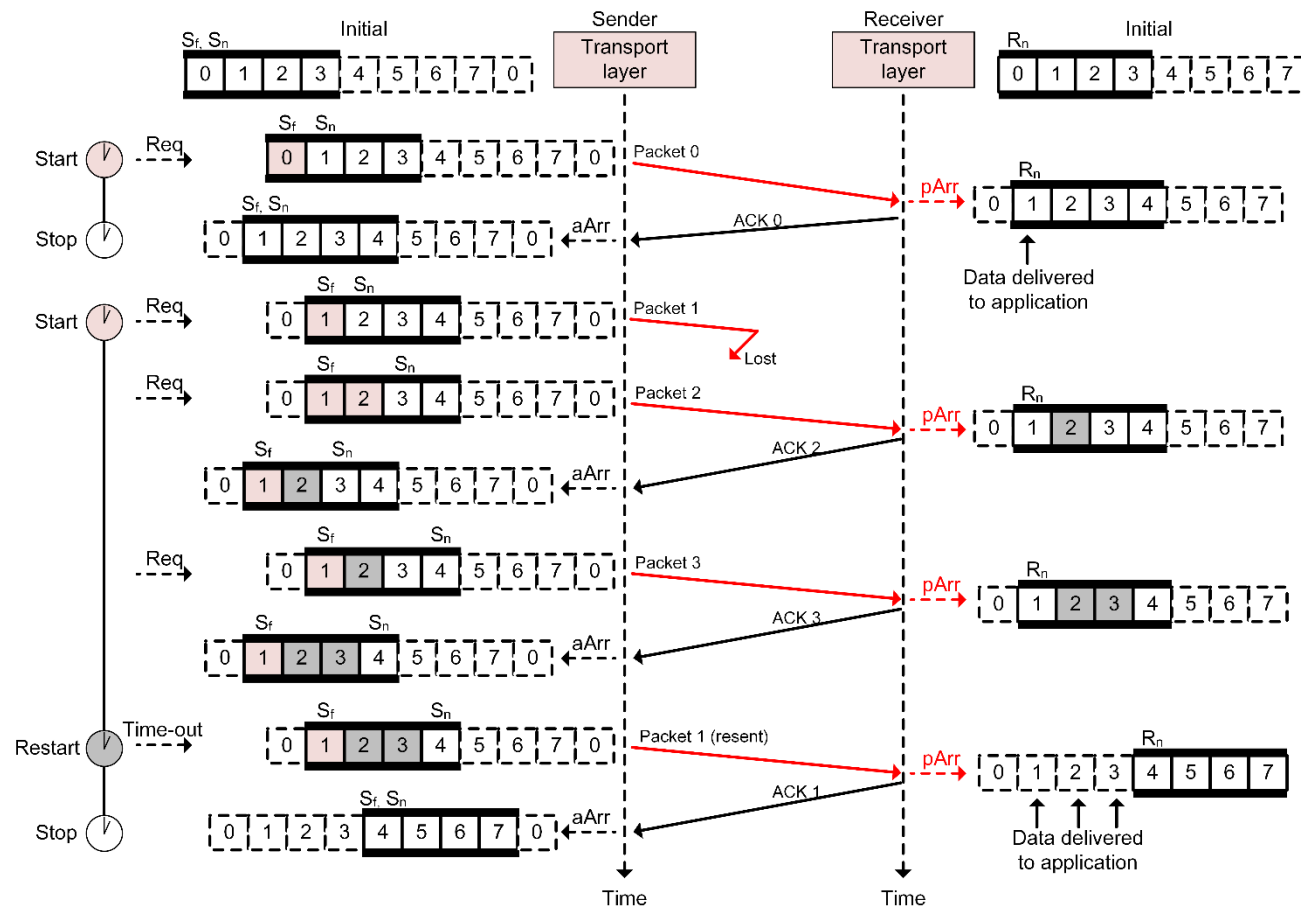


- 한계점

- 손실/오류가 발생한 패킷만 재전송하지만 송수신 측의 버퍼와 관리가 복잡함

# 전송 계층 개요

- 전송 계층 프로토콜 (32/34)
  - 선택적 반복(Selective-Repeat) 프로토콜 (7/8)
    - 예시 – 패킷이 손실된 경우



# 전송 계층 개요

- 전송 계층 프로토콜 (33/34)
  - 선택적 반복(Selective-Repeat) 프로토콜 (8/8)
    - 예제 13.3

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

- 풀이
  - GBN을 이용
    - 패킷 0, 1, 2가 훼손되지 않고 잘 수신됨을 의미
  - SR을 이용
    - 패킷 3이 훼손되지 않고 잘 수신됨을 의미
    - 다른 패킷들에 대한 정보는 없음

# 전송 계층 개요

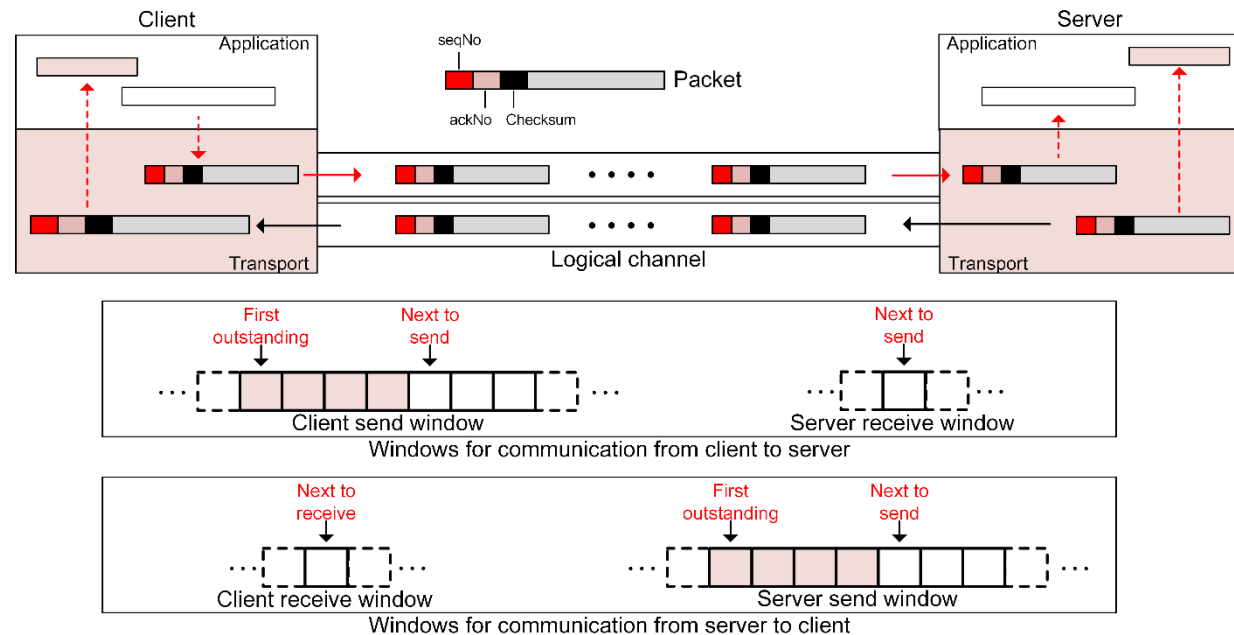
- 전송 계층 프로토콜 (34/34)

- 양방향 프로토콜: 피기배킹(Piggybacking)

- 실제 통신은 클라이언트와 서버간 양방향 데이터 전송
- 양방향 통신의 효율을 향상시키기 위해 사용

- 동작

- 데이터 전송 시,  
양방향 모두 패킷에  
데이터와 이전에  
수신한 데이터에  
대한 ACK 정보를  
실어서 전송



- 한계점

- 데이터가 없는 경우 ACK 전송이 지연될 수 있어 응답 지연 발생 가능

# 목 차

---

- 전송 계층 개요
- UDP (User Datagram Protocol)

# UDP

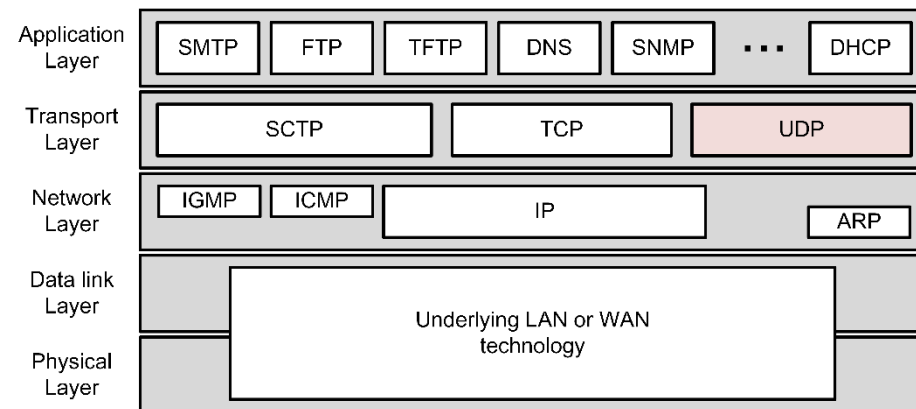
- UDP (User Datagram Protocol) (1/4)

- 정의

- 연결 설정 없이 데이터를 빠르게 전송하는데 사용되는 비연결형 전송 프로토콜

- 역할

- 응용 계층과 IP 계층 사이에 위치하며 응용 프로그램과 네트워크 동작 사이의 중재자 역할
- 포트 번호를 사용한 프로세스 대 프로세스 통신 생성
- 전송 계층에서 최소한의 수준에서 제어 메커니즘 제공
  - 흐름 제어 미제공
  - 수신된 패킷에 대한 확인응답 미제공
  - 오류 제어 기능 제공
    - 수신된 패킷에서 오류 탐지 시, 폐기



# UDP

---

- UDP (User Datagram Protocol) (2/4)

- UDP 특징

- 비연결형 서비스

- 연결 설정 및 해제 없음
    - UDP 패킷은 다른 패킷들과 독립적으로 처리

- 장점

- 지연 감소
    - 작은 오버헤드
    - 빠른 속도

- 단점

- 데이터 손실 위험
    - 비신뢰성
      - 순서 역전, 재전송

# UDP

- UDP (User Datagram Protocol) (3/4)
  - 프로세스 대 프로세스 통신
    - 포트 번호를 이용하여 프로세스 대 프로세스 통신 제공

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

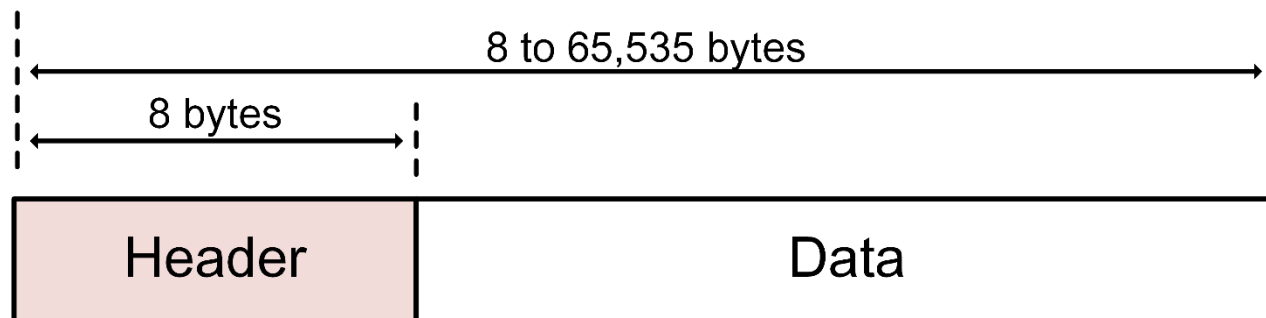
# UDP

---

- UDP (User Datagram Protocol) (4/4)
  - 대표적인 응용
    - 단순한 요청/응답 통신
      - 흐름 제어 및 오류 제어에 대한 요구가 적은 프로세스에 적합
      - e.g., DNS (Domain Name System) 질의/응답
    - 응용 계층에서 신뢰성 제어를 수행하는 프로세스
      - UDP에서 제공하지 않는 오류 제어 및 재전송 기능을 응용 계층에서 구현
      - e.g., TFTP (Trivial File Transfer Protocol)
    - 멀티캐스팅(Multicasting)
      - 멀티캐스팅을 지원하는 전송 프로토콜로 적합
      - e.g., TCP는 멀티캐스팅을 지원하지 않으므로 UDP 사용
    - 관리 및 경로 갱신 프로토콜
      - e.g., SNMP (Simple Network Management Protocol)과 같은 네트워크 관리 및 RIP (Routing Information Protocol)와 같은 경로 갱신

# UDP

- 사용자 데이터그램(User Datagram) (1/6)
  - 정의
    - UDP에서 사용하는 데이터 단위
  - 특징
    - 8바이트의 고정 헤더
    - 헤더를 포함한 전체는 8~65,535 바이트 크기



# UDP

- 사용자 데이터그램(User Datagram) (2/6)

- 헤더 구조

- 발신지 포트 번호

- 발신지 호스트 상에서 수행되는 프로세스가 사용하는 포트 번호
    - 16비트의 크기를 가지므로 0~65,535의 범위 내에서 사용 가능
    - 발신지 호스트가 요청을 전송하는 클라이언트일 경우, 포트 번호는 프로세스에 의해 요청되고 발신지 호스트의 UDP 소프트웨어가 선택한 임시 포트 번호
    - 발신지 호스트가 응답을 전송하는 서버일 경우, 포트 번호는 잘 알려진 포트 번호

|                    |                         |    |
|--------------------|-------------------------|----|
| 0                  | 16                      | 32 |
| Source port number | Destination port number |    |
| Total length       | Checksum                |    |

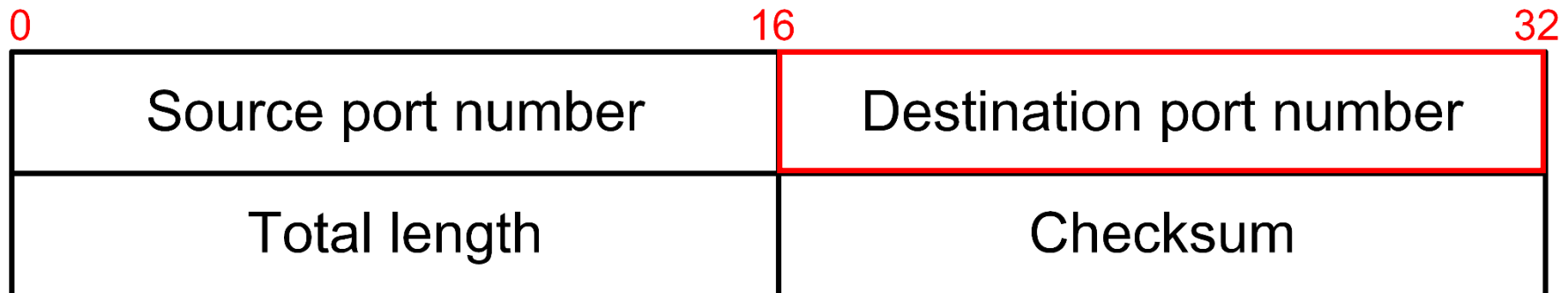
# UDP

- 사용자 데이터그램(User Datagram) (3/6)

- 헤더 구조

- 목적지 포트 번호

- 목적지 호스트 상에서 수행되는 프로세스가 사용하는 포트 번호
    - 16비트의 크기를 가지므로 0~65,535의 범위 내에서 사용 가능
    - 목적지 호스트가 클라이언트에 의해 요청이 보내어지는 서버일 경우, 포트 번호는 잘 알려진 포트 번호
    - 목적지 호스트가 서버가 보낸 응답을 받는 클라이언트일 경우, 포트 번호는 임시 포트 번호
      - 서버는 수신한 요청 패킷으로부터 얻을 수 있는 수신된 임시 포트 번호를 복사하여 사용



# UDP

- 사용자 데이터그램(User Datagram) (4/6)

- 헤더 구조

- 길이

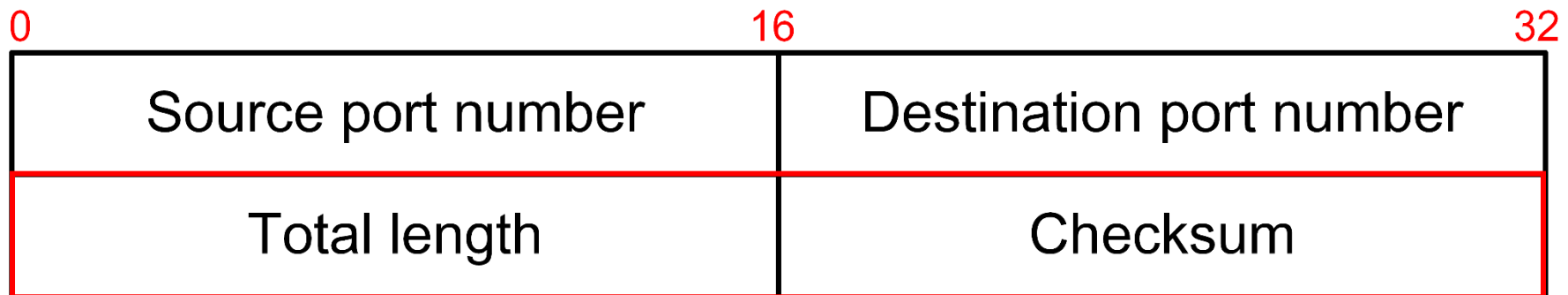
- 헤더와 데이터를 합한 사용자 데이터그램의 전체 길이

- 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

- 검사합

- 헤더와 데이터를 모두 포함한 사용자 데이터그램 전체에 대해 오류를 탐지하기 위해 사용



# UDP

- 사용자 데이터그램(User Datagram) (5/6)

- 헤더 구조

- 예제 14.1 (1/2)

다음은 16진수의 형식으로 UDP 헤드를 덤프한 것이다. 아래 질문에 답하라.

**CB84 000D 001C 001C**

- a. 발신지/목적지 포트 번호는 얼마인가?
- b. 사용자 데이터그램의 총 길이는 얼마인가?
- c. 데이터의 길이는 얼마인가?
- d. 데이터의 전송 방향이 어떻게 되는가?

- 풀이

- (a) 발신지/목적지 포트

- 발신지

- 처음 네 개의 16진수는 0xCB84이며, 이를 10진수로 표현 시 52,100

- 목적지

- 두 번째 4개의 16진수는 0x000D이며, 이를 10진수로 표현 시 13

- (b) 사용자 데이터그램 총 길이

- 세 번째 네 개의 16진수는 0x001C이며, 이를 10진수로 표현 시 28바이트

# UDP

- 사용자 데이터그램(User Datagram) (6/6)

- 헤더 구조

- 예제 14.1 (2/2)

다음은 16진수의 형식으로 UDP 헤드를 덤프한 것이다. 아래 질문에 답하라.

**CB84 000D 001C 001C**

- a. 발신지/목적지 포트 번호는 얼마인가?
- b. 사용자 데이터그램의 총 길이는 얼마인가?
- c. 데이터의 길이는 얼마인가?
- d. 데이터의 전송 방향이 어떻게 되는가?

- 풀이

- (c) 데이터의 길이

- 총 길이가 28바이트이고, 헤더의 길이가 8바이트이므로  $28 - 8 = 20$ 바이트

- (d) 데이터의 전송 방향

- 목적지 포트가 13으로 이는 잘 알려진 포트
    - 클라이언트로부터 서버로의 전송 방향

# UDP

---

- UDP 서비스 (2/11)

- 비연결형 서비스

- 연결 설정/연결 해제 과정이 없음
- UDP에 의해 전송되는 각각의 사용자 데이터그램은 서로 독립적
  - 사용자 데이터그램들 사이에 번호가 붙지 않음
  - 보낸 순서와 다르게 도착 가능
  - 서로 다른 경로로 전송 가능

- 흐름 제어와 혼잡 제어

- 흐름 제어와 혼잡 제어는 미제공
  - 윈도우 메커니즘이 존재하지 않아 흐름 제어 미제공
  - 비연결형 프로토콜이므로 혼잡 제어 미제공

# UDP

- UDP 서비스 (3/11)

- 오류 제어 (1/3)

- 검사합이 유일한 오류 제어 메커니즘
- 메시지의 손실/중복을 송신자가 알 수 없음
- 수신자가 검사합을 이용하여 오류 감지할 경우 폐기
  - 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

|                                                                |                |                                    |
|----------------------------------------------------------------|----------------|------------------------------------|
| 32-bit source IP address                                       |                |                                    |
| 32-bit destination IP address                                  |                |                                    |
| All 0s                                                         | 8-bit protocol | 16-bit UDP total length            |
| Source port address<br>16 bits                                 |                | Destination port address<br>16bits |
| UDP total length<br>16 bits                                    |                | Checksum<br>16 bits                |
| Data                                                           |                |                                    |
| (Padding must be added to make the data a multiple of 16 bits) |                |                                    |

# UDP

- UDP 서비스 (4/11)

- 오류 제어 (2/3)
  - 예제 14.2

다음 체크섬을 계산하라.

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

10011001 00010010 → 153.18  
00001000 01101001 → 8.105  
10101011 00000010 → 171.2  
00001110 00001010 → 14.10  
00000000 00010001 → 0 and 17  
00000000 00001111 → 15  
00000100 00111111 → 1087  
00000000 00001101 → 13  
00000000 00001111 → 15  
00000000 00000000 → 0(Checksum)  
01010100 01000101 → T and E  
01010011 01010100 → S and T  
01001001 01001110 → I and N  
01000111 00000000 → G and 0(Padding)  
-----  
10010110 11101011 → Sum  
**01101001 00010100** → Checksum

# UDP

- UDP 서비스 (5/11)

- 오류 제어 (3/3)
  - 예제 14.3

다음의 가상 상황에서 검사합으로 어떤 값이 전송되는가?

- a. 송신자는 검사합을 포함하지 않기로 결정한다.
- b. 송신자는 검사합을 포함하기로 결정했지만, 검사합의 값이 모두 1이다.
- c. 송신자는 검사합을 포함하기로 결정했지만, 검사합의 값이 모두 0이다.

- 풀이

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

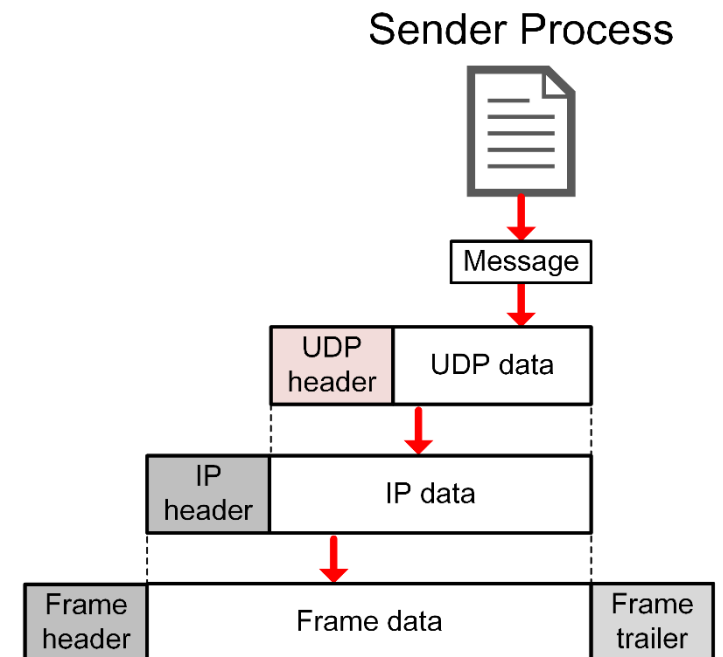
# UDP

- UDP 서비스 (6/11)

- 캡슐화와 역캡슐화 (1/2)

- 캡슐화

- 송신 프로세스가 생성한 메시지에 UDP 헤더 추가
    - UDP는 소켓 주소와 함께 사용자 데이터그램을 IP로 전달
    - IP는 IP 헤더를 붙여 네트워크 계층으로, 데이터링크 계층은 프레임 헤더를 붙여 물리 계층으로 전달
    - 물리 계층은 비트들을 전기 또는 광 신호로 부호화한 후 원격 시스템으로 전달



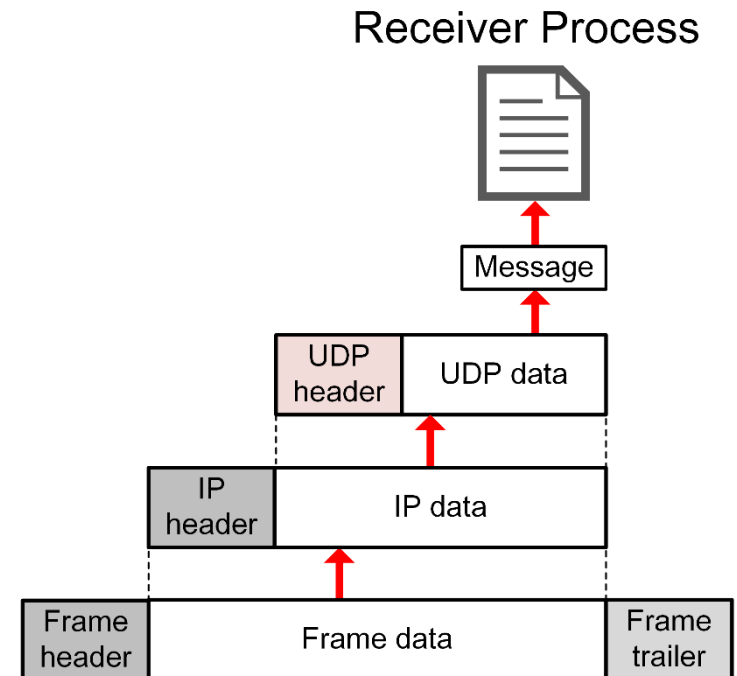
# UDP

- UDP 서비스 (7/11)

- 캡슐화와 역캡슐화 (2/2)

- 역캡슐화

- 메시지가 수신 측에 도착 시, 물리 계층은 신호를 비트로 복호화한 후 이를 데이터링크 계층으로 전달
    - 데이터링크 계층은 헤더 검사 후 제거하여 IP 계층으로 전달
    - IP 계층은 IP 헤더 검사 후 오류가 없으면 헤더 제거 후 UDP 계층으로 전달
    - UDP 계층은 헤더 검사 후 오류가 없는 경우, 수신 프로세스에 메시지 전달



# UDP

- UDP 서비스 (8/11)

- 큐잉(Queuing) (1/3)

- 정의

- UDP에서 포트 번호와 연결하여 송신 및 수신 메시지를 임시 저장하는 큐(Queue)를 관리하는 방식

\*큐(Queue)

먼저 들어온 데이터가 먼저 나가는 FIFO(First-In, First-Out, 선입선출) 구조의 자료구조.

- 특징

- 송신 큐(Outgoing Queue)
      - 전송할 메시지 저장
    - 수신 큐(Incoming Queue)
      - 수신한 데이터그램 저장
    - 프로세스 실행 시 큐가 생성되고 종료 시 제거
    - 송신 큐 초과 시 전송 대기, 수신 큐 초과 시 데이터그램 폐기

# UDP

- UDP 서비스 (9/11)

- 큐잉(Queuing) (2/3)

- 클라이언트 측

- 큐 생성

- 프로세스 실행 시 임시 포트 번호(Ephemeral Port)를 사용하여 큐 생성

- 송신

- 송신 큐의 메시지를 UDP가 하나씩 꺼내 헤더를 추가한 후 IP 계층으로 전달

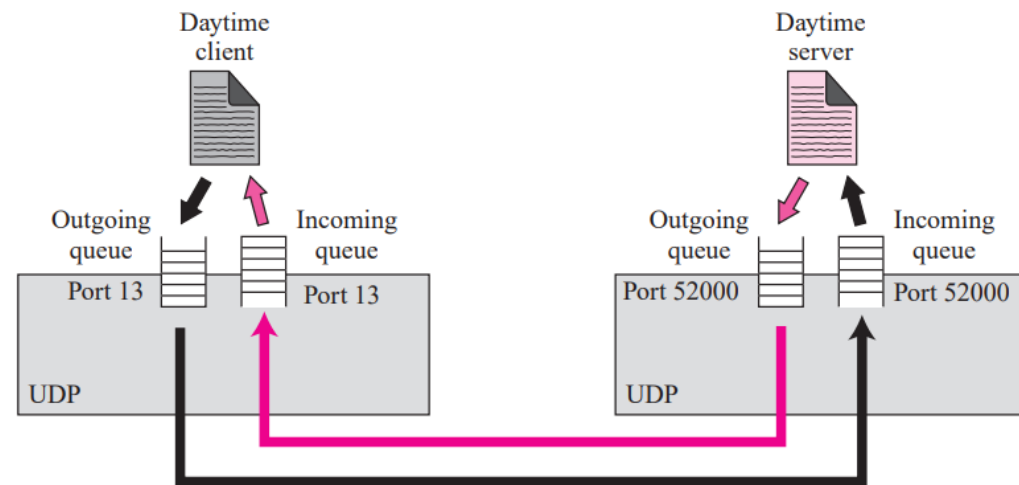
- 수신

- 목적지 포트 번호에 해당하는 수신 큐에 데이터그램 저장

- 수신 큐가 없으면 데이터그램  
폐기 및 ICMP 포트 도달 불가  
메시지 전송

- 특징

- 서로 다른 서버에서 수신한  
메시지도 동일한 수신 큐에  
저장



# UDP

- UDP 서비스 (10/11)

- 큐잉(Queuing) (3/3)

- 서버 측

- 큐 생성

- 서버 실행 시 잘 알려진 포트 번호(Well-known Port)를 사용하여 큐 생성

- 수신

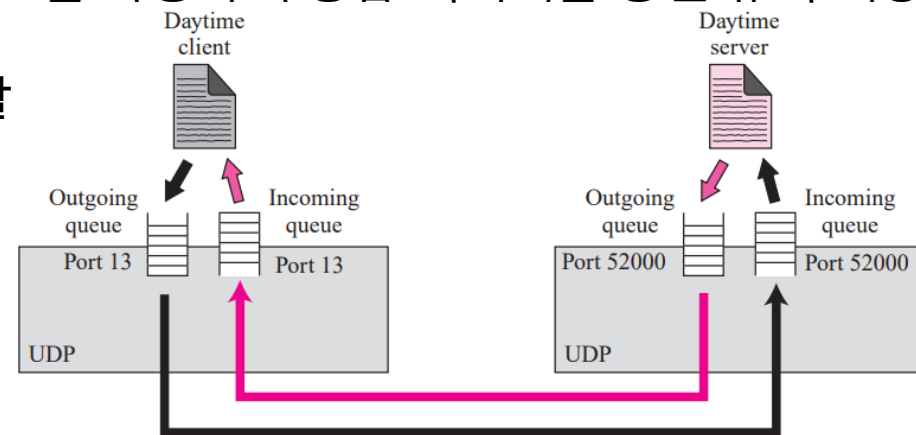
- 목적지 포트 번호에 해당하는 수신 큐에 데이터그램 저장
      - 수신 큐가 없으면 데이터그램 폐기 및 ICMP 포트 도달 불가 메시지 전송

- 송신

- 클라이언트 요청의 출발지 포트 번호를 사용하여 응답 메시지를 송신 큐에 저장
      - UDP가 메시지를 하나씩 꺼내  
헤더를 추가한 후 IP 계층으로 전달

- 특징

- 서로 다른 클라이언트에서 수신한  
메시지도 동일한 수신 큐에 저장



# UDP

---

- UDP 서비스 (11/11)

- 다중화와 역다중화

- 다중화

- 송신 측에서 UDP는 하나지만, 각 송신 프로세스마다 고유한 포트 번호를 사용하는 N:1 관계
    - 각 프로세스는 포트 번호로 구분
    - 각 프로세스의 메시지에 UDP 헤더를 추가하여 IP 계층에 전달

- 역다중화

- 수신 측에서 UDP는 하나지만, 여러 프로세스가 동시에 UDP를 사용하는 1:N 관계
    - 수신한 데이터그램의 목적지 포트 번호를 확인
    - 해당 포트 번호에 매핑된 정확한 수신 프로세스로 메시지 분배

# UDP

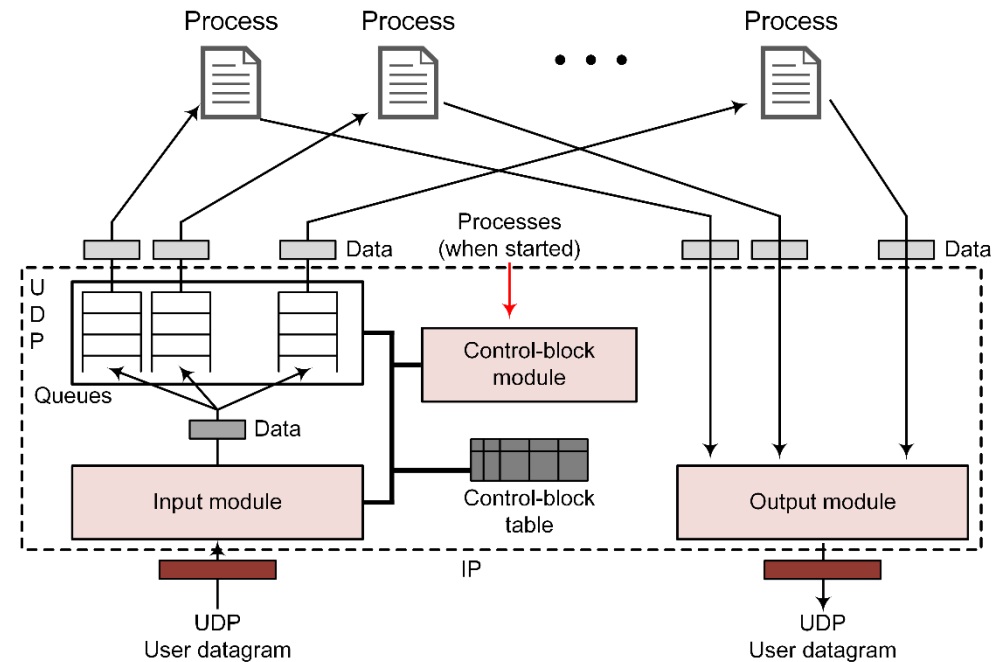
- UDP 패키지 (1/10)

- 정의

- UDP가 UDP 패킷을 송신 및 수신하는 과정을 표현한 것

- 구성 요소

- 입력 큐(Input Queue)
  - 제어 블록 테이블 (Control-Block Table)
  - 제어 블록 모듈 (Control-Block Module)
  - 입력 모듈(Input Module)
  - 출력 모듈(Output Module)

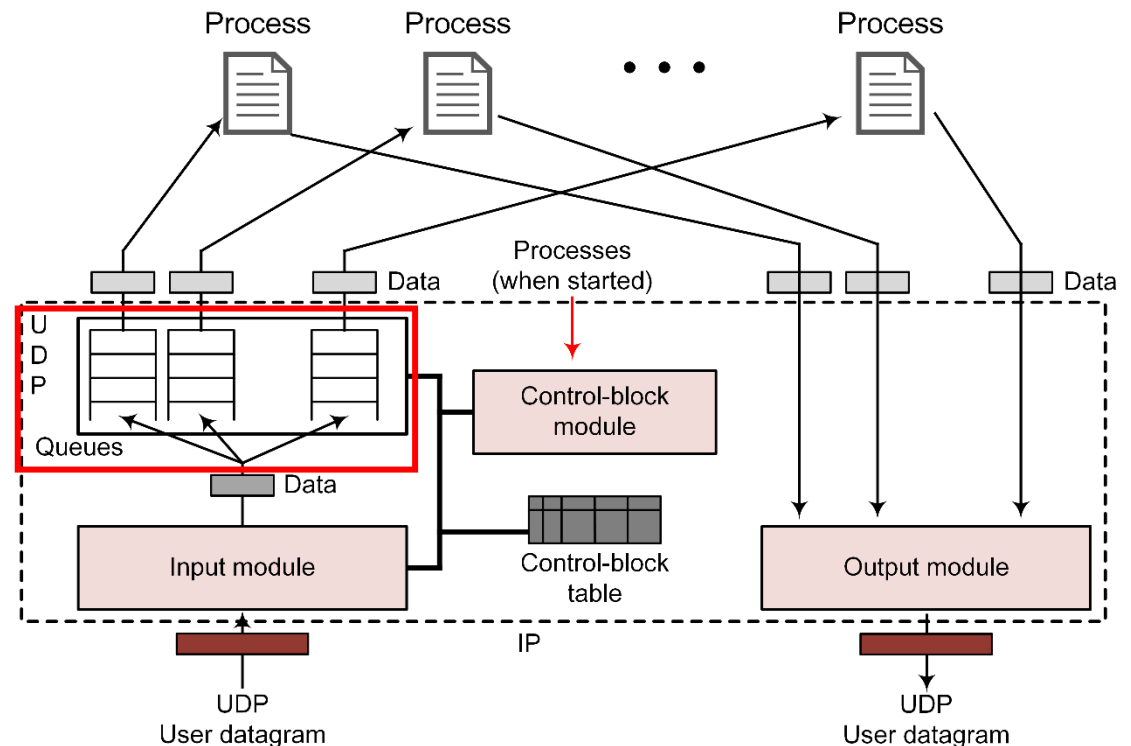


# UDP

- UDP 패키지 (2/10)

- 입력 큐(Input Queue)

- 각각의 프로세스 당 하나씩 지정되는 입력 큐의 집합을 사용
- 목적지 포트 번호를 기준으로 해당 프로세스의 입력 큐에 데이터그램 전달
- 프로세스가 데이터를 읽을 때까지 수신한 데이터그램을 대기시킴

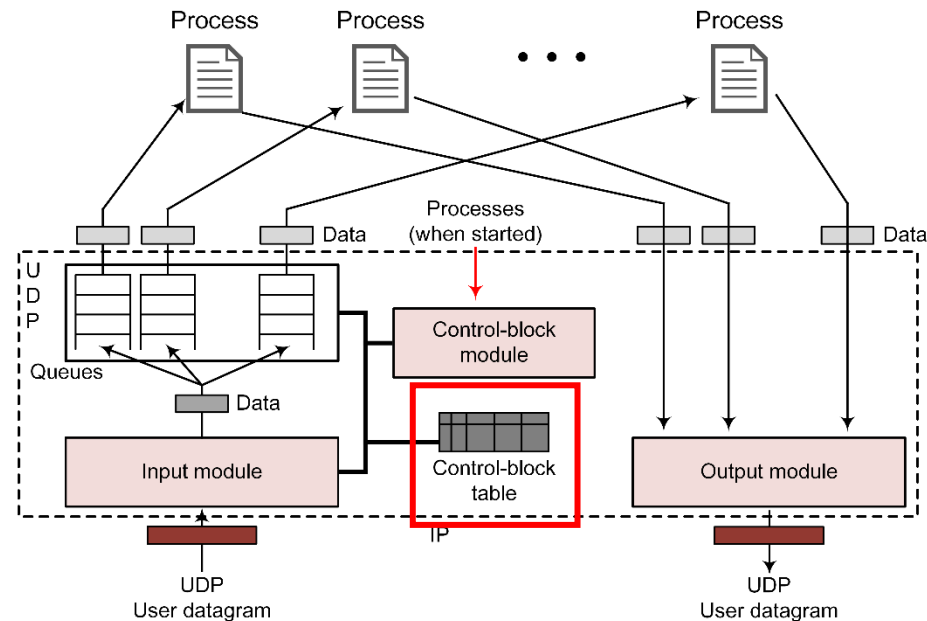


# UDP

- UDP 패키지 (3/10)

- 제어 블록 테이블(Control-Block Table)

- 열린 포트를 추적하기 위해 사용
- 테이블의 각 엔트리는 FREE이거나 IN-USE의 두 값 중 하나를 가짐
  - 상태(State), 프로세스 ID, 포트 번호, 상응하는 큐 번호 등 최소한 4개의 필드 소유



# UDP

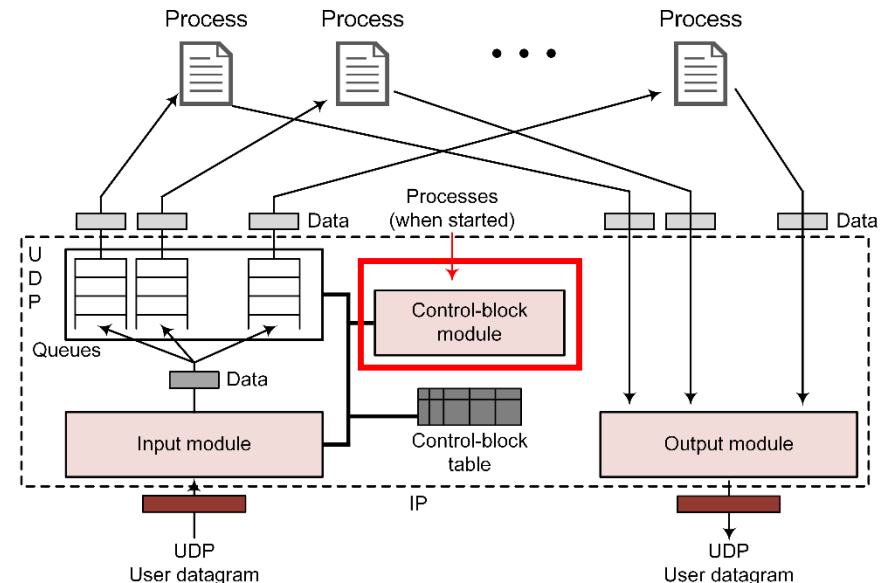
## • UDP 패키지 (4/10)

### • 제어 블록 모듈(Control-Block Module)

- 제어 블록 테이블 관리 담당
- 프로세스가 시작될 때 운영체제로부터 포트 번호 요청
  - 서버에게 잘 알려진 포트, 클라이언트에게는 임시 포트 번호 배당
- 프로세스는 프로세스 ID와 포트 번호를 제어 블록 모듈에 보내어 프로세스를 위한 엔트리를 테이블에 생성
- 큐 번호의 필드는 0의 값 가짐

#### UDP\_Control\_Block\_Module (User Datagram)

```
{  
  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
```

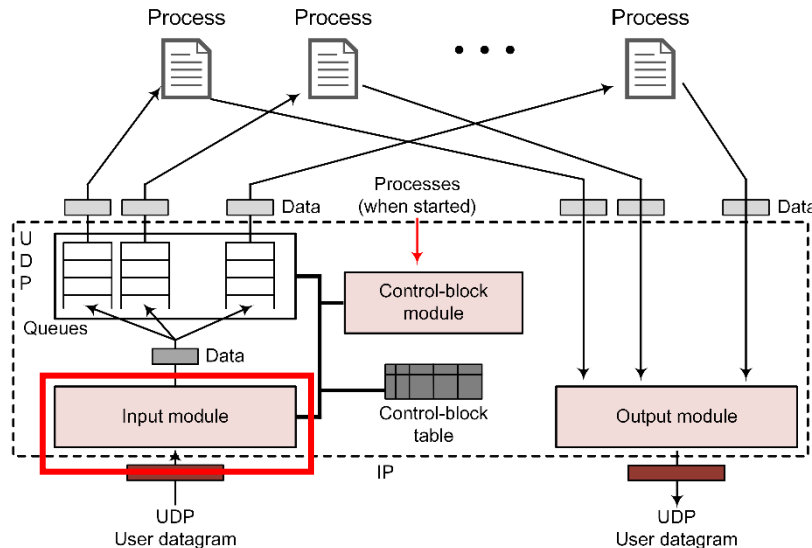


# UDP

## • UDP 패키지 (5/10)

### • 입력 모듈(Input Module)

- IP로부터 사용자 데이터그램 수신
- 제어 블록 테이블을 탐색하여 수신한 사용자 데이터그램과 같은 포트 번호를 갖는 엔트리 탐색
  - 엔트리가 있을 경우, 입력 모듈은 엔트리 내의 정보를 사용하여 데이터그램을 큐에 삽입
  - 없을 경우, ICMP 메시지 생성



#### 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

- UDP 패키지 (6/10)

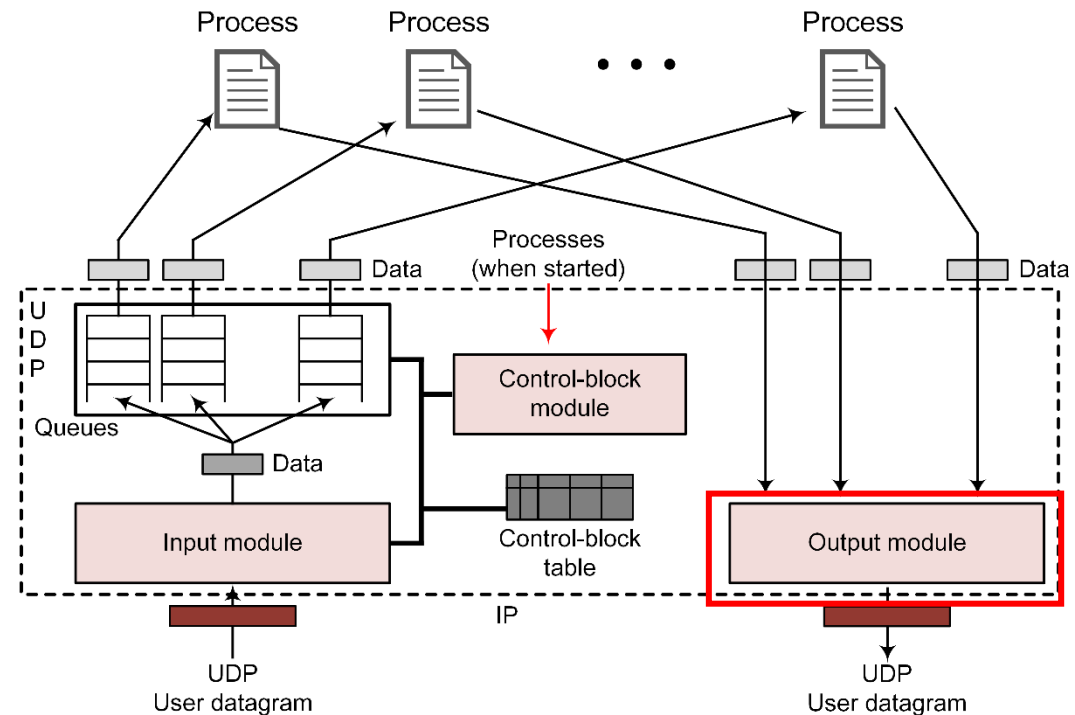
- 출력 모듈(Output Module)

- 사용자 데이터그램 생성 후 전송하는 임무 담당

- 의사코드

**UDP\_Output\_Module (User Datagram)**

```
{  
  Create a user datagram  
  Send the user datagram  
  Return  
} // End module
```



# UDP

- UDP 패키지 (7/10)

- 예제 14.4

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 사용되어 있음을 알 수 있다. 입력 모듈은 데이터를 38번 큐에 보낸다. 제어 블록 테이블은 변하지 않는다. 이때 표에서 해당 테이블을 찾아라.

| 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 (8/10)

- 예제 14.5

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. 이때 테이블의 변화를 작성하라.

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

# UDP

- UDP 패키지 (9/10)

- 예제 14.6

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. 이때 테이블의 변화를 작성하라.

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

# UDP

---

- UDP 패키지 (10/10)

- 예제 14.7

몇 초 후 52,222 포트에 사용자 데이터그램이 도착한다. 이 경우, 이후 상황에 대해 설명하라.

- 풀이
      - 입력 모듈이 테이블을 검사한 결과 이 목적지를 위한 엔트리를 발견하지 못함
      - 사용자 데이터그램은 폐기되고 발신지에 포트 도달 불가 메시지를 보낼 것을 ICMP에 요청

---

# Thanks!

서진우([jinwoo@pel.sejong.ac.kr](mailto:jinwoo@pel.sejong.ac.kr))