2026/10/12, 2026 네트워크 기초 세미나

TCP/IP 프로토콜

- 15장 TCP -

김 민 식 (minsik@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

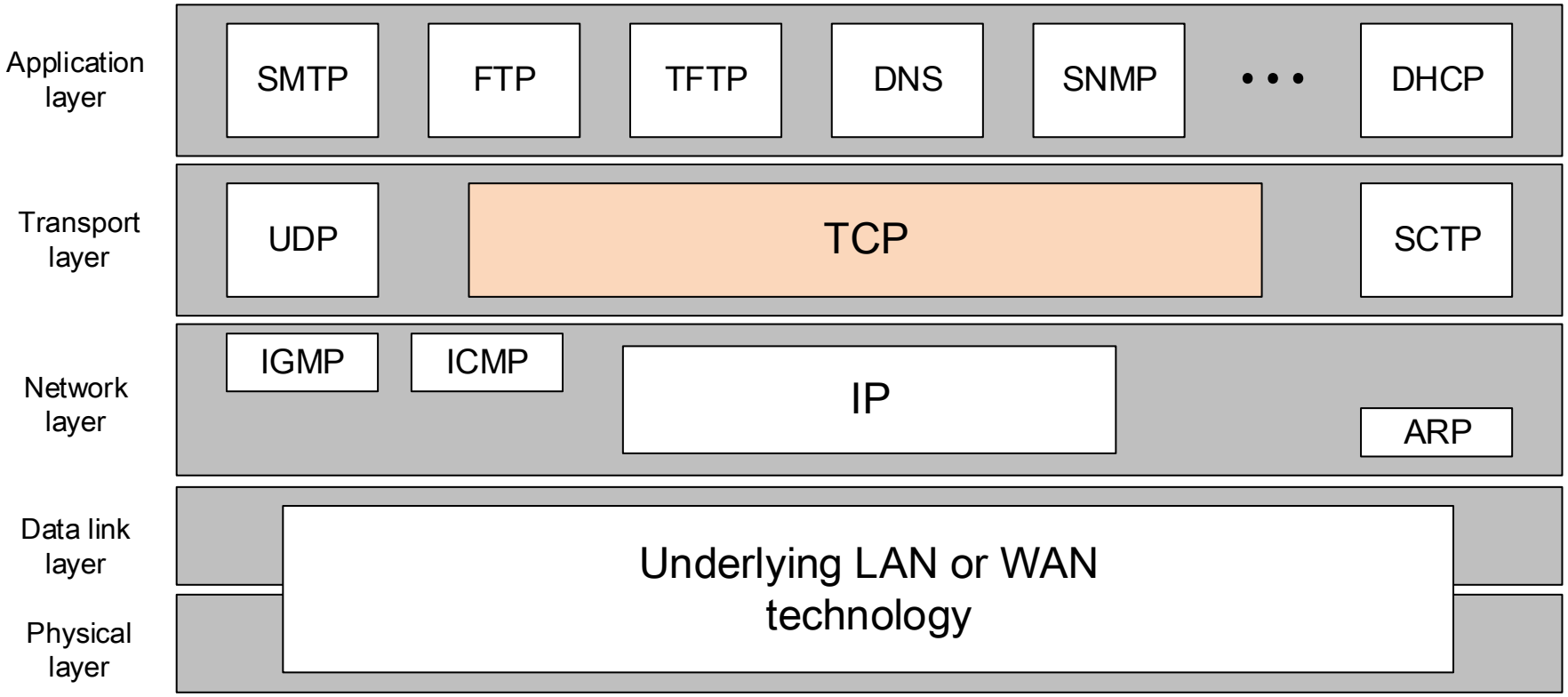
- TCP 개요
- TCP 연결 및 전송
- TCP 제어
- TCP 동작 메커니즘 및 구현

목 차

- TCP 개요
- TCP 연결 및 전송
- TCP 제어
- TCP 동작 메커니즘 및 구현

TCP 개요

- TCP(Transmission Control Protocol)
  - 정의
    - 연결을 설정한 후 신뢰성 있는 바이트 스트림 전달 서비스를 제공하는 전송 계층 프로토콜



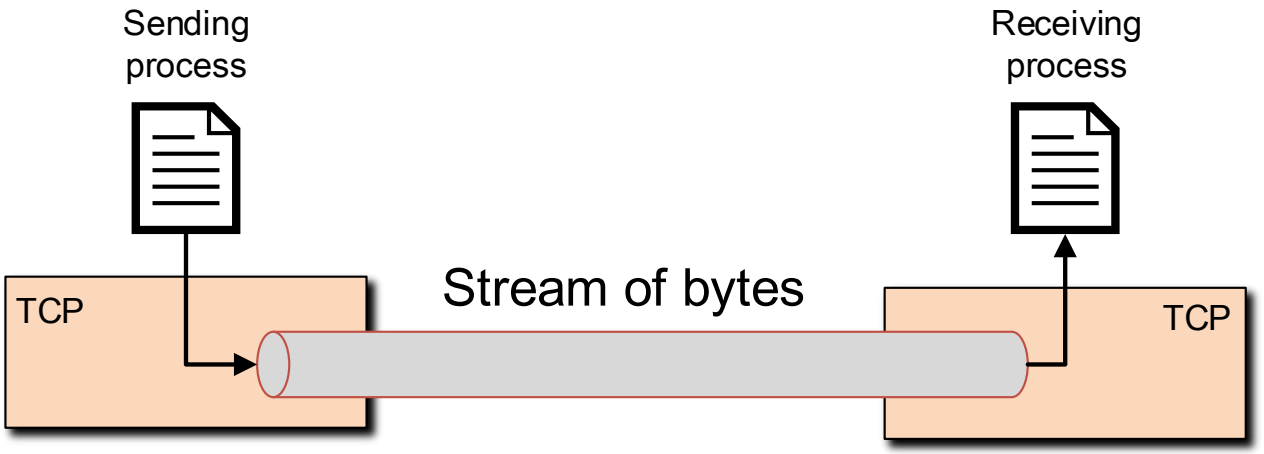
TCP 개요

- TCP 서비스 (1/5)
  - 프로세스 대 프로세스 통신
    - TCP는 포트 번호를 이용하여 프로세스 대 프로세스 통신을 제공함

포트 번호	프로토콜	설명
7	Echo	받은 데이터그램을 그대로 송신자에게 반환
9	Discard	받은 데이터그램을 폐기
13	Daytime	날짜와 시간을 반환
17	Quote	일일 격언 반환
19	Chargen	무작위 문자 스트림 반환
20, 21	FTP	파일 전송 프로토콜(데이터, 제어)
23	TELNET	원격 터미널 접속 프로토콜
25	SMTP	이메일 송신 프로토콜
53	DNS	도메인 이름과 IP 주소 간 변환
79	Finger	특정 사용자 정보 조회
80	HTTP	웹 문서 전송

TCP 개요

- TCP 서비스 (2/5)
  - 스트림 배달 서비스 (1/2)
    - 응용 프로그램의 데이터를 연속된 바이트 스트림 형태로 송수신함
    - 송신 프로세스는 바이트 스트림을 생성하고, 수신 프로세스는 전달된 바이트 스트림을 읽음



TCP 개요

- TCP 서비스 (3/5)
  - 스트림 배달 서비스 (2/2)
    - 송신 / 수신 버퍼(Buffer)
      - 송신 프로세스와 수신 프로세스의 데이터 처리 속도가 서로 다를 수 있으므로 데이터를 임시 저장하기 위한 버퍼를 사용함
      - 송신 버퍼는 전송할 데이터와 전송 후 아직 확인 응답되지 않은 데이터를 저장함
      - 수신 버퍼는 수신한 데이터를 응용 프로그램이 읽기 전까지 저장함
    - 세그먼트(Segment)
      - TCP는 바이트 스트림을 여러 바이트 단위로 묶어 세그먼트 형태로 전송함
      - 각 세그먼트에는 제어 정보를 위한 TCP 헤더가 추가됨
      - 세그먼트의 크기는 항상 동일하지 않음

*버퍼(Buffer)
데이터를 바로 처리하지 못할 때를 대비해 일시적으로 저장해두는 메모리 공간

TCP 개요

- TCP 서비스 (4/5)
  - 전이중 통신(Full-Duplex)
    - 송신과 수신이 동시에 가능하여 데이터를 양방향으로 전송할 수 있음
    - 각 TCP는 송신 버퍼와 수신 버퍼를 가짐
  - 다중화(Multiplexing) 및 역다중화(Demultiplexing)
    - 송신 측에서 여러 프로세스의 데이터를 모아 전송하고, 수신 측에서 포트 번호를 기준으로 해당 프로세스에 전달함
    - 연결 지향 프로토콜이므로 프로세스 간 연결이 설정되어야 함

*다중화(Multiplexing)
여러 프로세스의 데이터를 하나의 전송 계층에서 모아 전송하는 과정
*역다중화(Demultiplexing)
수신한 데이터를 목적지 포트 번호에 따라 해당 프로세스로 전달하는 과정

TCP 개요

- TCP 서비스 (5/5)
  - 연결 지향 서비스(Connection-Oriented Service)
    - 데이터 전송 전에 송신 측과 수신 측 사이에 가상 연결을 설정해야 함
    - 연결 설정 → 데이터 전송 → 연결 종료의 과정으로 통신함
  - 신뢰성 서비스(Reliable Service)
    - 데이터가 오류 없이 정상적으로 도착했는지 확인하기 위해 확인응답(ACK) 메커니즘을 사용함

TCP 개요

- TCP 특징 (1/3)
  - 번호화 시스템(Numbering System)
    - TCP는 데이터의 각 바이트에 번호를 부여하여 전송 순서와 수신 여부를 관리함
    - 번호화에는 바이트 번호, 순서 번호(Sequence Number), 확인응답 번호(Acknowledgement Number)가 사용됨
      - 바이트 번호(Byte Number)
        - TCP는 전송되는 모든 데이터 바이트에 번호를 부여함
        - 첫 번째 바이트 번호는 $0 \sim 2^{32} - 1$ 사이에서 임의로 결정됨
      - 순서 번호(Sequence Number)
        - 세그먼트에 포함된 첫 번째 데이터 바이트의 번호를 의미함
      - 확인응답 번호(Acknowledgement Number)
        - 수신 측이 다음에 받기를 기대하는 바이트 번호를 의미함
        - 이전 번호까지의 바이트가 정상적으로 수신되었음을 나타내는 누적 확인응답 방식임

TCP 개요

- TCP 특징 (2/3)
  - 번호화 시스템(Numbering System)
    - 예제 15.1

첫 번째 바이트 번호가 10,001일 때, 3,000 바이트의 데이터를 1,000 바이트의 세그먼트 3개로 보내는 경우, 각 세그먼트의 순서 번호를 구하시오.

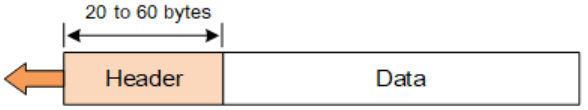
- 세그먼트 1: 순서 번호 10,001 / 데이터 범위: 10,001 ~ 11,000
- 세그먼트 2: 순서 번호 11,001 / 데이터 범위: 11,001 ~ 12,000
- 세그먼트 3: 순서 번호 12,001 / 데이터 범위: 12,001 ~ 13,000

TCP 개요

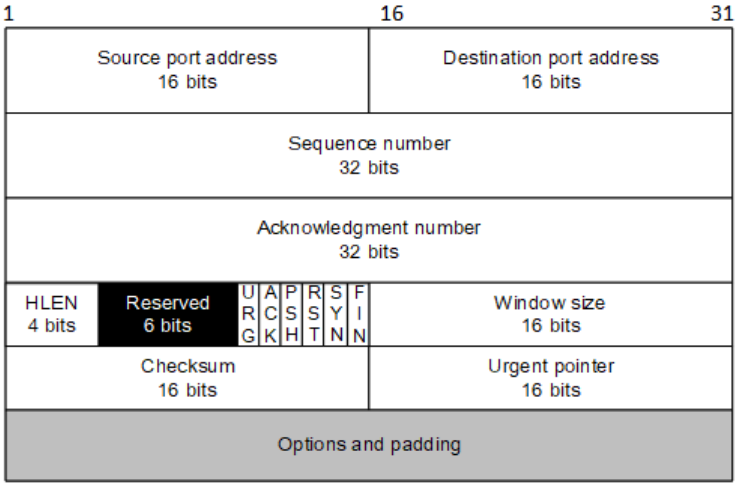
- TCP 특징 (3/3)
  - 흐름 제어(Flow Control)
    - 수신 측의 처리 능력을 초과하지 않도록 송신 데이터의 양을 조절하는 기능
      - e.g., 수신 버퍼의 여유 공간이 부족하여 송신 측의 전송량을 줄여야 하는 상황
  - 오류 제어(Error Control)
    - 데이터가 손실되거나 오류가 발생한 경우 이를 처리하여 신뢰성 있는 데이터 전송을 보장하는 기능
      - e.g., 전송 중 세그먼트가 손실되어 재전송이 필요한 상황
  - 혼잡 제어(Congestion Control)
    - 네트워크의 혼잡 상태에 따라 송신 측의 데이터 전송량을 조절하는 기능
      - e.g ., 네트워크에 트래픽이 몰려 라우터에서 세그먼트가 손실되는 상황

TCP 개요

- 세그먼트 구조 (1/8)
  - TCP 세그먼트는 TCP 헤더 + 응용 프로그램 데이터로 구성됨
  - TCP 헤더의 길이는 20~60바이트임
  - 헤더에는 포트 번호, 순서 번호, 확인응답 번호, 제어 플래그, 윈도우 크기, 체크섬 등의 필드가 포함됨



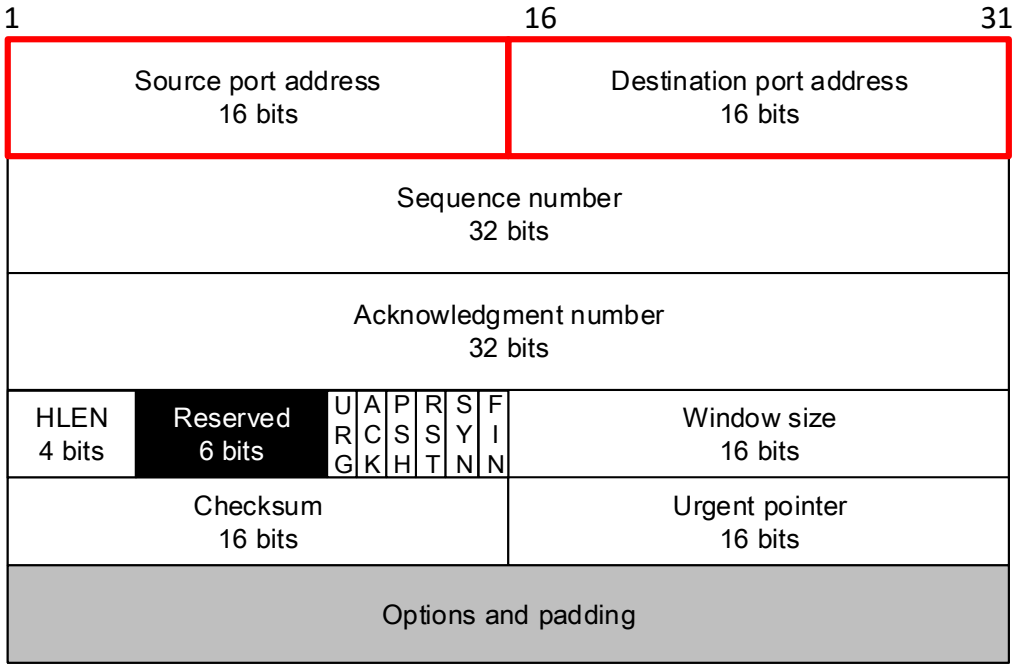
a. Segment



b. Header

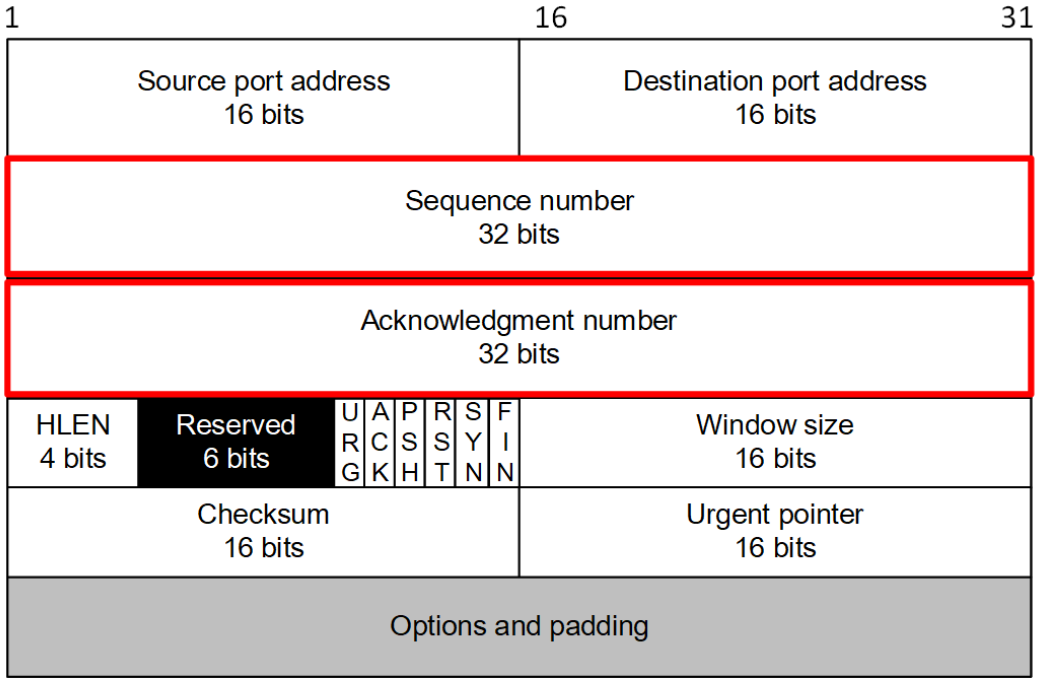
TCP 개요

- 세그먼트 구조 (2/8)
  - 발신지 포트 주소(Source Port Address)
    - 세그먼트를 전송하는 응용 프로그램의 포트 번호
  - 목적지 포트 주소(Destination Port Address)
    - 세그먼트를 수신하는 응용 프로그램의 포트 번호



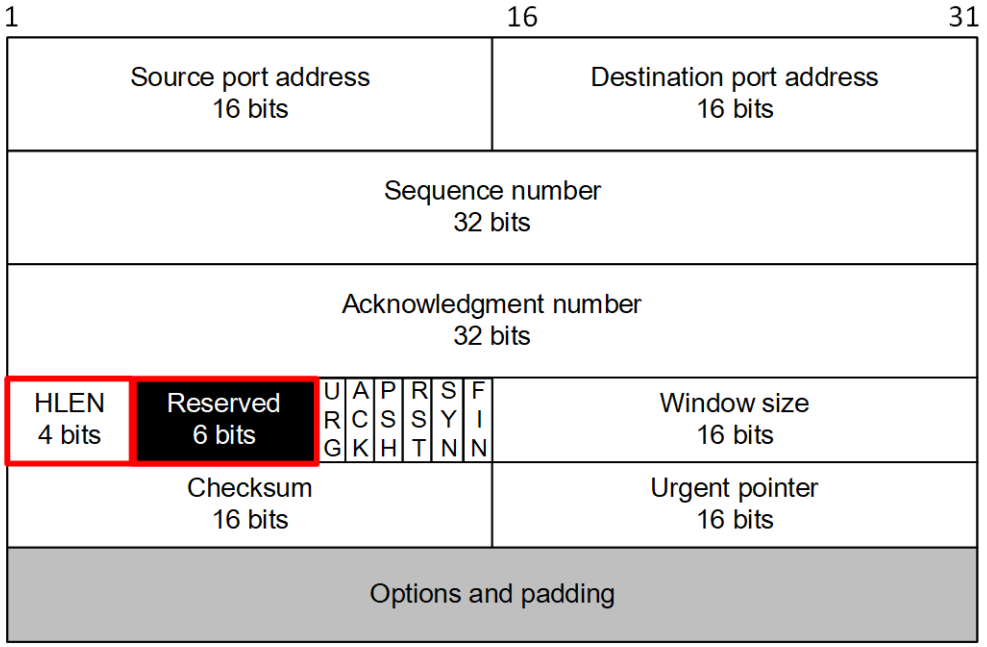
TCP 개요

- 세그먼트 구조 (3/8)
  - 순서 번호(Sequence Number)
    - 세그먼트에 포함된 데이터의 첫 번째 바이트 번호
  - 확인응답 번호(Acknowledgement Number)
    - 수신 측이 다음에 수신하기를 기대하는 바이트 번호



TCP 개요

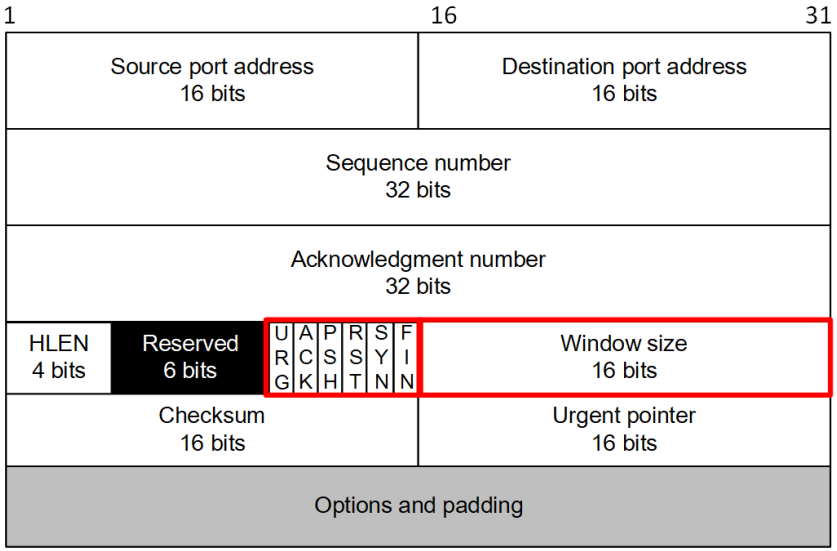
- 세그먼트 구조 (4/8)
  - 헤더 길이(HLEN, Header Length)
    - TCP 헤더의 길이를 나타내는 4비트 필드로, 4바이트 단위로 표현함
  - 예약(Reserved)
    - 향후 사용을 위해 예약된 6비트 필드



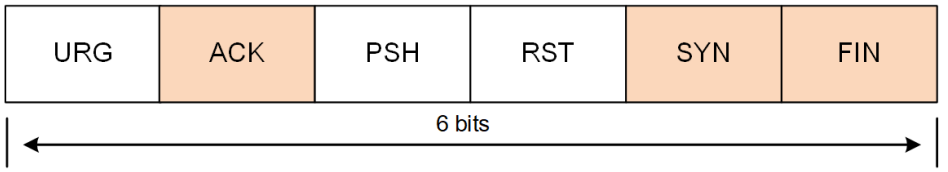
TCP 개요

- 세그먼트 구조 (5/8)
  - 제어(Control)
    - TCP 연결 및 데이터 전송을 제어하기 위해 사용하는 6개의 플래그(Flag) 비트
  - 윈도우 크기(Window Size)
    - 수신 측이 현재 수신할 수 있는 데이터의 양을 나타내며, 송신 측의 전송량을 조절하는 데 사용되는 필드

*플래그(Flag)
TCP가 지금 어떤 동작을 하려는지 알려주는 0/1 신호



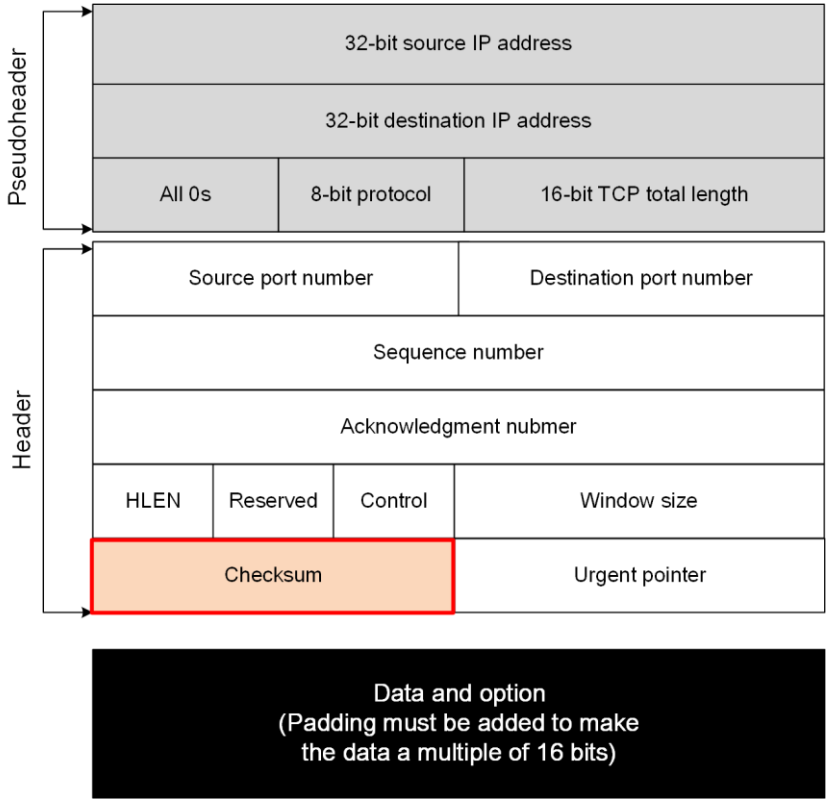
URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



TCP 개요

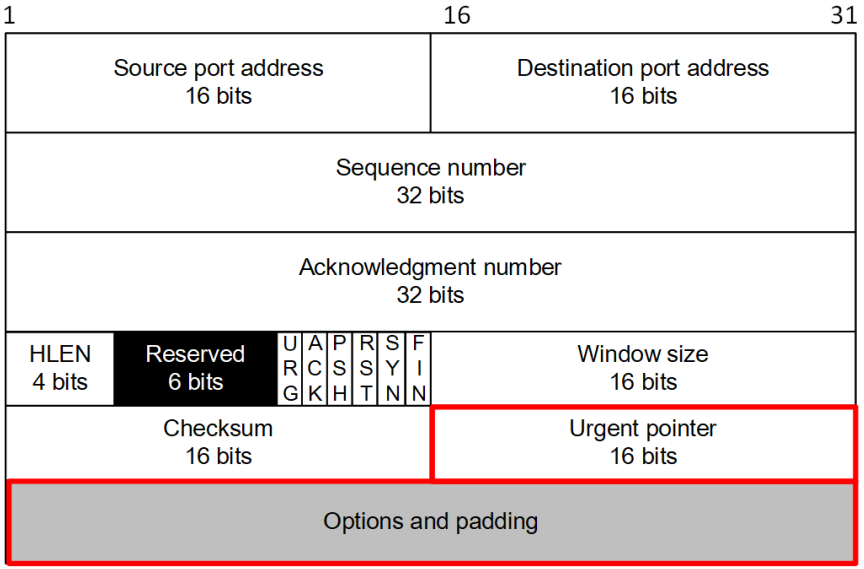
- 세그먼트 구조 (6/8)
  - 체크섬(Checksum)
    - TCP 세그먼트 전송 중 오류가 발생했는지 검사하기 위한 필드
    - 계산 시 IP 주소 등을 포함한 의사 헤더(Pseudo Header)도 함께 사용함

*의사 헤더(Pseudo Header)
TCP 데이터가 올바른 IP 주소 정보와 함께 전달됐는지 까지 확인하기 위해 체크섬 계산에만 임시로 사용하는 정보



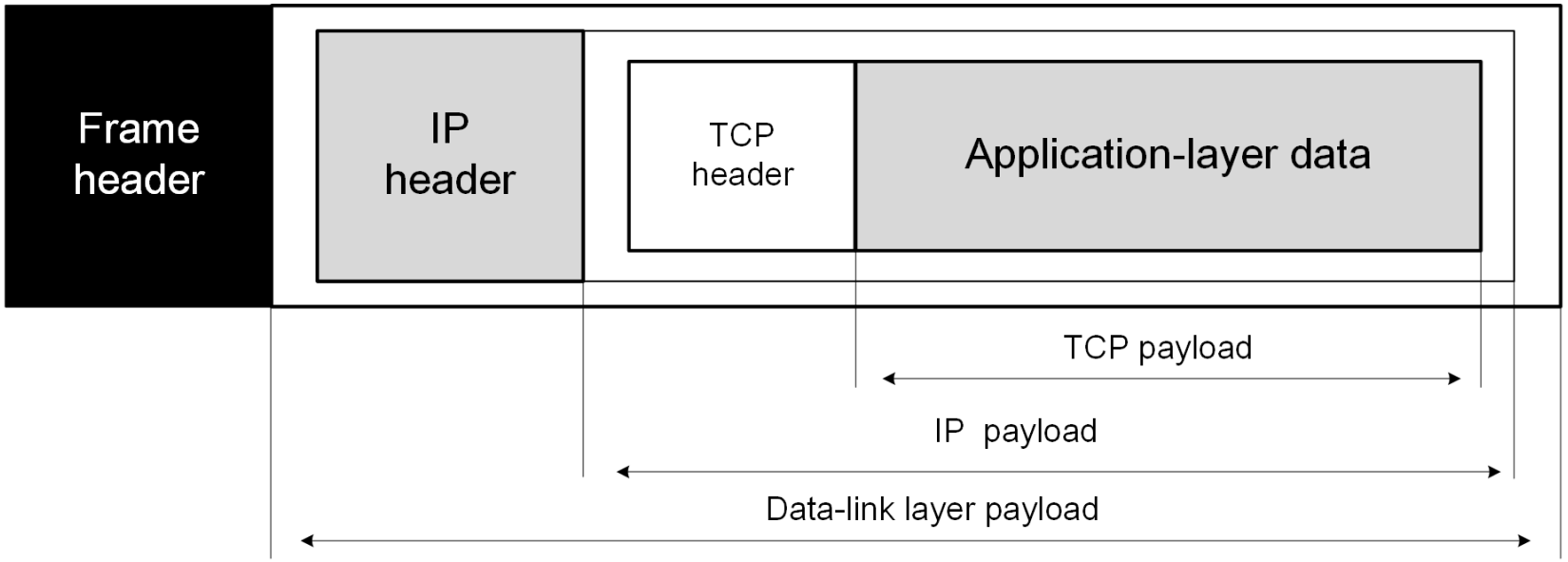
TCP 개요

- 세그먼트 구조 (7/8)
  - 긴급 포인터(Urgent Pointer)
    - 세그먼트 안에서 긴급 데이터의 끝 위치를 나타내는 16비트 필드로, URG 플래그가 설정된 경우에 사용됨
  - 옵션(Option)
    - TCP의 기본 헤더 외에 추가적인 제어 정보를 담는 영역으로, 최대 40바이트까지 사용할 수 있음



TCP 개요

- 세그먼트 구조 (8/8)
  - 캡슐화(Encapsulation)
    - TCP 세그먼트는 응용 계층에서 전달받은 데이터를 캡슐화함
    - 생성된 TCP 세그먼트는 IP 데이터그램에 캡슐화되고, IP 데이터그램은 데이터링크 계층의 프레임에 캡슐화됨



목 차

- TCP 개요
- TCP 연결 및 전송
- TCP 제어
- TCP 동작 메커니즘 및 구현

TCP 연결 및 전송

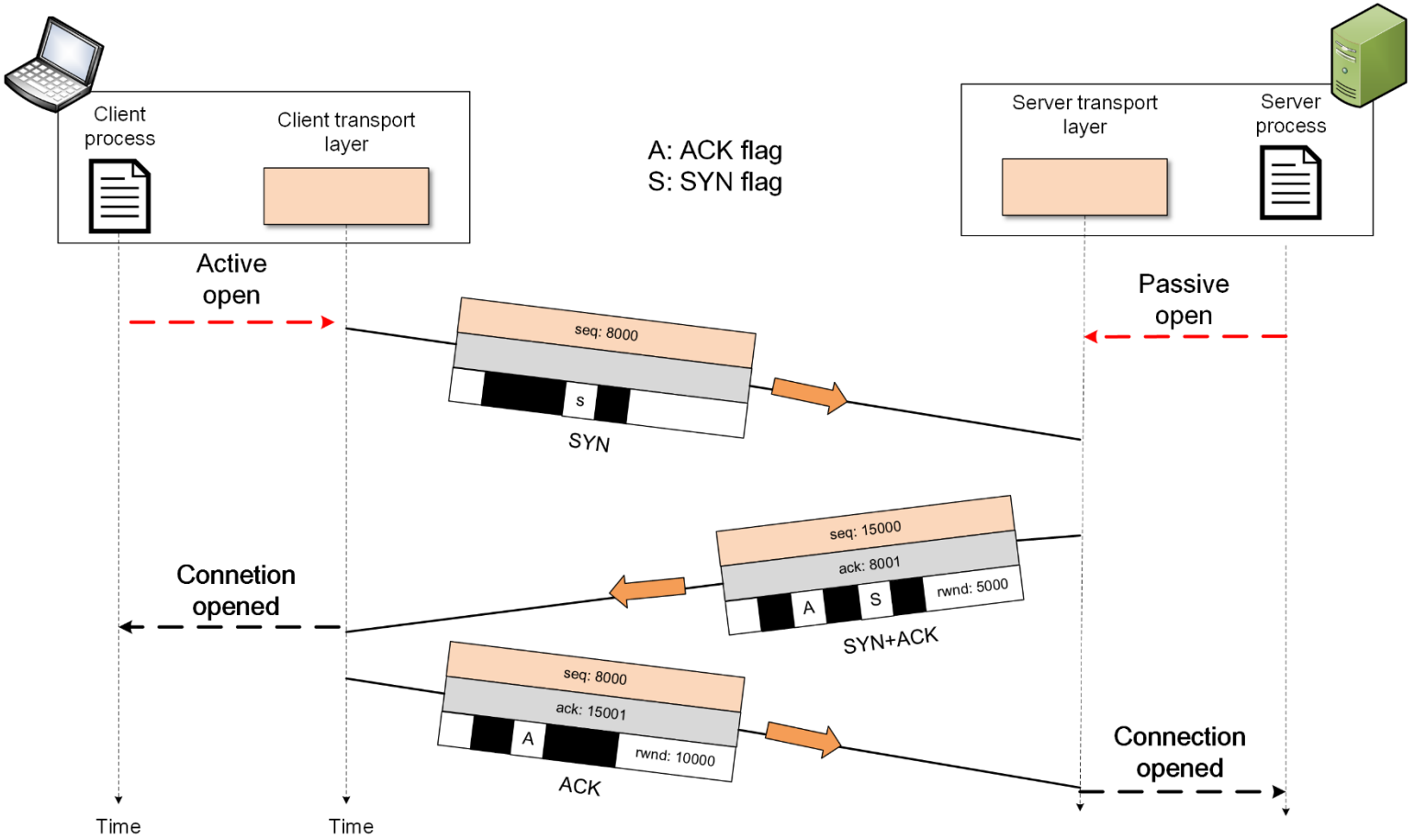
- 개요
  - TCP는 연결 지향 프로토콜로, 데이터 전송 전에 송신 측과 수신 측 사이에 가상 연결을 설정함
  - TCP 통신은 연결 설정 → 데이터 전송 → 연결 종료의 과정으로 이루어짐

TCP 연결 및 전송

- 연결 설정 (1/2)
  - 개방 요청
    - 서버 프로그램은 자신의 TCP에게 Passive Open을 요청하여 클라이언트의 연결 요청을 받을 준비가 되었음을 알림
    - 클라이언트 프로그램은 자신의 TCP에게 Active Open을 요청하여 특정 서버와 연결을 설정하고자 함을 알림
  - 3단계 핸드셰이크(Three-Way Handshake)
    1. 클라이언트는 순서 번호를 동기화하기 위해 초기 순서 번호(ISN, Initial Sequence Number)를 포함한 SYN 세그먼트를 서버에 전송함
    2. 서버는 자신의 ISN과 클라이언트의 SYN에 대한 확인응답 번호를 포함한 SYN+ACK 세그먼트를 전송함
    3. 클라이언트는 서버의 SYN에 대한 확인을 위해 ACK 세그먼트를 전송함

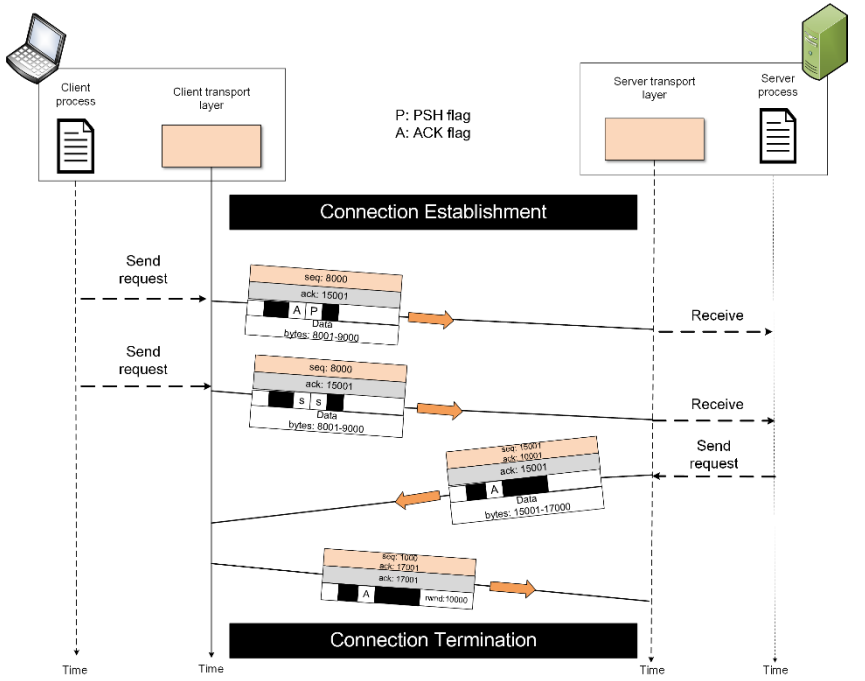
TCP 연결 및 전송

- 연결 설정 (2/2)
  - 3단계 핸드셰이크(Three-Way Handshake)



TCP 연결 및 전송

- 데이터 전송 (1/2)
  - TCP 연결이 설정되면 클라이언트와 서버는 양방향으로 데이터와 확인응답(ACK)을 전송할 수 있음
  - 데이터와 확인응답이 같은 방향으로 전송되는 경우, 하나의 세그먼트에 함께 포함하여 전송할 수 있음
  - 데이터에 확인응답을 함께 실어 보내는 방식을 피기배킹(Piggybacking)이라고 함



TCP 연결 및 전송

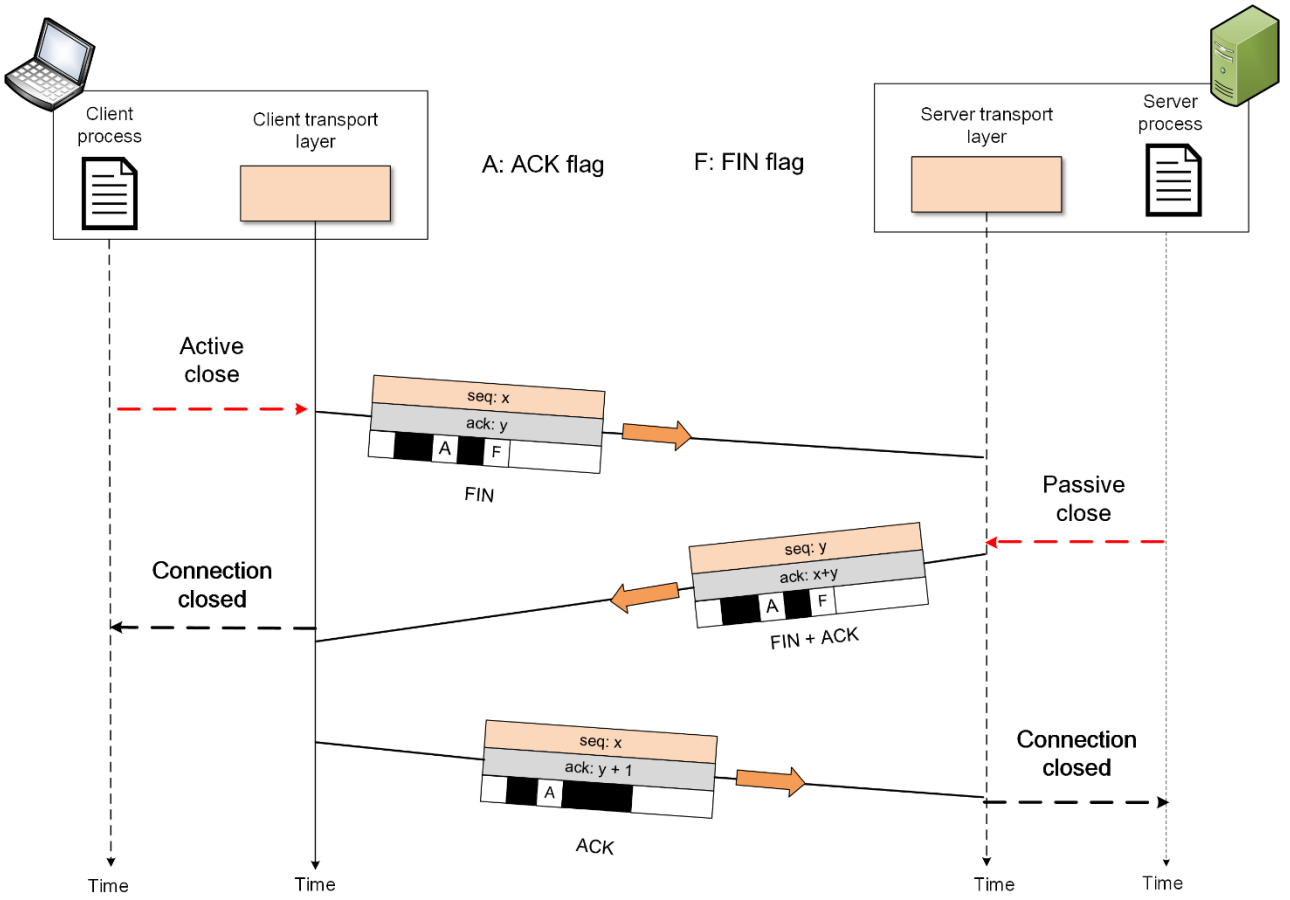
- 데이터 전송 (2/2)
  - 푸싱 데이터(Pushing Data)
    - 버퍼가 채워질 때까지 기다리지 않고 즉시 전송하며 PSH 플래그를 설정함
    - 수신 TCP는 해당 데이터를 가능한 한 빨리 응용 프로그램에 전달함
  - 긴급 데이터(Urgent Data)
    - 특별한 처리가 필요한 데이터에 URG 플래그를 설정하여 전송함
    - Urgent Pointer를 이용해 긴급 데이터의 끝 위치를 나타냄

TCP 연결 및 전송

- 연결 종료 (1/4)
  - 종료 요청
    - 클라이언트 프로그램은 자신의 TCP에게 Active Close를 요청하여 서버와의 연결을 종료할 것임을 알림
    - 서버 프로그램은 자신의 TCP에게 Passive Close를 요청하여 해당 클라이언트와의 연결을 종료할 것임을 알림
  - 3단계 핸드셰이크
    - 클라이언트 TCP가 FIN 세그먼트를 전송함
    - 서버는 FIN에 대한 확인응답과 반대 방향의 종료를 알리기 위해 FIN + ACK 세그먼트를 전송함
    - 클라이언트는 서버의 FIN에 대한 확인응답으로 ACK 세그먼트를 전송함

TCP 연결 및 전송

- 연결 종료 (2/4)
  - 3단계 핸드셰이크
    - 데이터를 포함하지 않는 FIN 세그먼트도 하나의 순서 번호를 소비함



TCP 연결 및 전송

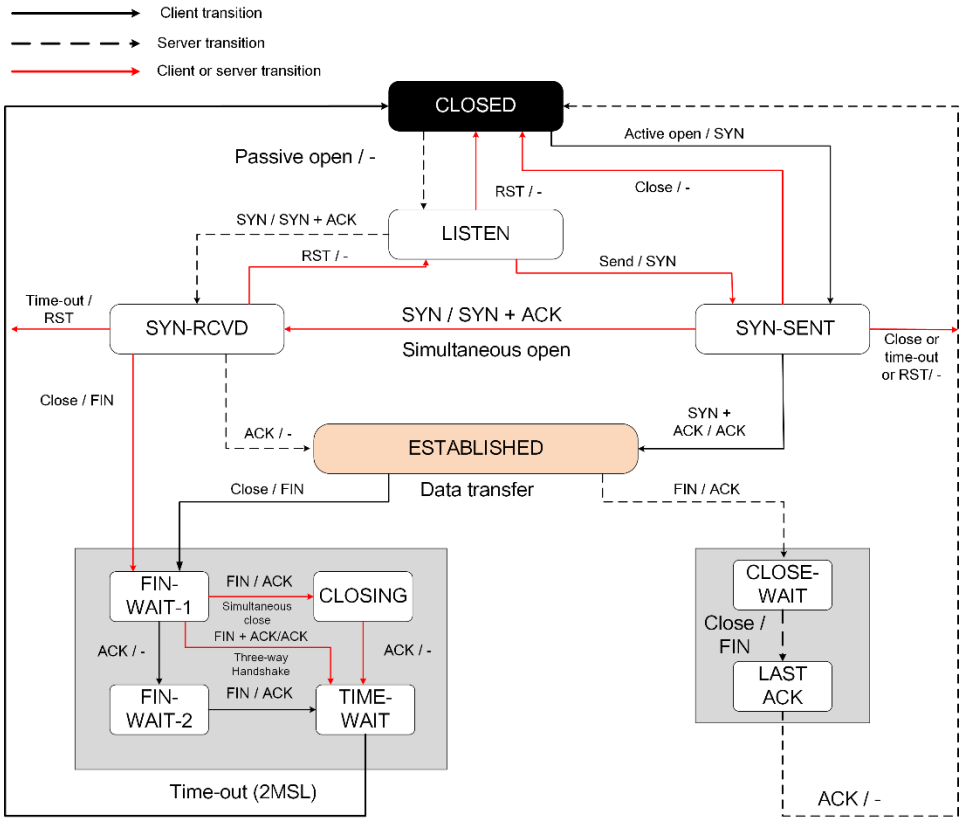
- 연결 종료 (3/4)
  - 반-닫기(Half-Close)
    - 한쪽 방향의 데이터 전송만 종료하고 반대 방향의 데이터 전송은 계속 유지함
    - 먼저 종료하는 측이 FIN을 전송하고, 상대 측은 필요한 데이터 전송을 완료한 후 FIN을 전송하여 연결을 완전히 종료함

TCP 연결 및 전송

- 연결 종료 (4/4)
  - 연결 리셋(Connection Reset)
    - 연결 거절(Denying a Connection)
      - 존재하지 않는 포트 등에 연결 요청이 들어온 경우 RST 세그먼트를 보내 연결 요청을 거절함
    - 연결 중단(Aborting a Connection)
      - 비정상적인 상황이 발생하면 RST 세그먼트를 보내 기존 연결을 즉시 중단함
    - 휴지 연결의 종료(Terminating an Idle Connection)
      - 상대 TCP가 오랫동안 동작하지 않는 상태로 유지되면 RST 세그먼트를 보내 연결을 종료함

TCP 연결 및 전송

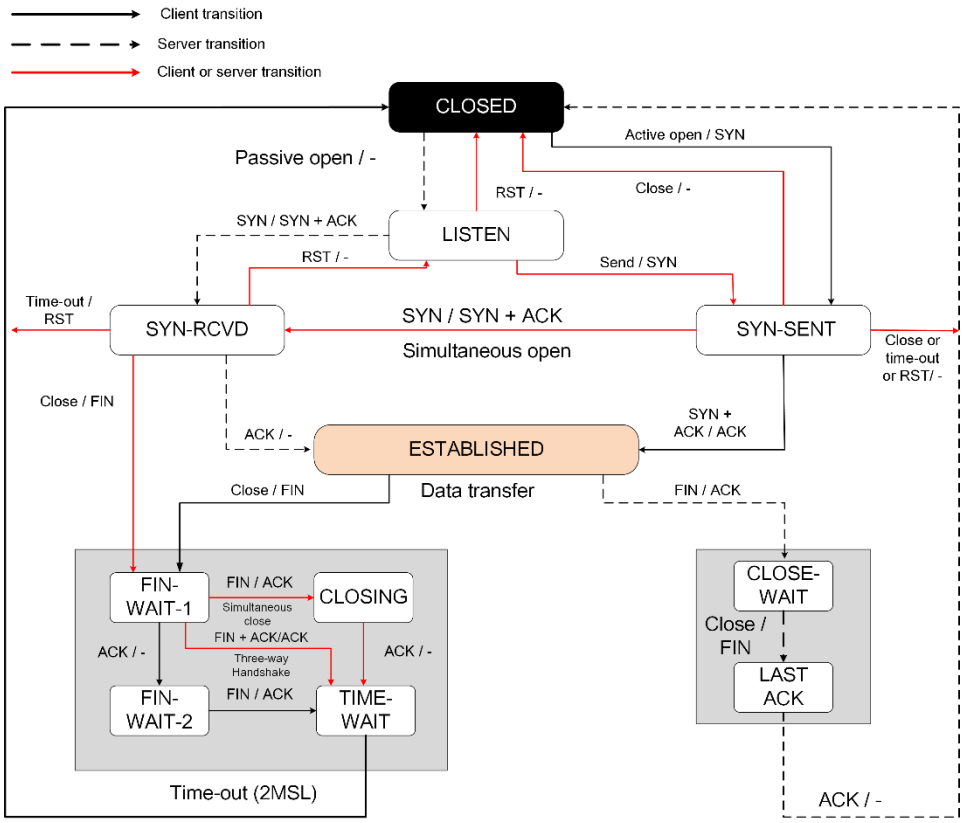
- 상태 전이도(State Transition Diagram) (1/2)
  - 정의
    - TCP의 연결 설정, 데이터 전송, 연결 종료 과정에서 발생하는 상태와 이벤트에 따른 상태 변화를 나타낸 유한 상태 머신



TCP 연결 및 전송

- 상태 전이도(State Transition Diagram) (2/2)
  - 표현
    - 도형 내 단어는 상태를 나타냄

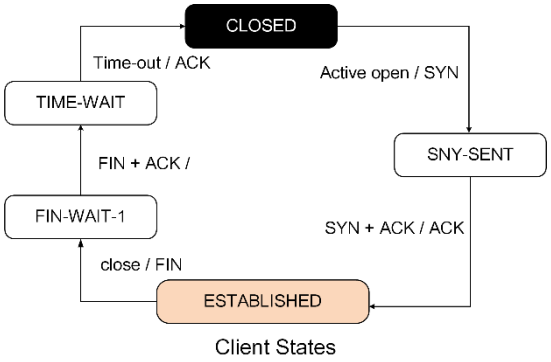
**최대 세그먼트 수명(MSL, Maximum Segment Lifetime)
네트워크 상에서 TCP 세그먼트가 살아남을 수 있는 최대 시간

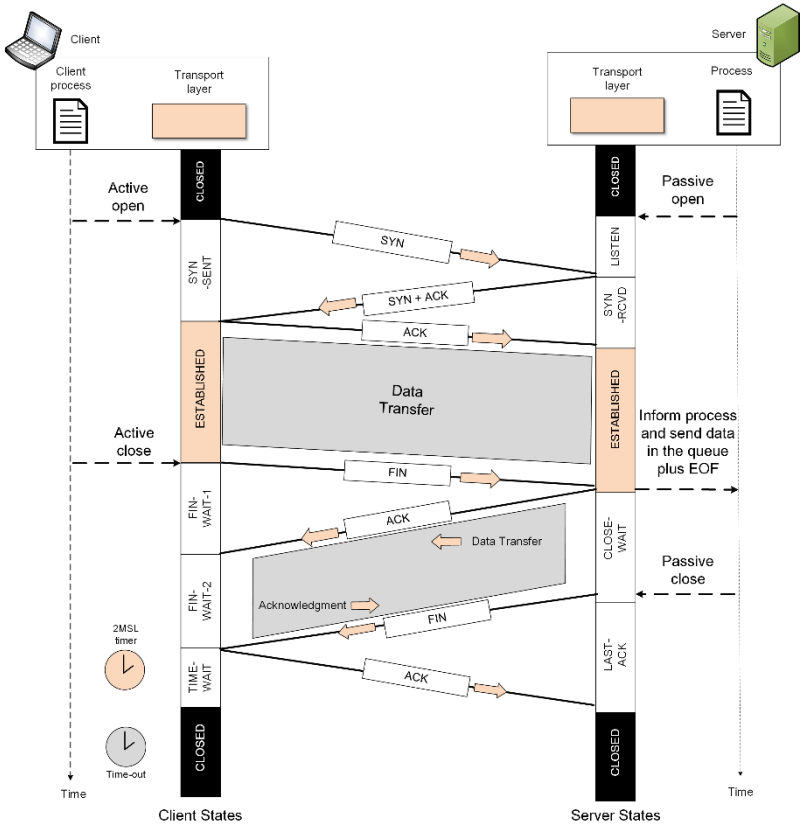


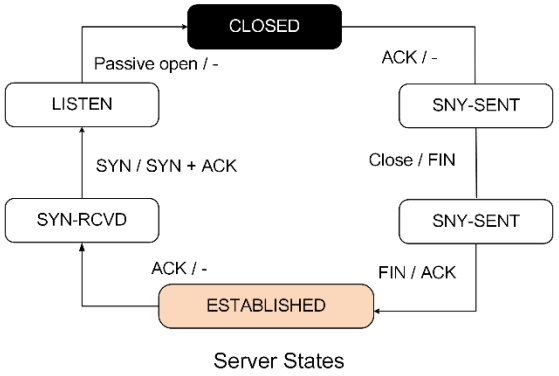
상태	설명
CLOSED	연결이 존재하지 않음
LISTEN	수동 개방 요청을 수신하고 SYN을 기다리고 있음
SYN-SENT	SYN을 전송하고 ACK를 기다리고 있음
SYN-RCVD	SYN+ACK을 전송하고 ACK를 기다리고 있음
ESTABLISHED	연결이 수립되고 데이터 전송 과정을 진행함
FIN-WAIT-1	첫 번째 FIN을 전송하고 ACK를 기다리고 있음
FIN-WAIT-2	첫 번째 FIN에 대한 ACK를 수신하고 두 번째 FIN을 기다리고 있음
CLOSE-WAIT	첫 번째 FIN을 수신받고 ACK를 전송한 후, 연결 종료를 기다리고 있음
TIME-WAIT	두 번째 FIN을 수신받고 ACK를 전송한 후, 2MSL 타임아웃을 기다리고 있음
LAST-ACK	두 번째 FIN을 전송하고 ACK를 기다리고 있음
CLOSING	양측이 동시에 닫기로 결정함

TCP 연결 및 전송

- 연결 및 전송 시나리오 (1/6)
  - 연결 설정 및 반-닫기
    - 한쪽이 먼저 데이터 전송을 종료하지만, 반대 방향의 데이터 전송은 계속 유지해야 하는 경우 발생함

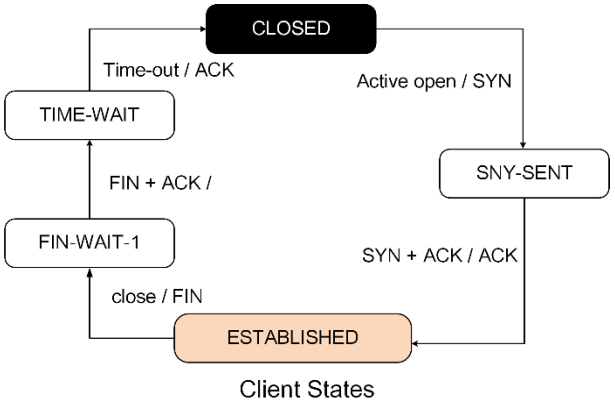


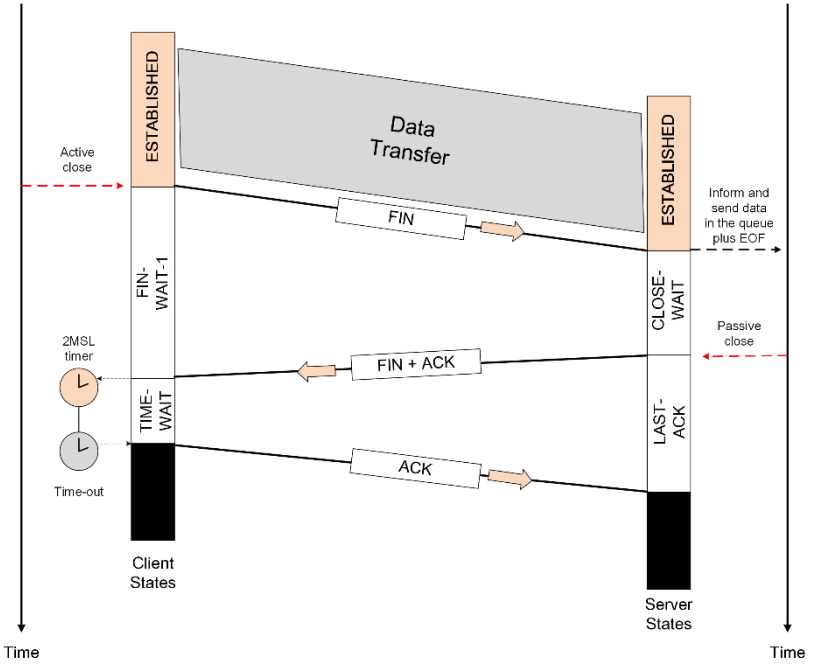


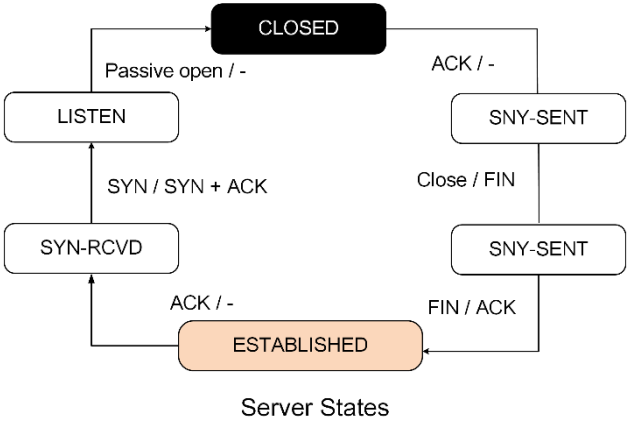


TCP 연결 및 전송

- 연결 및 전송 시나리오 (2/6)
  - 연결 설정 및 3단계 핸드셰이크를 이용한 연결 종료
    - 한쪽의 FIN에 대해 상대 측이 자신의 FIN과 ACK를 함께 전송하는 경우 발생함

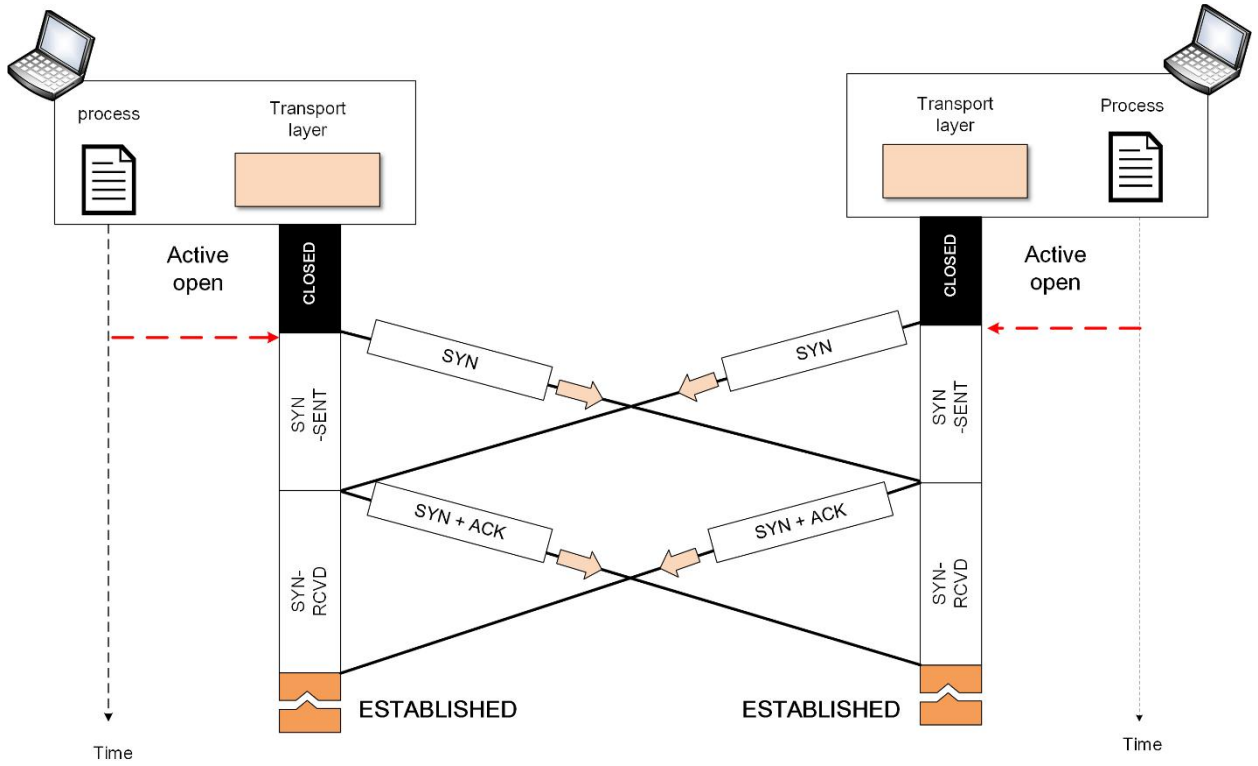






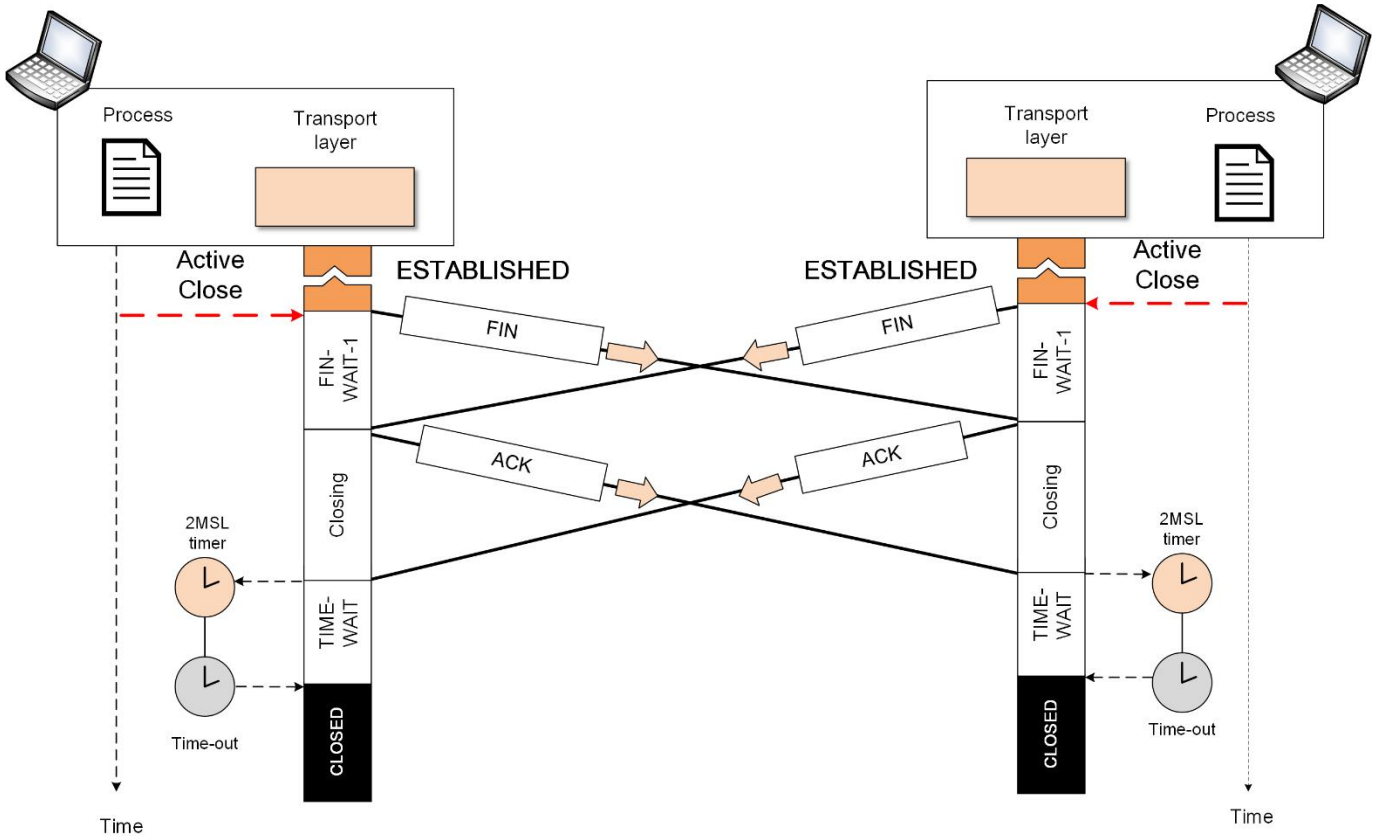
TCP 연결 및 전송

- 연결 및 전송 시나리오 (3/6)
  - 동시 개방
    - 두 호스트가 거의 동시에 Active Open을 수행하는 경우 발생함



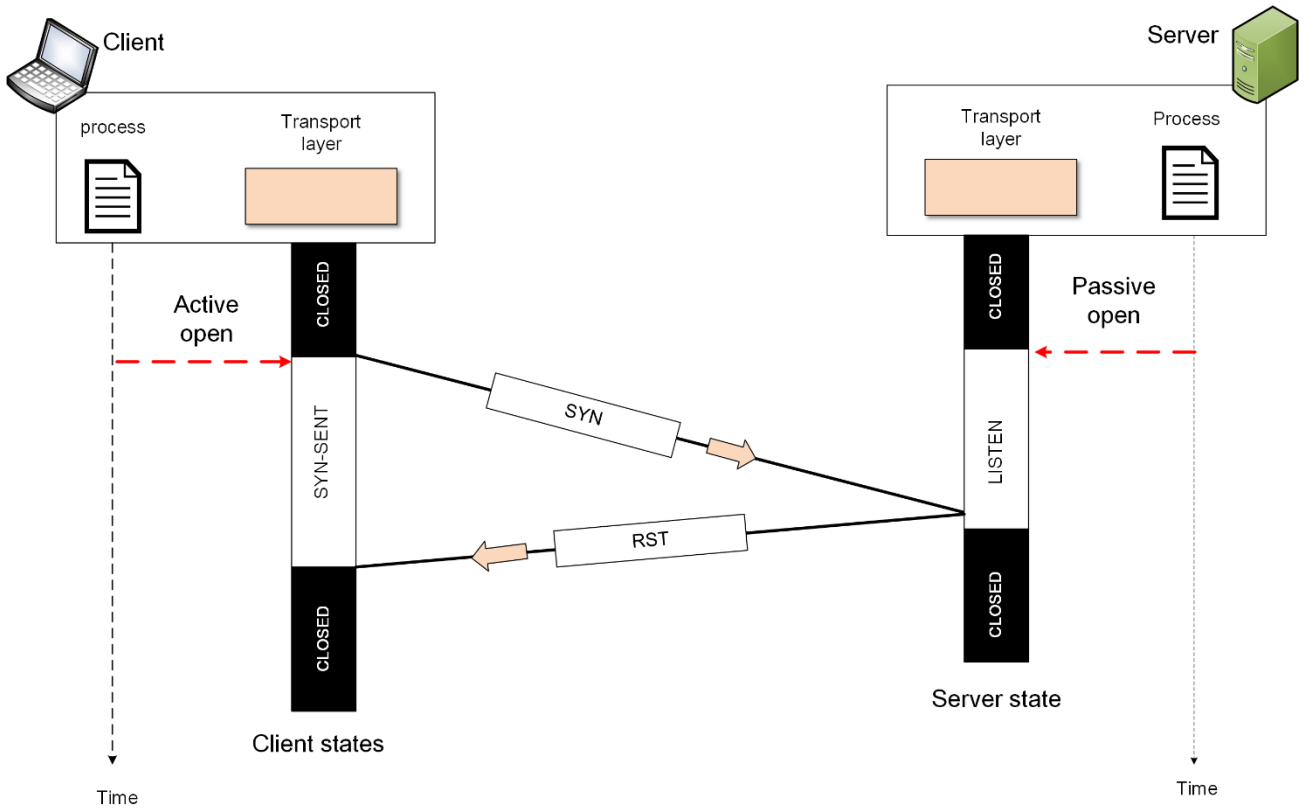
TCP 연결 및 전송

- 연결 및 전송 시나리오 (4/6)
  - 동시 닫기
    - 두 호스트가 동시에 연결 종료를 시도하는 경우 발생함



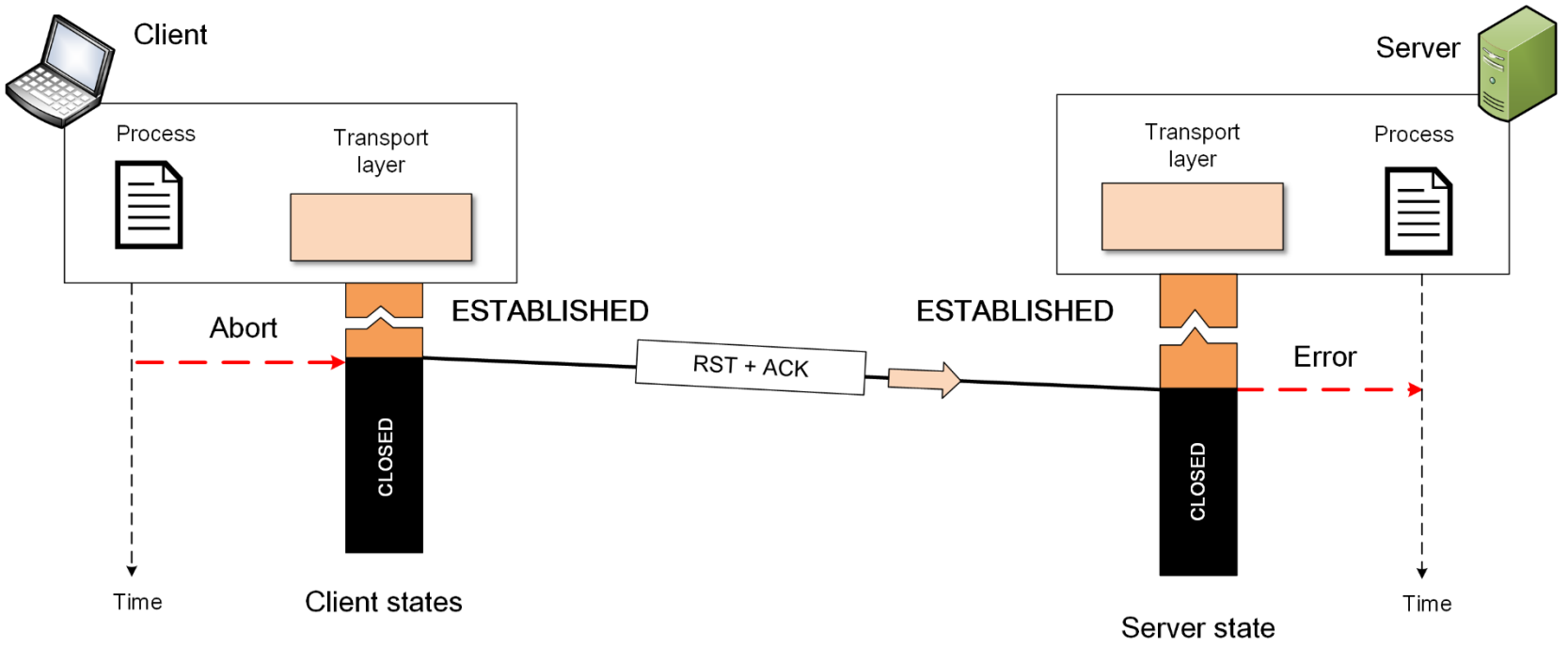
TCP 연결 및 전송

- 연결 및 전송 시나리오 (5/6)
  - 연결 거절
    - 서버 TCP가 클라이언트의 연결 요청을 받아들이지 않는 경우 발생함



TCP 연결 및 전송

- 연결 및 전송 시나리오 (6/6)
  - 연결 중단
    - 루프로 인한 프로세스 고장이나 잘못된 연결의 세그먼트 수신 등 비정상적인 상황에서 발생함



목 차

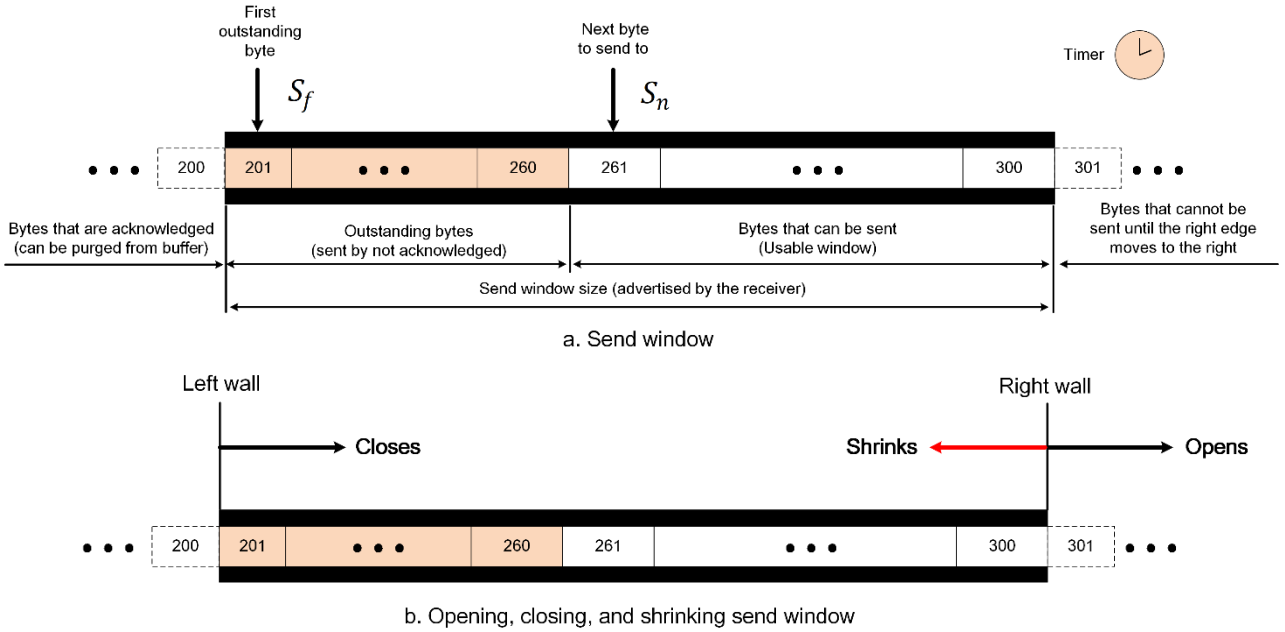
- TCP 개요
- TCP 연결 및 전송
- TCP 제어
- TCP 동작 메커니즘 및 구현

TCP 제어

- TCP 윈도우 (1/2)
  - 정의
    - 송·수신 측 버퍼에서 한 번에 전송하거나 수신할 수 있는 데이터의 범위
  - 특징
    - 클라이언트와 서버가 각각 송신 윈도우와 수신 윈도우를 가짐
    - 세그먼트 단위가 아니라 바이트 단위로 동작함
    - 흐름 제어와 혼잡 제어의 핵심 메커니즘으로 사용됨

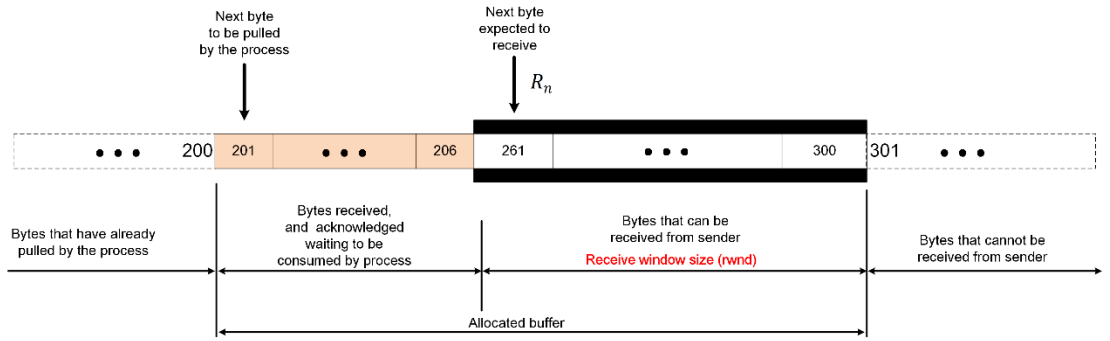
TCP 제어

- TCP 윈도우 (2/2)
  - 송신 윈도우
    - 송신 측이 현재 전송할 수 있는 바이트의 범위를 나타냄
    - 흐름 제어에 따라 윈도우가 열리거나(Open), 닫히거나(Close), 축소(Shrink)될 수 있음
    - 전송한 데이터의 타임아웃을 확인하기 위해 하나의 타이머를 사용함

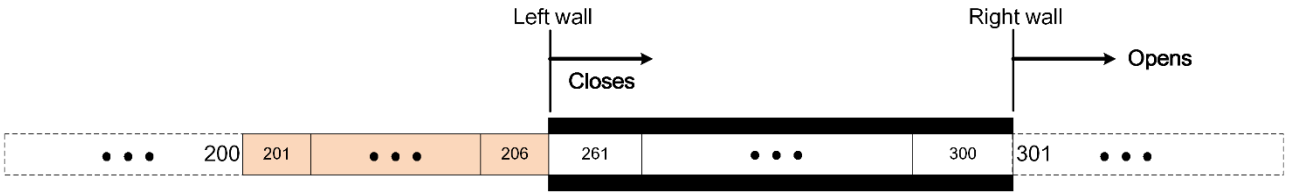


TCP 제어

- TCP 윈도우 (2/2)
  - 수신 윈도우
    - 수신 측이 현재 추가로 받을 수 있는 바이트의 범위를 나타냄
    - 송신 측의 혼란을 방지하기 위해 윈도우 축소는 허용하지 않음
    - 수신 버퍼가 넘치지 않도록, 응용 프로세스가 아직 읽지 않은 데이터 양을 고려하여 크기가 결정됨



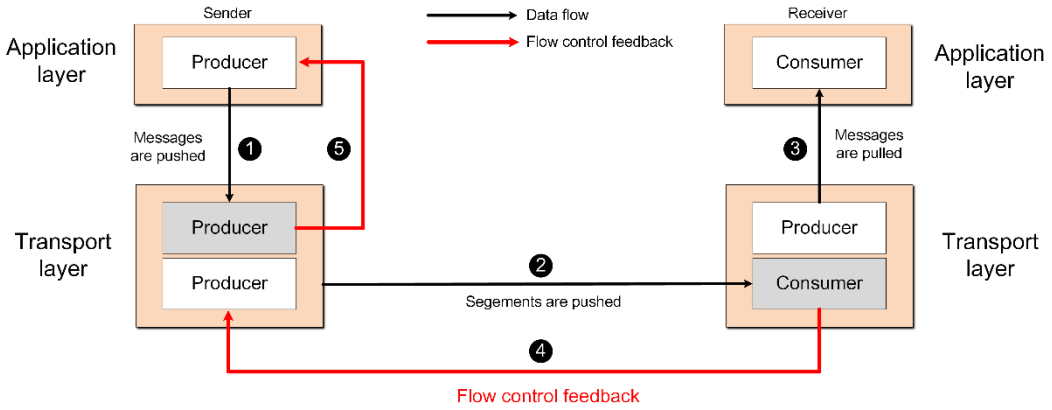
a. Receive window and allocated buffer



b. Opening and closing of receive window

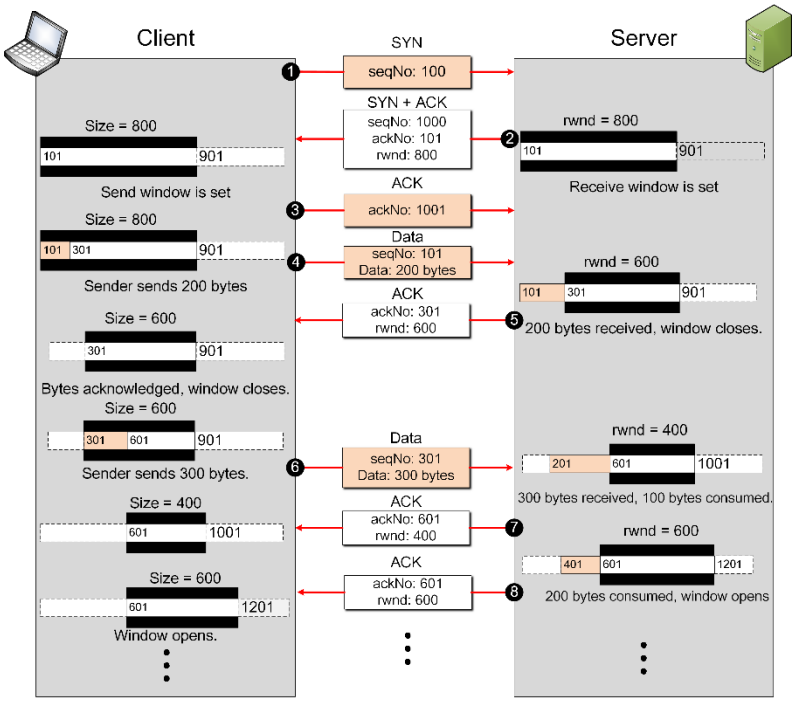
TCP 제어

- 흐름 제어 (1/7)
  - 정의
    - 송신 측이 수신 측의 처리 능력을 초과하는 속도로 데이터를 전송하지 않도록 제어하는 메커니즘
  - 동작
    - 수신 TCP는 ACK 세그먼트를 보낼 때 수신 윈도우(rwnd, Receive Window)를 함께 전달하여 송신 TCP가 전송량을 조절하도록 함
    - 송신 TCP는 송신 버퍼가 가득 차면 송신 프로세스로부터 더 이상 데이터를 받지 않도록 제어함



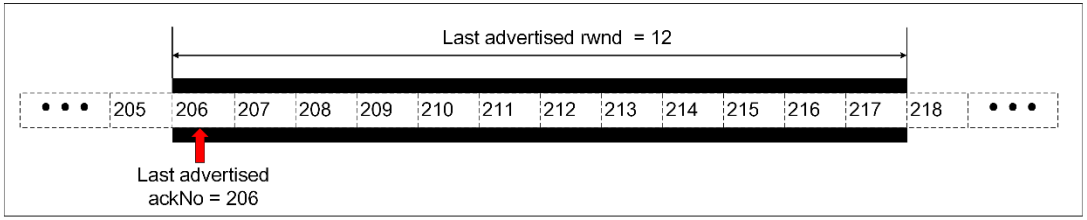
TCP 제어

- 흐름 제어 (2/7)
  - 윈도우 열기 및 닫기
    - 송신 윈도우는 새로운 ACK를 수신하면 앞부분이 닫히고, 수신 측이 광고한 rwnd에 따라 뒤쪽이 열림
    - 수신 윈도우는 데이터를 수신하면 남은 공간이 줄어들어 닫히고, 응용 프로세스가 데이터를 읽으면 공간이 생겨 다시 열림

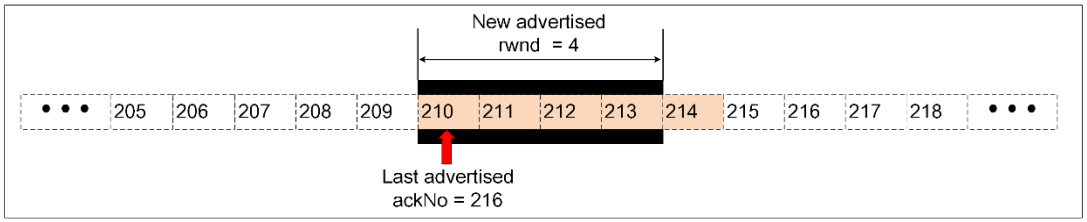


TCP 제어

- 흐름 제어 (3/7)
  - 윈도우 축소 (1/2)
    - 수신 측이 새로운 ACK와 rwnd를 광고하면, 송신 윈도우의 범위가 변경됨
    - 이전보다 작은 rwnd가 광고되면 송신 윈도우의 오른쪽 경계가 왼쪽으로 이동하여 윈도우가 축소될 수 있음
    - 이미 전송 가능하다고 판단한 범위가 줄어들면 송신 측의 전송에 혼란이 발생할 수 있으므로 일반적으로 윈도우 축소는 허용하지 않음



a. The window after the last advertisement



b. The window after the new advertisement; window has shrunk

TCP 제어

- 흐름 제어 (4/7)
  - 윈도우 축소 (2/2)
    - 수신 버퍼가 가득 찬 경우, 예외적으로 rwnd = 0을 광고하여 윈도우를 일시적으로 폐쇄할 수 있음
    - 수신 측이 윈도우를 폐쇄하면 송신 측은 실제 윈도우 크기를 축소하지 않고, 새로운 rwnd가 광고될 때까지 데이터 전송을 중단함
    - 윈도우가 폐쇄된 상태에서도 교착상태(Deadlock)를 방지하기 위해 송신 측은 탐침(Probe) 세그먼트를 전송할 수 있음

*탐침 세그먼트
송신 TCP가 수신 윈도우의 현재 상태를 확인하기 위해 보내는 작은 세그먼트

TCP 제어

- 흐름 제어 (5/7)
  - 어리석은 윈도우 신드롬(SWS, Silly Window Syndrome) (1/3)
    - 정의
      - 흐름 제어 과정에서 송신자와 수신자 사이에 작은 크기의 세그먼트가 불필요하게 자주 전송되는 비효율적인 현상
    - 특징
      - 송신 응용 프로그램이 데이터를 천천히 생성하거나 수신 응용 프로그램이 데이터를 천천히 소비하는 경우 발생할 수 있음
      - 적은 양의 데이터를 포함한 세그먼트가 반복적으로 전송되어 전송 효율이 감소함

TCP 제어

- 흐름 제어 (6/7)
  - 어리석은 윈도우 신드롬(SWS, Silly Window Syndrome) (2/3)
    - 송신 측에서 발생하는 SWS
      - 송신 응용 프로그램이 데이터를 천천히 생성하면, 적은 양의 데이터를 포함한 작은 세그먼트가 반복적으로 전송될 수 있음
    - 해결 방안 – Nagle 알고리즘
      1. 송신 TCP는 처음 전달받은 데이터를 즉시 세그먼트로 만들어 전송함
      2. 이전에 전송한 데이터의 ACK를 기다리는 동안 새로 생성된 작은 데이터를 송신 버퍼에 모아둠
      3. ACK를 수신하거나 최대 세그먼트 크기(MSS)만큼 데이터가 모이면 하나의 세그먼트로 전송함
      4. 작은 세그먼트의 반복적인 전송 횟수를 줄여 전송 효율을 높임

TCP 제어

- 흐름 제어 (7/7)
  - 어리석은 윈도우 신드롬(SWS, Silly Window Syndrome) (3/3)
    - 수신 측에서 발생하는 SWS
      - 수신 응용 프로그램의 데이터 소비 속도가 느려 작은 rwnd가 반복적으로 광고되면서 발생함
    - 해결 방안 – Clark 알고리즘
      - 수신 버퍼에 충분한 공간이 생길 때까지 rwnd = 0을 광고함
      - 최대 크기의 세그먼트를 수용할 수 있거나 수신 버퍼가 절반 이상 비면 새로운 rwnd를 광고함
      - 작은 윈도우가 반복적으로 광고되는 것을 방지하여 작은 세그먼트의 전송을 줄임
    - 해결 방안 – 지연된 확인응답(Delayed ACK)
      - 수신 버퍼에 충분한 공간이 생길 때까지 ACK 전송을 잠시 지연함
      - 수신 응용 프로그램이 데이터를 읽어 충분한 버퍼 공간이 생기면 ACK를 전송함

TCP 제어

- 오류 제어 (1/12)
  - 정의
    - 송신 측의 전체 데이터 스트림이 순서대로, 오류 없이 수신 응용 프로그램에 전달되도록 보장하는 메커니즘
  - 체크섬
    - 세그먼트와 의사 헤더를 이용하여 전송 중 세그먼트가 훼손되었는지 검사함
    - 체크섬 검사 결과 오류가 확인되면 해당 세그먼트를 폐기하고 손실된 것으로 처리함

TCP 제어

- 오류 제어 (2/12)
  - 확인응답(Acknowledgement)
    - 누적 확인응답(Accumulative Acknowledgement)
      - 순서대로 도착한 마지막 데이터까지 정상적으로 수신했음을 ACK 번호로 알림
      - TCP 헤더의 ACK 플래그가 1로 설정된 경우 확인응답 번호가 유효함
    - 선택 확인응답(SACK, Selective Acknowledgement)
      - 순서에 어긋나게 도착한 데이터 블록이나 중복 수신된 블록 정보를 송신 측에 알려줌
      - TCP 헤더의 옵션 영역을 통해 전달함

TCP 제어

- 오류 제어 (3/12)
  - 확인응답 – 전송 규칙 (1/2)
    1. 수신 측에서 보낼 데이터가 있으면 데이터 세그먼트에 ACK 번호를 함께 포함하여 전송함
    2. 보낼 데이터가 없고 순서대로 세그먼트를 수신한 경우, 새 세그먼트가 도착하거나 일정 시간이 지날 때까지 ACK 전송을 잠시 보류할 수 있음
    3. 순서대로 세그먼트를 수신했지만 이전 세그먼트가 아직 확인응답되지 않은 경우 즉시 ACK를 전송함

TCP 제어

- 오류 제어 (4/12)
  - 확인응답 – 전송 규칙 (2/2)
    4. 기대한 것보다 큰 순서 번호의 세그먼트를 수신하면, 현재 기대하는 순서 번호를 ACK 번호로 전송함
    5. 누락된 세그먼트를 수신하면 즉시 ACK를 전송함
    6. 중복 세그먼트를 수신하면 해당 세그먼트를 폐기하고 ACK를 전송함

TCP 제어

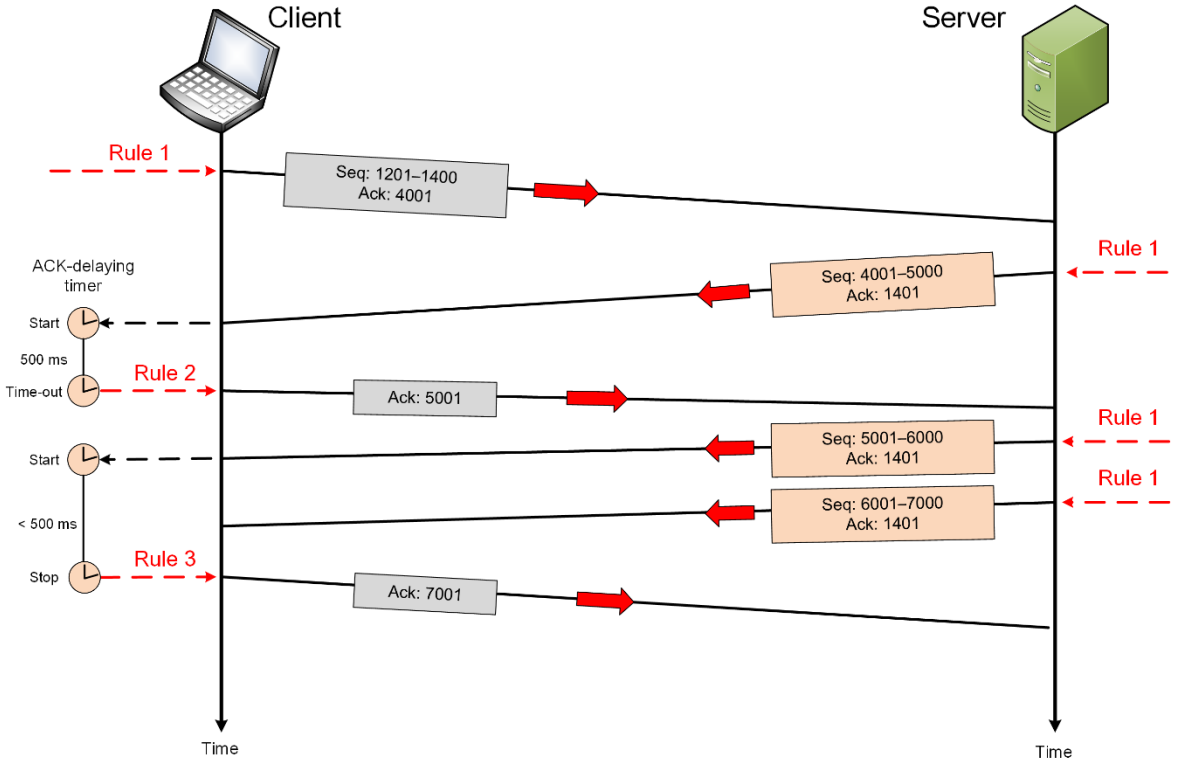
- 오류 제어 (5/12)
  - 재전송
    - 재전송 타임아웃(RTO, Retransmission Time-out)
      - 송신 TCP는 연결마다 하나의 RTO 타이머를 사용함
      - 타이머가 만료되면 버퍼 앞쪽의 세그먼트를 재전송하고 타이머를 다시 시작함
      - RTO 값은 세그먼트의 왕복 시간인 RTT(Round Trip Time)를 기반으로 갱신됨
    - 빠른 재전송(Fast Retransmit)
      - 동일한 세그먼트에 대한 중복 ACK를 3번 수신하면 타임아웃을 기다리지 않고 해당 세그먼트를 즉시 재전송함

TCP 제어

- 오류 제어 (6/12)
  - 순서가 어긋난 세그먼트
    - 순서에 어긋나게 도착한 세그먼트는 즉시 폐기하지 않고 일시적으로 저장함
    - 누락된 세그먼트가 도착할 때까지 해당 세그먼트들을 순서가 어긋난 상태로 유지함
    - 누락된 세그먼트가 도착하여 순서가 맞춰지기 전까지 수신 응용 프로그램에 전달하지 않음

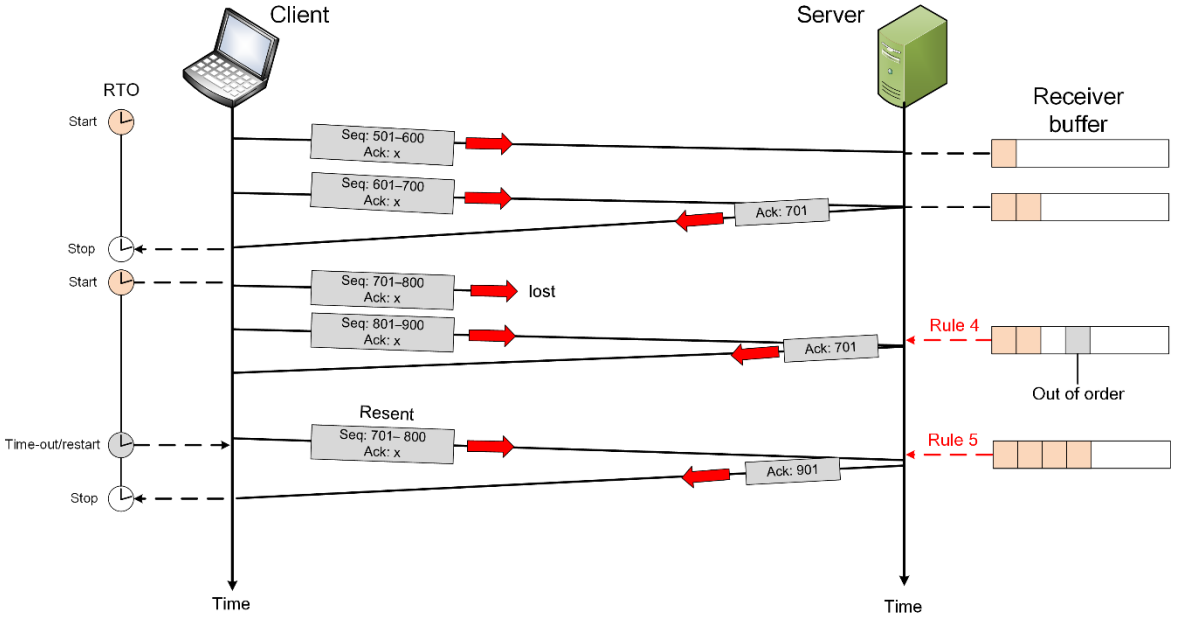
TCP 제어

- 오류 제어 (7/12)
  - 시나리오 (1/6)
    - 정상 동작
      - 수신 측에서 보낼 데이터가 있는 경우, ACK를 별도로 보내지 않고 데이터 세그먼트에 ACK 번호를 함께 포함하여 전송함



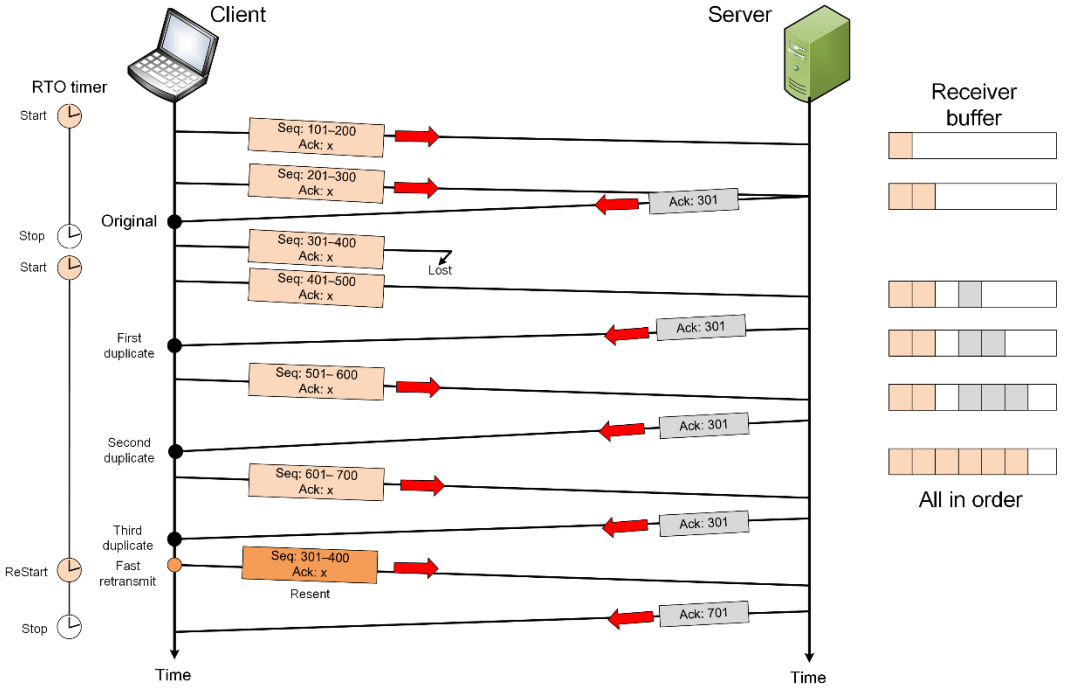
TCP 제어

- 오류 제어 (8/12)
  - 시나리오 (2/6)
    - 손실 세그먼트
      - 기대한 순서보다 큰 순서 번호의 세그먼트를 수신하면, 현재 수신을 기대하는 순서 번호를 ACK 번호로 하여 즉시 전송함
      - 누락되었던 세그먼트가 도착하면, 현재까지 연속적으로 수신된 데이터의 다음 순서 번호를 ACK 번호로 전송함



TCP 제어

- 오류 제어 (9/12)
  - 시나리오 (3/6)
    - 빠른 재전송(Fast Retransmit)
      - 동일한 ACK 번호를 가진 중복 ACK를 3번 수신하면 손실이 발생한 것으로 판단함
      - RTO 타임아웃을 기다리지 않고 해당 세그먼트를 즉시 재전송함

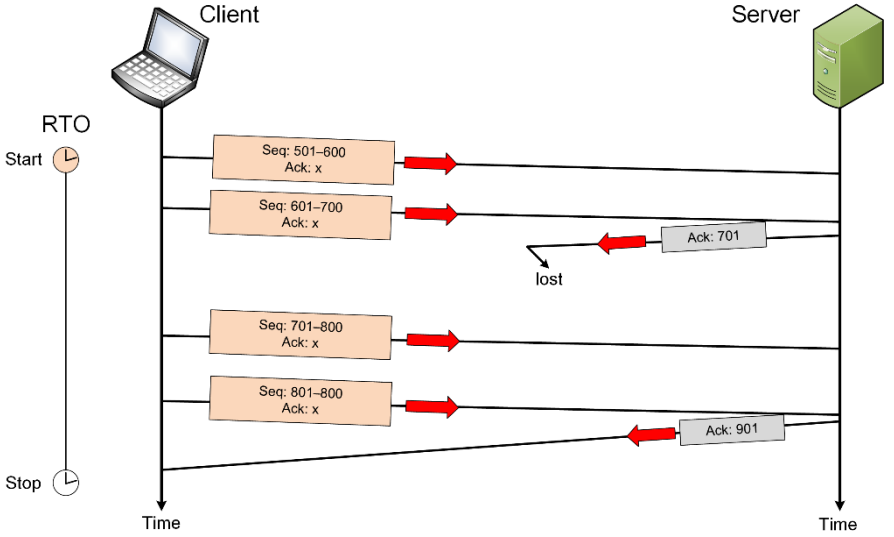


TCP 제어

- 오류 제어 (10/12)
  - 시나리오 (4/6)
    - 지연 세그먼트
      - 네트워크 경로의 지연 차이로 특정 세그먼트가 늦게 도착할 수 있음
      - 송신 측이 이를 손실로 판단해 재전송한 뒤, 원래 세그먼트가 뒤늦게 도착하면 중복 세그먼트가 발생함
    - 중복 세그먼트
      - 같은 데이터가 이미 수신된 상태에서 동일한 세그먼트가 다시 도착한 경우 발생함
      - 수신 측은 중복 세그먼트를 폐기하고, 현재 기대하는 ACK 번호를 가진 ACK를 전송함

TCP 제어

- 오류 제어 (11/12)
  - 시나리오 (5/6)
    - 자동으로 교정되는 ACK의 손실
      - TCP는 누적 확인응답 방식을 사용하므로 손실된 ACK의 정보가 다음 ACK에 포함될 수 있음
      - 다음 ACK가 정상적으로 도착하면 이전 ACK의 손실이 자동으로 보완됨
      - 송신 TCP는 ACK가 하나 손실된 사실을 인지하지 못할 수도 있음



TCP 제어

- 오류 제어 (12/12)
  - 시나리오 (6/6)
    - 확인응답의 손실로 발생하는 교착 상태
      - 수신 측이 rwnd = 0으로 윈도우를 폐쇄한 뒤, 다시 윈도우를 열기 위해 보낸 ACK가 손실되면 교착 상태가 발생할 수 있음
        - 수신 측은 송신 측의 데이터 전송을 기다리고, 송신 측은 새로운 rwnd가 전달되기를 기다리는 상태가 됨
      - 영속 타이머(Persistence Timer)와 탐침(Probe) 세그먼트를 이용해 교착 상태를 방지함
        - 영속 타이머가 만료되면 1바이트의 데이터를 포함한 Probe 세그먼트를 전송함

*영속 타이머(Persistence Timer)
수신 측이 데이터를 다시 받을 수 있는
상태인지 확인하기 위해 사용하는 타이머

TCP 제어

- 혼잡 제어 (1/4)
  - 정의
    - 네트워크 혼잡을 방지하기 위해 송신 측의 전송 속도를 조절하는 메커니즘
  - 혼잡 윈도우(cwnd, Congestion Window)
    - 네트워크가 송신 측의 데이터 발생 속도보다 느리게 데이터를 전달하는 경우, 송신 측의 전송 속도를 조절하는 데 사용함
    - 송신 측은 rwnd와 cwnd를 함께 고려하며, 실제 윈도우 크기는 두 값 중 작은 값으로 결정함

TCP 제어

- 혼잡 제어 (2/4)
  - 혼잡 제어 원칙 (1/3)
    - 느린 시작(Slow Start) : 지수 증가(Exponential Increase)
      - 혼잡 윈도우 cwnd는 하나의 최대 세그먼트 크기인 MSS(Maximum Segment Size)에서 시작함
      - ACK가 도착할 때마다 cwnd를 1 MSS씩 증가시켜 지수적으로 증가함
      - 각 세그먼트에 개별 ACK가 전송되는 경우 cwnd는 RTT마다 1 → 2 → 4 → 8 MSS와 같이 지수적으로 증가함
      - cwnd가 느린 시작 임계치 ssthresh에 도달하면 느린 시작을 종료하고 혼잡 회피 단계로 전환함
      - 지연 ACK를 사용하는 경우 ACK 수가 줄어 증가 속도는 느려질 수 있지만 지수 증가 형태는 유지됨

TCP 제어

- 혼잡 제어 (3/4)
  - 혼잡 제어 원칙 (2/3)
    - 혼잡 회피(Congestion Avoidance) : 가산 증가(Additive Increase)
      - cwnd가 ssthresh에 도달하면 느린 시작 단계를 종료하고 혼잡 회피 단계로 전환함
      - 혼잡 발생 가능성을 줄이기 위해 cwnd의 증가 방식을 지수 증가에서 가산 증가로 변경함
      - 전체 윈도우 크기만큼의 세그먼트가 확인응답될 때마다 cwnd를 1 MSS씩 증가함
      - RTT 기준으로 cwnd가 $i \rightarrow i+1 \rightarrow i+2 \rightarrow i+3$과 같이 선형적으로 증가함

TCP 제어

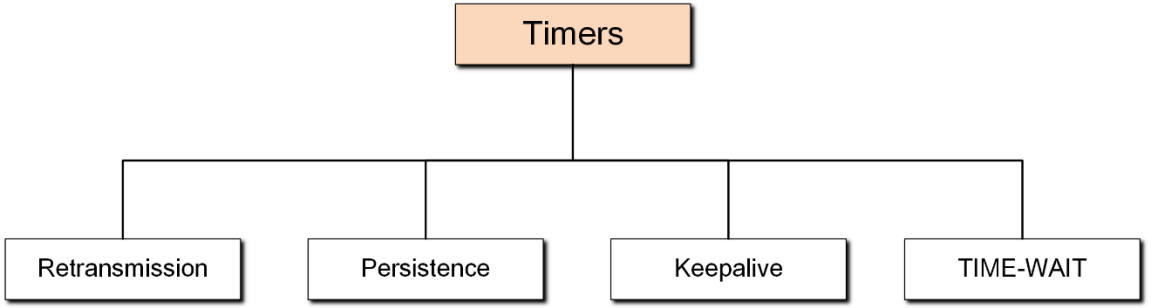
- 혼잡 제어 (4/4)
  - 혼잡 제어 원칙 (3/3)
    - 혼잡 감지(Congestion Detection) : 지수 감소(Multiplicative Decrease)
      - 세그먼트 재전송이 필요한 상황을 혼잡 발생의 신호로 판단함
      - RTO 타임아웃이 발생하면 ssthresh를 현재 윈도우 크기의 절반으로 설정하고 cwnd를 1 MSS로 줄인 후 느린 시작 단계부터 다시 시작함
      - 중복 ACK를 3번 수신하면 ssthresh를 현재 윈도우 크기의 절반으로 설정하고 cwnd를 ssthresh로 설정한 후 혼잡 회피 단계를 시작함
      - RTO 타임아웃은 혼잡 가능성이 더 높은 상황으로 판단하여 더 강하게 전송량을 감소시킴

목 차

- TCP 개요
- TCP 연결 및 전송
- TCP 제어
- TCP 동작 메커니즘 및 구현

TCP 동작 메커니즘 및 구현

- 타이머 (1/9)
  - 정의
    - 특정 이벤트를 일정 시간 내에 처리하지 못했을 때 동작하도록 설정된 시간 측정 메커니즘
  - 종류
    - 재전송 타이머(Retransmission Timer)
    - 영속 타이머(Persistence Timer)
    - 킵얼라이브 타이머(Keepalive Timer)
    - 시간 대기 타이머(TIME-WAIT Timer)



TCP 동작 메커니즘 및 구현

- 타이머 (2/9)
  - 재전송 타이머
    - 정의
      - ACK를 기다리는 시간인 RTO 값으로 설정되는 타이머
    - 동작 과정
      1. 손실된 세그먼트를 재전송하기 위해 사용하는 타이머로, 확인응답을 기다리는 시간인 RTO 값으로 설정함
      2. 송신 버퍼의 맨 앞에 있는 세그먼트를 전송할 때 타이머를 구동함
      3. 타이머가 만료되면 버퍼의 첫 번째 세그먼트를 재전송하고 타이머를 다시 구동함
      4. 누적 확인응답된 세그먼트는 버퍼에서 삭제함
      5. 송신 버퍼가 비어 있으면 타이머를 정지하고, 데이터가 남아 있으면 타이머를 다시 구동함

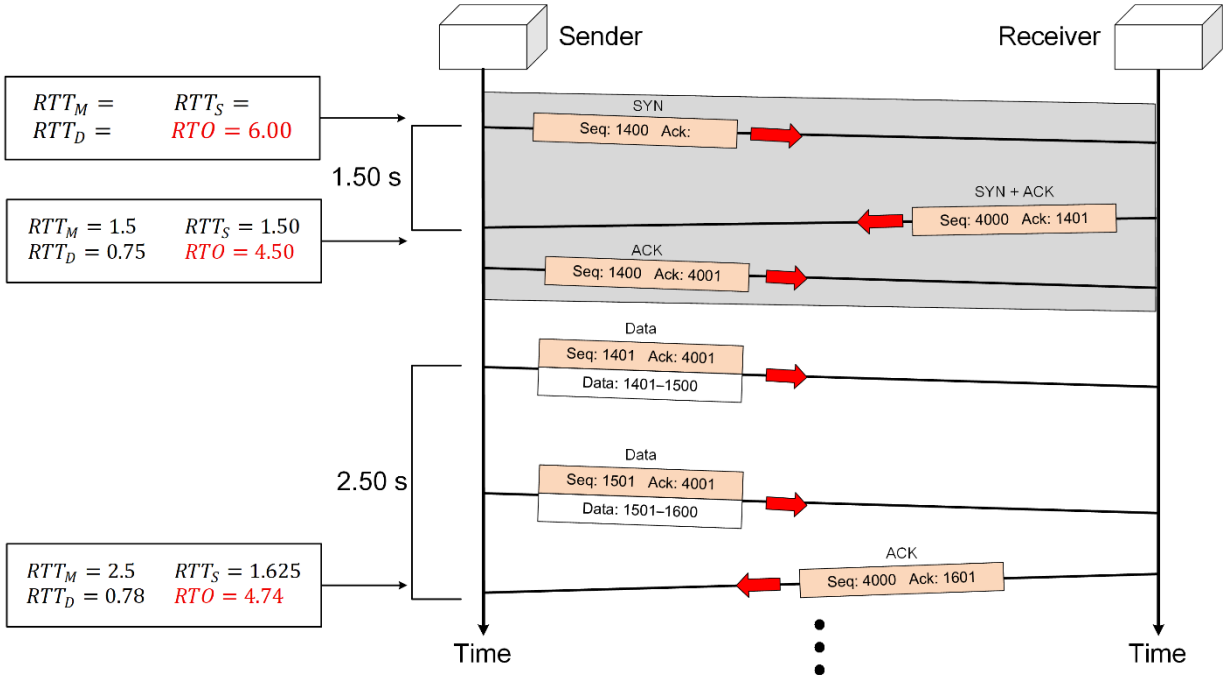
TCP 동작 메커니즘 및 구현

- 타이머 (3/9)
  - 재전송 타이머
    - MRTT(Measured Round Trip Time)
      - 세그먼트를 전송한 시점부터 해당 세그먼트의 ACK를 수신할 때까지 실제로 측정한 왕복 시간
    - SRTT(Smoothed Round Trip Time)
      - 네트워크 상태에 따라 변하는 MRTT 값을 완화하기 위해 이전 SRTT와 새로운 MRTT의 가중 평균을 사용함
      - RTT의 평균적인 수준을 나타내는 값으로 사용함
    - RTTVAR(RTT Variation)
      - RTT가 얼마나 크게 변하는지를 나타내는 값
      - 네트워크 지연의 변동성을 RTO 계산에 반영하기 위해 사용함

TCP 동작 메커니즘 및 구현

- 타이머 (4/9)
  - 재전송 타이머
    - RTO(Retransmission Time-Out)
      - 재전송 타이머에 설정되는 대기 시간을 의미함
      - SRTT, RTTVAR을 기반으로 $RTO = RTT_S + 4 \times RTT_D$로 계산함

*RTT_M: 실제로 측정한 왕복 시간
*RTT_S: 이전 측정값들을 반영한 평균적인 왕복 시간
*RTT_D: 왕복 시간이 얼마나 불규칙하게 변하는지 나타내는 값
*RTO: ACK가 오지 않을 때 재전송하기까지 기다리는 시간



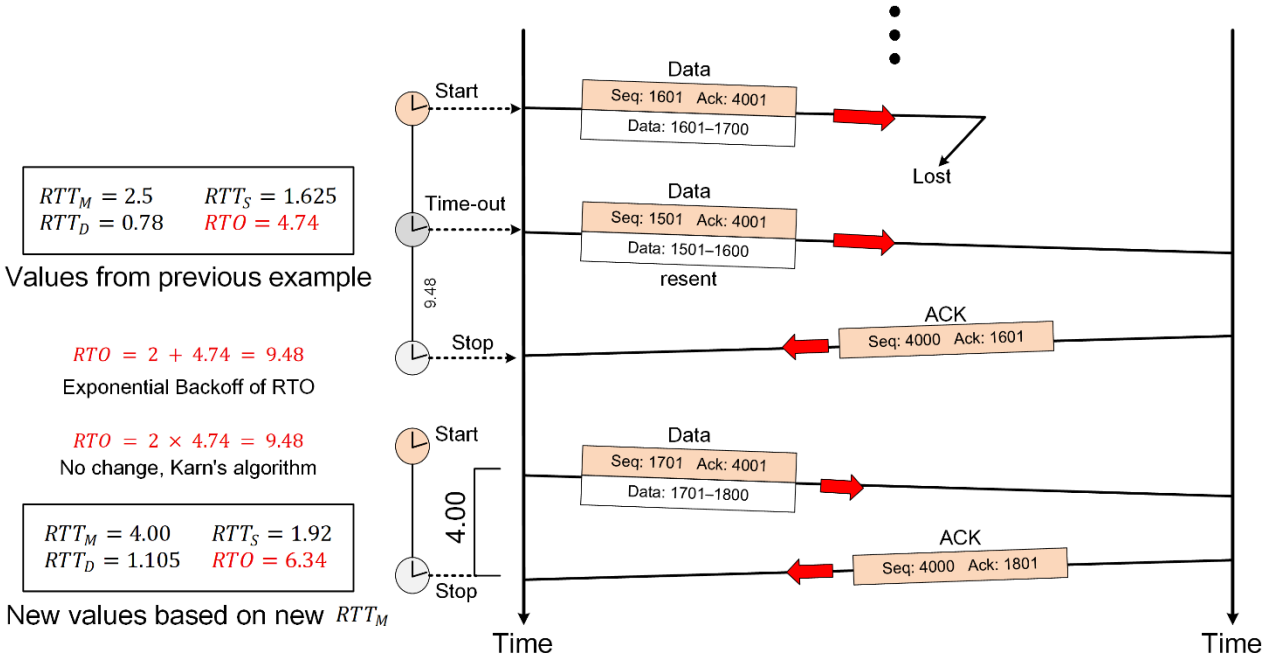
TCP 동작 메커니즘 및 구현

- 타이머 (5/9)
  - 재전송 타이머
    - Karn 알고리즘
      - TCP는 측정한 RTT를 기반으로 SRTT와 RTTVAR를 갱신하여 RTO를 계산함
      - 세그먼트가 재전송되면 수신한 ACK가 최초 전송과 재전송 중 어느 것에 대한 응답인지 구분하기 어려움
      - 부정확한 RTT가 RTO 계산에 반영되는 것을 방지하기 위해 Karn 알고리즘을 사용함
        - 재전송된 세그먼트의 RTT는 새로운 RTT 계산에 사용하지 않음
        - 재전송 없이 정상적으로 ACK를 수신한 경우에만 RTT를 갱신함

TCP 동작 메커니즘 및 구현

- 타이머 (6/9)
  - 재전송 타이머
    - 지수 백오프(Exponential Backoff)
      - 세그먼트가 재전송될 때마다 RTO 값을 두 배로 증가시킴
      - 한 번 재전송하면 기존 RTO의 2배, 두 번 재전송하면 4배로 설정함
      - 재전송이 반복될수록 RTO가 지수적으로 증가함

*RTT_M: 실제로 측정한 왕복 시간
*RTT_S: 이전 측정값들을 반영한 평균적인 왕복 시간
*RTT_D: 왕복 시간이 얼마나 불규칙하게 변하는지 나타내는 값
*RTO: ACK가 오지 않을 때 재전송하기까지 기다리는 시간



TCP 동작 메커니즘 및 구현

- 타이머 (7/9)
  - 영속 타이머(Persistence Timer)
    - 정의
      - 수신 TCP가 윈도우 크기를 0으로 알린 상황에서 발생할 수 있는 교착 상태를 해결하기 위해 사용하는 타이머
    - 동작 과정
      1. 송신 TCP가 윈도우 크기 0인 ACK를 수신하면 영속 타이머를 시작함
      2. 타이머가 만료되면 1바이트의 새로운 데이터를 포함한 Probe 세그먼트를 전송함
      3. 응답이 없으면 Probe를 다시 전송하고 영속 타이머 값을 두 배로 증가시켜 재설정함
      4. 타이머 값이 임계치에 도달하면 윈도우가 다시 열릴 때까지 60초마다 Probe 세그먼트를 전송함

TCP 동작 메커니즘 및 구현

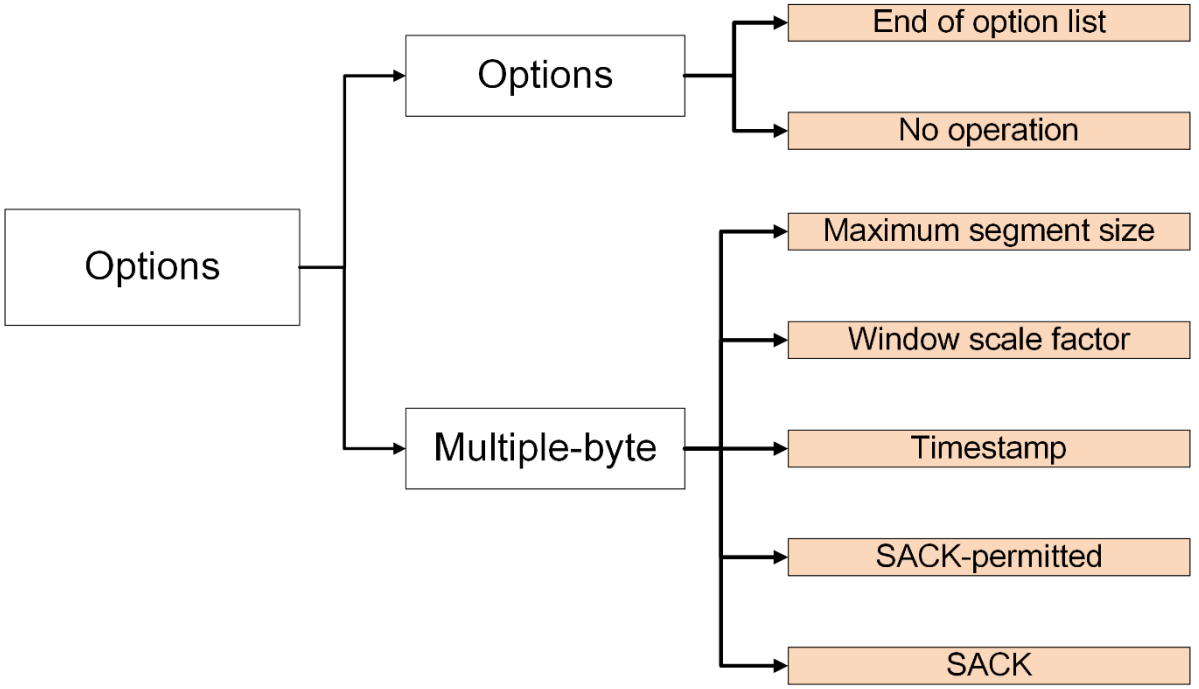
- 타이머 (8/9)
  - 킵얼라이브 타이머(Keepalive Timer)
    - 정의
      - TCP 연결이 오랫동안 아무 통신 없이 열린 상태로 유지되는 것을 방지하기 위해 사용하는 타이머
    - 동작 과정
      1. 서버가 클라이언트로부터 세그먼트를 수신할 때마다 타이머를 다시 설정함
      2. 일정 시간 동안 세그먼트를 받지 못하면 Probe 세그먼트를 전송함
      3. 여러 번 Probe를 전송해도 응답이 없으면 클라이언트가 동작하지 않는 것으로 판단하고 연결을 종료함

TCP 동작 메커니즘 및 구현

- 타이머 (9/9)
  - 시간 대기 타이머(TIME-WAIT Timer, 2MSL Timer)
    - 정의
      - TCP 연결 종료 과정에서 TIME-WAIT 상태 동안 사용하는 타이머
    - 필요성
      - 마지막 ACK가 손실되어 상대 TCP가 FIN을 재전송하는 경우 다시 ACK를 전송할 수 있도록 일정 시간 연결 상태를 유지함
      - 이전 연결의 중복 세그먼트가 동일한 소켓 주소를 사용하는 새로운 연결에 전달되는 것을 방지함

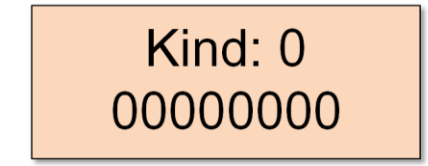
TCP 동작 메커니즘 및 구현

- 옵션 (1/9)
  - 개요
    - TCP 헤더에 포함할 수 있는 최대 40바이트의 선택적 정보
    - 목적지에 추가 정보를 전달하거나 다른 옵션의 정렬을 맞추기 위해 사용함

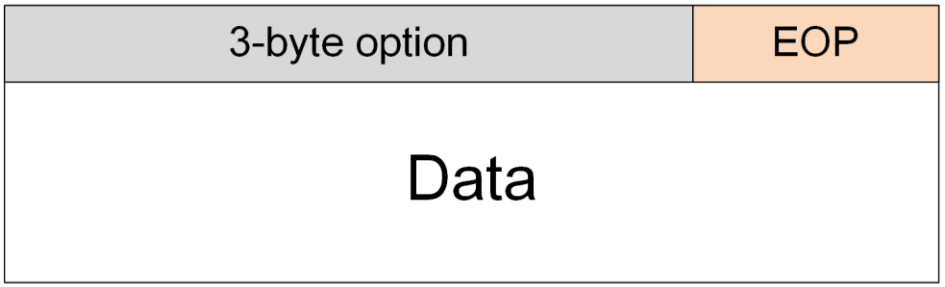


TCP 동작 메커니즘 및 구현

- 옵션 (2/9)
  - 단일 바이트 옵션(Single-byte Options) (1/2)
    - 옵션 종료(EOP, End-of-Option)
      - 옵션 영역의 끝에서 패딩을 위해 사용하는 1바이트 옵션
      - 옵션 영역의 마지막 옵션으로만 사용할 수 있으며 한 번만 사용할 수 있음
      - EOP 뒤의 응용 프로그램 데이터가 다음 32비트 워드 경계에서 시작하도록 정렬함



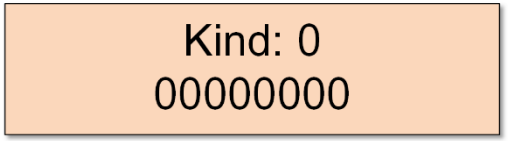
a. End of option list



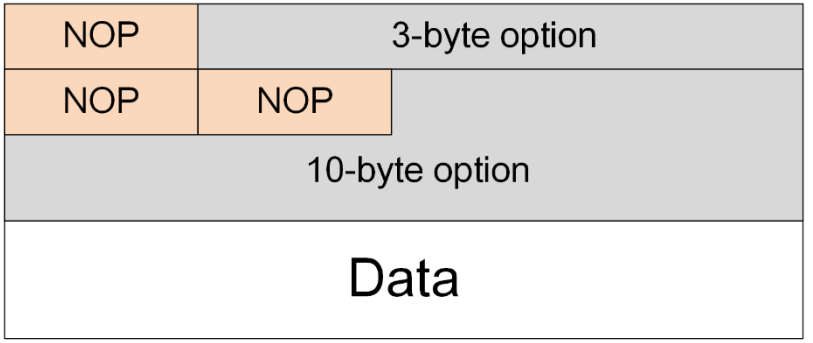
b. Used for padding

TCP 동작 메커니즘 및 구현

- 옵션 (3/9)
  - 단일 바이트 옵션(Single-byte Options) (2/2)
    - 무동작(NOP, No-Operation)
      - 옵션 사이의 정렬을 맞추기 위해 사용하는 1바이트 채우기(Filler) 옵션
      - 보통 다른 옵션 앞에 위치하여 해당 옵션이 32비트 워드 경계에 맞도록 정렬함



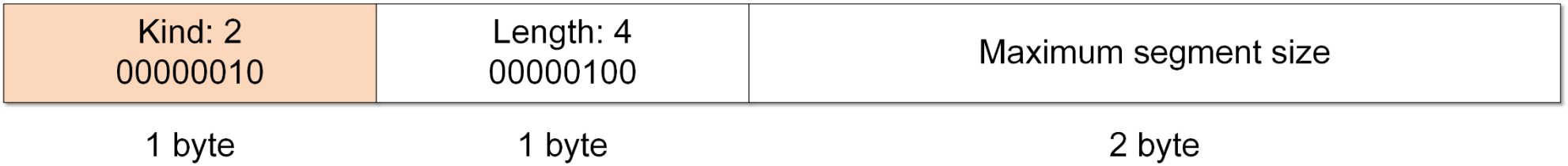
a. No operation option



b. Used to align beginning of an option

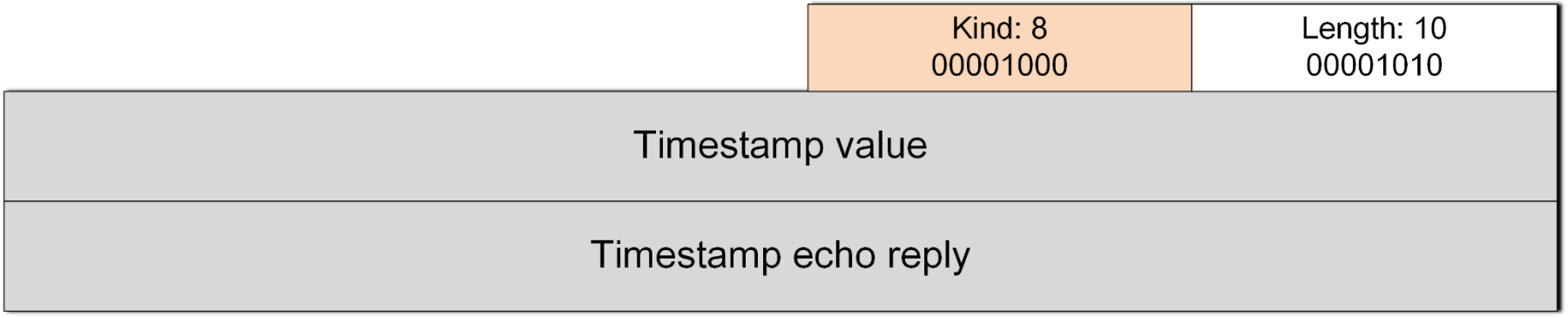
TCP 동작 메커니즘 및 구현

- 옵션 (4/9)
  - 다중 바이트(Multiple-byte) 옵션 (1/6)
    - 최대 세그먼트 크기(MSS, Maximum Segment Size)
      - 목적지 TCP가 수신할 수 있는 데이터 부분의 최대 크기를 나타내는 옵션
      - 이름과 달리 TCP 세그먼트 전체 크기가 아니라 데이터의 최대 크기를 의미함
      - 16비트 필드로 표현되며 값의 범위는 0~65,535바이트임
      - MSS 값은 연결 설정 과정에서 결정되며, 각 TCP는 자신이 수신할 세그먼트의 MSS를 지정함
      - MSS를 지정하지 않은 경우 교재에서는 기본값을 536바이트로 설명함



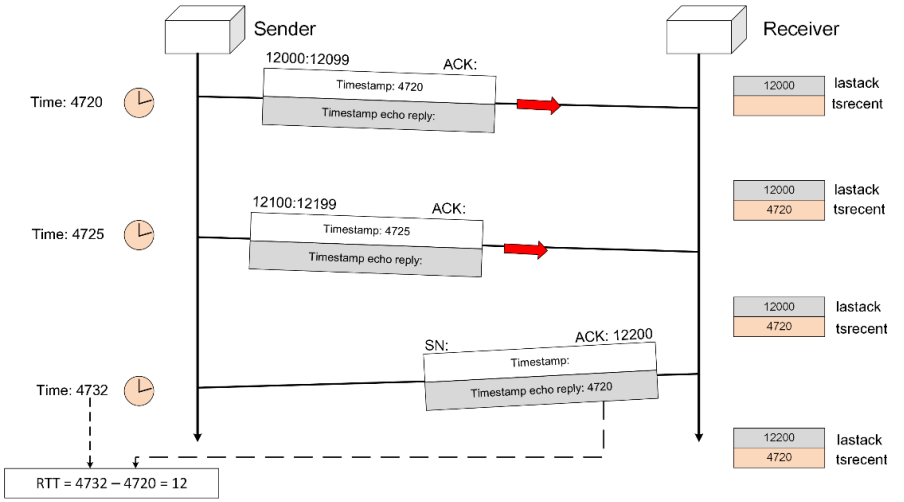
TCP 동작 메커니즘 및 구현

- 옵션 (5/9)
  - 다중 바이트(Multiple-byte) 옵션 (2/6)
    - 타임스탬프(Timestamp)
      - 송신 시점의 시간 정보를 기록하기 위해 사용하는 10바이트 옵션
      - Timestamp Value 필드에 송신 측의 시간 값을 기록함
      - Timestamp Echo Reply 필드에 상대 측이 받은 Timestamp 값을 되돌려 보냄
      - 왕복 시간 RTT 측정과 순서 번호 겹침 방지에 활용함



TCP 동작 메커니즘 및 구현

- 옵션 (6/9)
  - 다중 바이트(Multiple-byte) 옵션 (3/6)
    - 타임스탬프(Timestamp)
      - RTT 측정
        1. 송신 TCP가 세그먼트를 보낼 때 현재 시각을 Timestamp에 기록함
        2. 수신 TCP는 lastack에 해당하는 바이트를 포함한 세그먼트의 Timestamp 값을 tsrecent에 저장함
        3. 누적 ACK를 전송할 때 tsrecent 값을Timestamp Echo Reply에 포함함
        4. 송신 TCP가 ACK를 수신함
        5. 현재 시각에서 Echo Reply 값을 빼서 RTT를 계산함



TCP 동작 메커니즘 및 구현

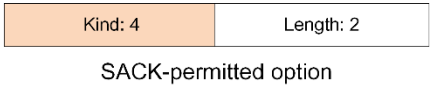
- 옵션 (7/9)
  - 다중 바이트(Multiple-byte) 옵션 (4/6)
    - 타임스탬프(Timestamp)
      - 순서 번호 겹침 방지(PAWS, Protection Against Wrapped Sequence Number)
        - TCP의 순서 번호는 32비트이므로 고속 연결에서는 같은 연결 안에서도 순서 번호가 다시 반복될 수 있음
        - 이전 순서 번호 주기의 지연 세그먼트가 새로운 주기의 세그먼트로 잘못 인식될 수 있음
        - 세그먼트 식별에 순서 번호와 타임스탬프를 함께 사용하여 서로 다른 세그먼트를 구분함
        - 같은 순서 번호라도 타임스탬프 값이 다르면 서로 다른 시점에 전송된 세그먼트로 판단할 수 있음

TCP 동작 메커니즘 및 구현

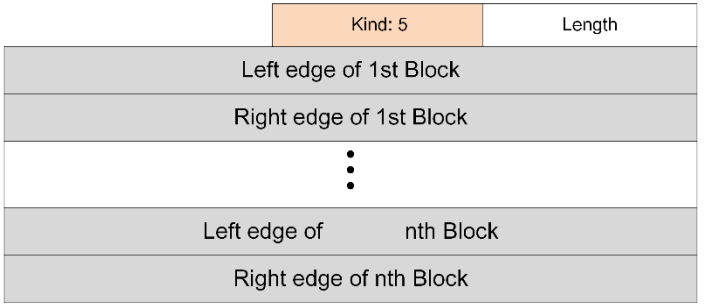
- 옵션 (8/9)
  - 다중 바이트(Multiple-byte) 옵션 (5/6)
    - SACK-Permitted 및 SACK 옵션
      - TCP의 기본 ACK는 누적 확인응답 방식으로, 순서에 어긋나게 도착한 데이터나 중복 세그먼트 정보를 직접 알려주지 못함
      - SACK은 실제로 도착한 데이터 블록을 송신 측에 알려 손실된 세그먼트를 더 정확하게 판단할 수 있도록 함
      - SACK-Permitted 옵션은 연결 설정 과정에서 SACK 사용 가능 여부를 알리는 데 사용함
      - SACK 옵션은 양쪽이 사용에 동의한 경우 데이터 전송 과정에서 사용함

TCP 동작 메커니즘 및 구현

- 옵션 (9/9)
  - 다중 바이트(Multiple-byte) 옵션 (6/6)
    - SACK-Permitted 및 SACK 옵션
      - SACK-허용 옵션은 2바이트 길이를 가지며 연결 설정 과정에서만 사용함
        - SYN 세그먼트에 포함하여 자신이 SACK 옵션을 지원할 수 있음을 알림
        - 상대가 SYN+ACK 세그먼트에 SACK-허용 옵션을 포함하면 양쪽 모두 데이터 전송 중 SACK 옵션을 사용할 수 있음
      - SACK 옵션은 가변 길이를 가지며 데이터 전송 단계에서 사용함
        - 순서에 어긋나게 수신된 데이터 블록의 정보를 포함함



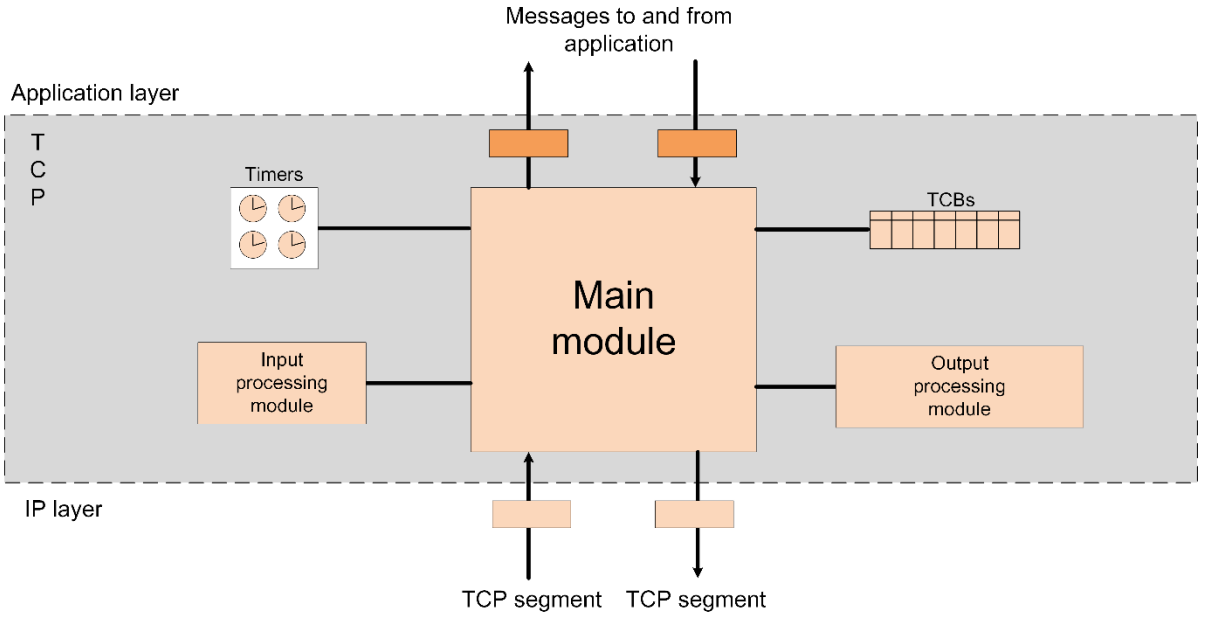
SACK-permitted option



SACK option

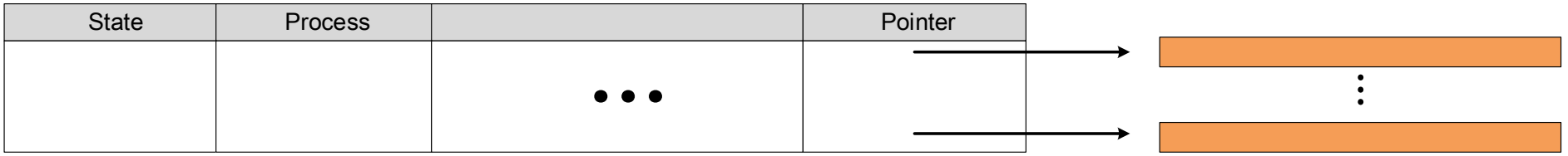
TCP 동작 메커니즘 및 구현

- TCP 패키지 (1/5)
  - 정의
    - TCP 구현을 위한 메커니즘의 집합
  - 구성 요소
    - 전송 제어 블록 TCB
    - 타이머
    - 메인 모듈
    - 입력 처리 모듈
    - 출력 처리 모듈



TCP 동작 메커니즘 및 구현

- TCP 패키지 (2/5)
  - 전송 제어 블록(TCB, Transmission Control Block) (1/2)
    - 정의
      - TCP 연결을 제어하기 위해 각 연결에 대한 정보를 저장하는 구조
    - 구성 요소
      - 상태 정보
      - 식별자 정보
      - 송수신 윈도우 관리 정보
      - 순서 번호 정보
      - 타이머 정보



TCP 동작 메커니즘 및 구현

- TCP 패키지 (3/5)
  - 전송 제어 블록(TCB, Transmission Control Block) (2/2)

필드	설명
상태	연결 상태를 정의함
프로세스	클라이언트 또는 서버로서 기기에 연결을 사용하는 프로세스를 정의함
로컬 IP 주소	연결을 사용하는 로컬 기기의 IP 주소를 정의함
로컬 포트 번호	연결을 사용하는 로컬 포트 번호를 정의함
원격 IP 주소	연결을 사용하는 원격지 기기의 IP 주소를 정의함
원격 포트 번호	연결을 사용하는 원격지 포트 번호를 정의함
인터페이스	지역 인터페이스를 정의
로컬 윈도우	로컬 TCP의 윈도우에 대한 정보를 담고 있음
원격 윈도우	원격지 TCP의 윈도우에 대한 정보를 담고 있음
송신 순서 번호	송신 순서 번호를 가짐
수신 순서 번호	수신 순서 번호를 가짐
송신 ACK 번호	전송한 ACK 번호 값을 가짐
RTT	MRTT, SRTT 등 다양한 RTT의 정보를 보관함
타임아웃 값	재전송 타임아웃, 영속 타임아웃, 킵얼라이브 타임아웃 등 여러 타임아웃 값을 보관함
버퍼 크기	로컬 TCP의 버퍼 크기를 정의함
버퍼 포인터	응용 프로그램에 읽히기 전의 수신 데이터의 위치를 알려주는 포인터 값을 가짐

TCP 동작 메커니즘 및 구현

- TCP 패키지 (4/5)
  - 메인 모듈(Main Module)
    - 정의
      - TCP 세그먼트 도착, 타임아웃 발생, 응용 프로그램의 메시지 수신 시 호출되는 모듈
    - 특징
      - 현재 TCP 연결 상태에 따라 수행할 동작이 달라짐
      - 상태 전이 다이어그램의 각 상태를 기준으로 동작을 처리함
      - 11개의 TCP 상태를 각각의 경우(case)로 구분하여 구현함
      - ESTABLISHED 상태에서 데이터나 ACK를 수신하면 입력 처리 모듈을 호출함
      - 응용 프로그램의 데이터 전송 요청이 발생하면 출력 처리 모듈을 호출함

TCP 동작 메커니즘 및 구현

- TCP 패키지 (5/5)
  - 입력 처리 모듈(Input Processing Module)
    - ESTABLISHED 상태에서 수신한 데이터와 ACK의 처리를 담당함
    - 필요한 ACK 전송, 윈도우 크기 광고, 오류 검사 등을 수행함
  - 출력 처리 모듈(Output Processing Module)
    - ESTABLISHED 상태에서 응용 프로그램으로부터 전달받은 데이터의 전송을 담당함
    - 재전송 타임아웃, 영속 타임아웃 등 데이터 전송과 관련된 처리를 수행함
  - 타이머(Timers)
    - TCP의 각 동작에서 시간과 관련된 이벤트를 관리함
    - 재전송 타이머, 영속 타이머, 킵얼라이브 타이머, TIME-WAIT 타이머를 사용함

Thanks!

김 민 식(minsik@pel.sejong.ac.kr)