2025/09/26, 2025 보안 기초 세미나

TCP/IP 프로토콜

-15장 TCP-

이 성 현(seonghyeon@pel.Sejong.ac.kr)

세종대학교 프로토콜공학연구실

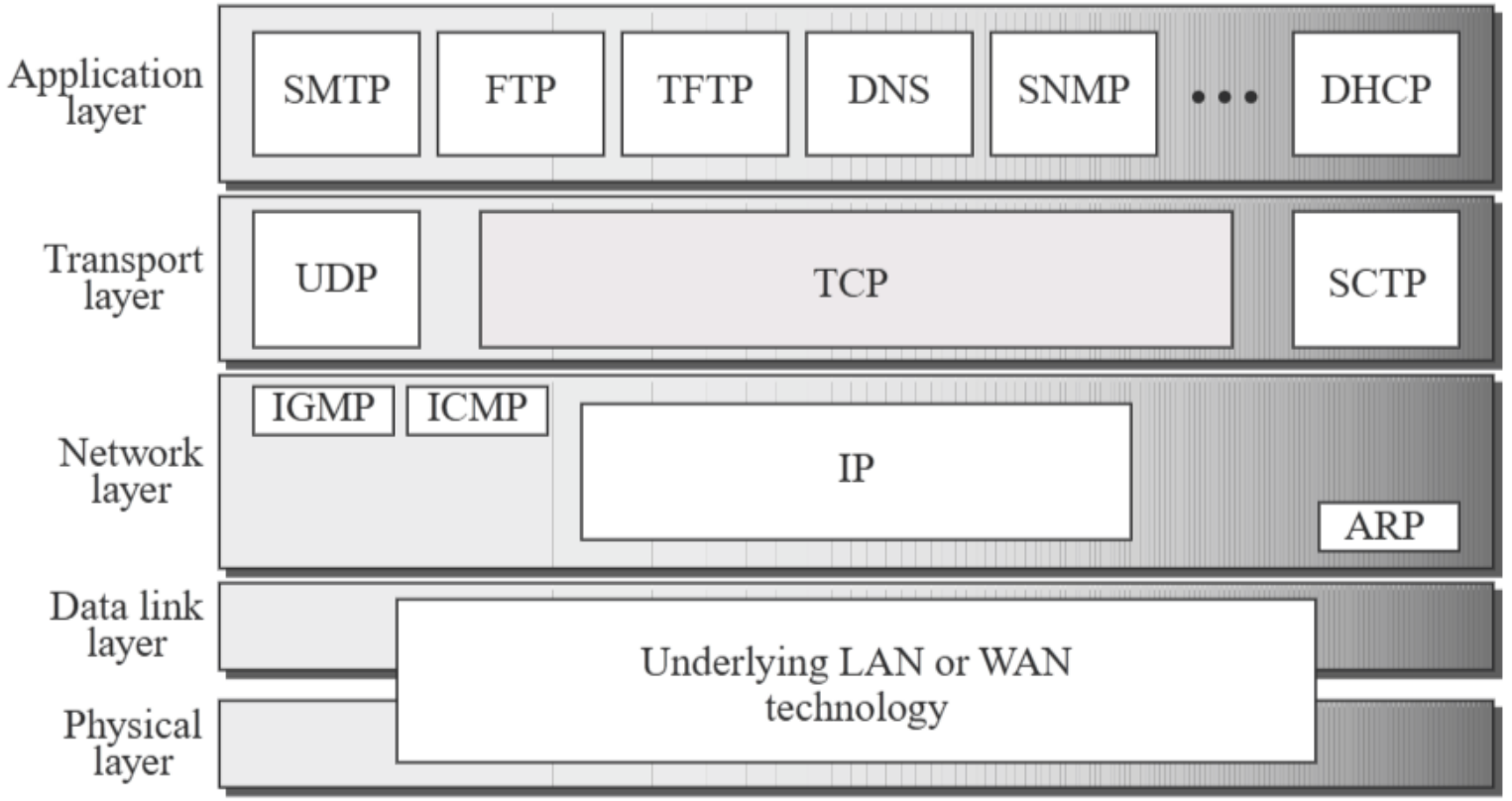
목 차

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

목 차

TCP 개요

- TCP (Transmission Control Protocol)
  - 정의
    - 신뢰성 있는 스트림 배달 서비스를 제공하는 전송 계층 프로토콜



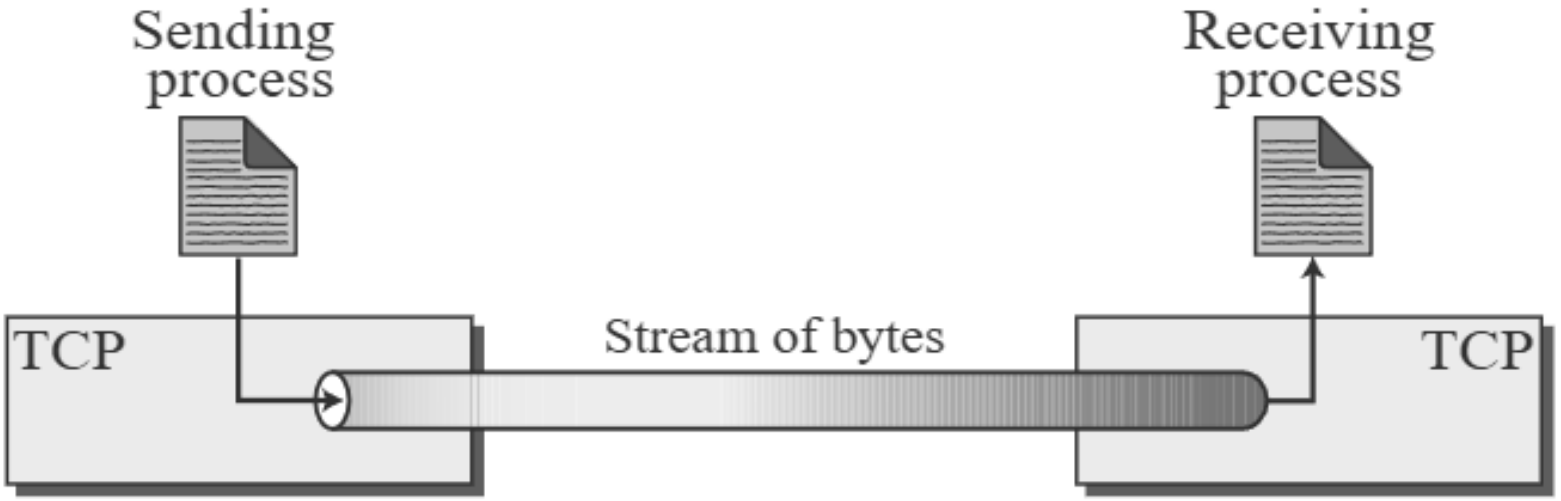
TCP 개요

- TCP 서비스 (1/6)
  - 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

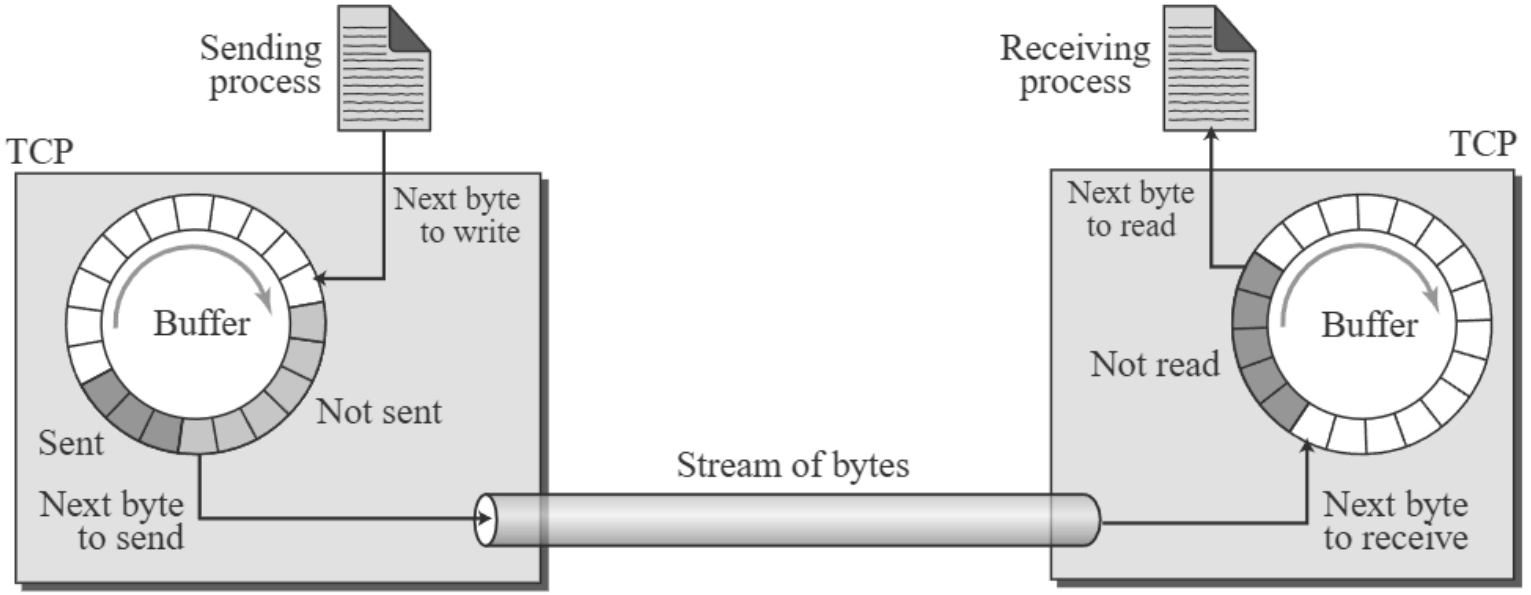
TCP 개요

- TCP 서비스 (2/6)
  - 스트림 배달 서비스 (1/3)
    - TCP에서 송신 / 수신 프로세스는 바이트 스트림의 형태로 데이터를 전송 / 수신할 수 있음
    - 송신 프로세스는 바이트 스트림을 생성하고 수신 프로세스는 바이트 스트림을 소비함



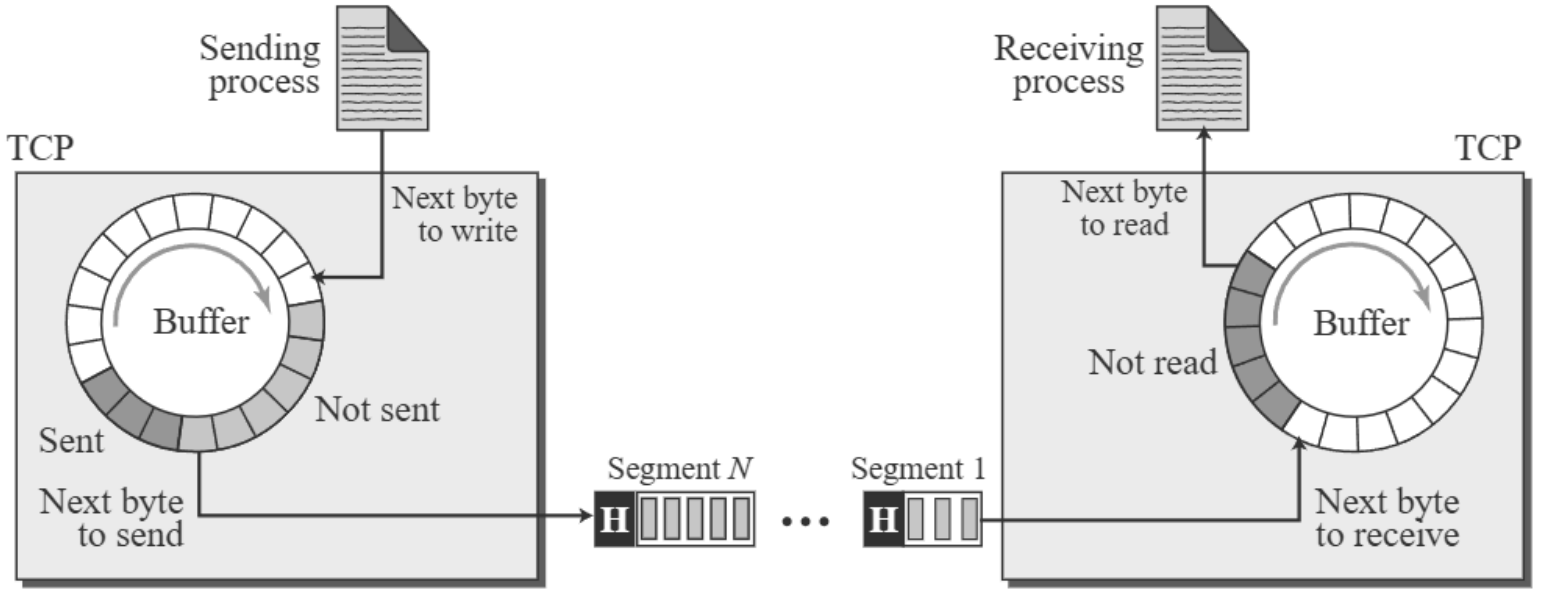
TCP 개요

- TCP 서비스 (3/6)
  - 스트림 배달 서비스 (2/3)
    - 송신 / 수신 버퍼
      - 송신 / 수신 프로세스가 동일한 속도로 데이터를 생성하거나 소비하지 않을 수 있으므로 TCP에서는 데이터를 저장하기 위한 버퍼가 필요함
      - TCP는 전송된 바이트도 확인응답을 수신하기 전까지 버퍼에 보관함



TCP 개요

- TCP 서비스(4/6)
  - 스트림 배달 서비스(3/3)
    - 세그먼트
      - IP는 바이트 스트림의 형태가 아닌 패킷의 형태로 데이터를 전달하므로 TCP는 일련의 바이트를 세그먼트라고 하는 패킷으로 그룹화함
      - 세그먼트 크기가 동일할 필요는 없음



TCP 개요

- TCP 서비스(5/6)
  - 전 이중 통신(Full-duplex)
    - 전 이중 서비스를 제공하여 데이터를 동시에 양방향으로 전송할 수 있음
    - 각 TCP는 송신 버퍼와 수신 버퍼를 가지고 있으며 세그먼트는 양방향으로 전송됨
  - 다중화(Multiplexing) 및 역다중화(Demultiplexing)
    - 송신 측에서 다중화를 수행하고 수신 측에서 역다중화를 수행함
    - 연결 지향 프로토콜이므로 해당 프로세스 간에 연결이 설정되어야 함

*다중화(Multiplexing)
여러 데이터를 하나의 채널로 합쳐서 보내는 과정
*역다중화(Demultiplexing)
도착한 데이터를 목적지에 맞게 분리하는 과정

TCP 개요

- TCP 서비스(6/6)
  - 연결 지향 서비스
    - 두 노드 간 프로세스가 데이터를 주고 받을 때 가상의 연결을 설정해야 함
    - 가상 연결을 수립함으로써 IP를 통해 전송되어도 신뢰성 있는 데이터 전송을 보장함
  - 신뢰성 서비스
    - 데이터가 안전하고 오류 없이 잘 도착했는지 확인하기 위해 확인응답 메커니즘을 이용함

TCP 개요

- TCP 특징 (1/3)
  - 번호화 시스템 (1/2)
    - 바이트 번호
      - TCP는 데이터의 각 바이트에 번호를 매김
      - $0 \sim 2^{32} - 1$ 사이의 임의의 값을 시작 바이트 번호로 설정함
    - 순서 번호(Sequence Number)
      - 전송하고자 하는 세그먼트의 첫 번째 바이트 번호를 순서 번호로 하여 하나의 순서 번호를 할당함
      - 데이터 없이 제어 정보만을 포함하는 세그먼트는 하나의 순서 번호를 사용함
      - 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

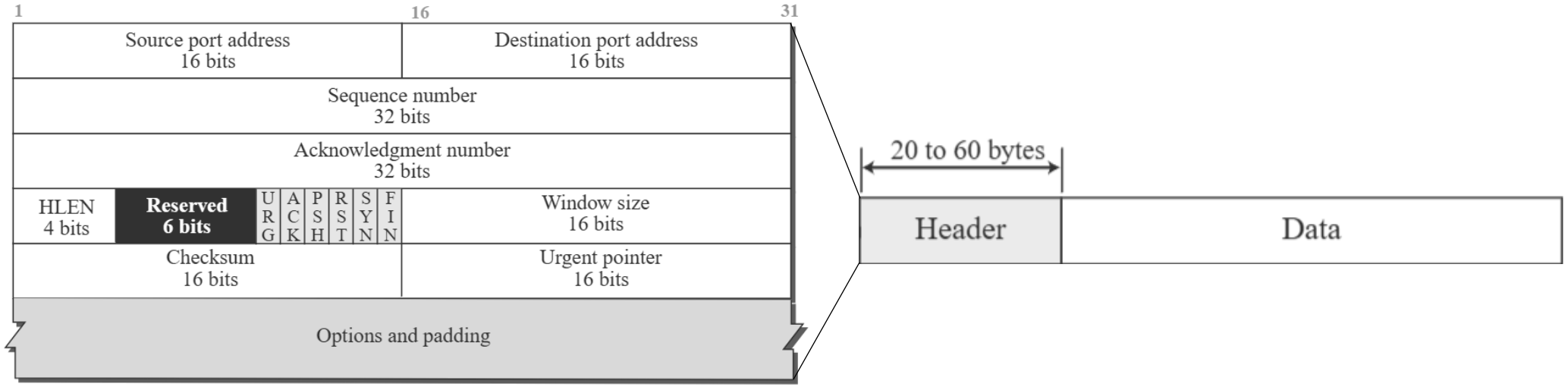
- TCP 특징 (2/3)
  - 번호화 시스템 (2/2)
    - 확인응답 번호 (ACK, Acknowledgement Number)
      - 각 TCP는 자신이 바이트를 수신했다는 것을 확인시키기 위해 확인응답 번호를 이용함
      - 확인응답 번호는 자신이 수신하기를 기대하는 다음 바이트의 번호를 나타냄
        - e.g., 1,000까지의 바이트를 수신했다면 1,001을 확인응답 번호로 하여 ACK 세그먼트를 전송함
      - 확인응답 번호는 누적적(Cumulative)임
        - 확인응답 번호 이전의 바이트 번호를 가진 바이트는 수신됐음을 의미함

TCP 개요

- TCP 특징 (3/3)
  - 흐름 제어(Flow Control)
    - 수신 측에서 데이터가 과도하게 수신되어 데이터가 손실되는 일을 방지함
  - 오류 제어(Error Control)
    - 신뢰성 있는 서비스를 제공하기 위해 오류 제어 메커니즘을 구현함
  - 혼잡 제어(Congestion Control)
    - 망의 혼잡을 고려하여 송신 측에서 전송되는 데이터의 양을 조절함

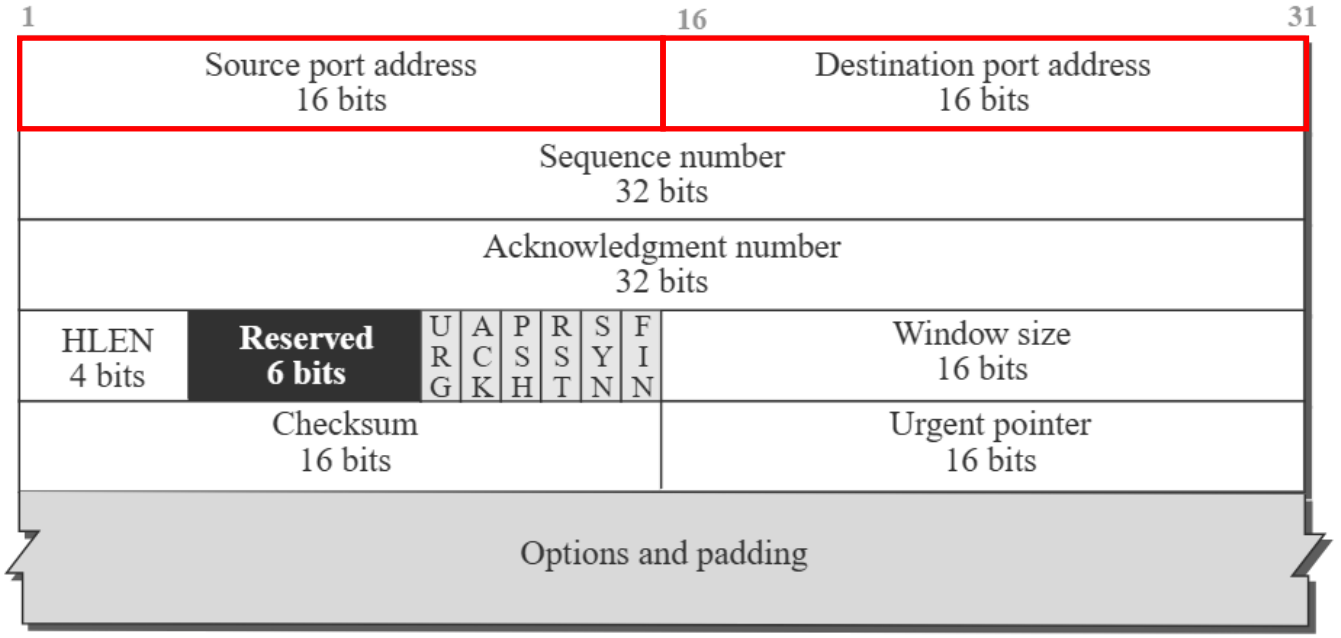
TCP 개요

- 세그먼트 (1/8)
  - 정의
    - TCP가 전송하는 데이터 단위
  - 포맷
    - 세그먼트는 20~60 바이트의 헤더와 응용 프로그램으로부터 생성되는 데이터로 구성되어 있음



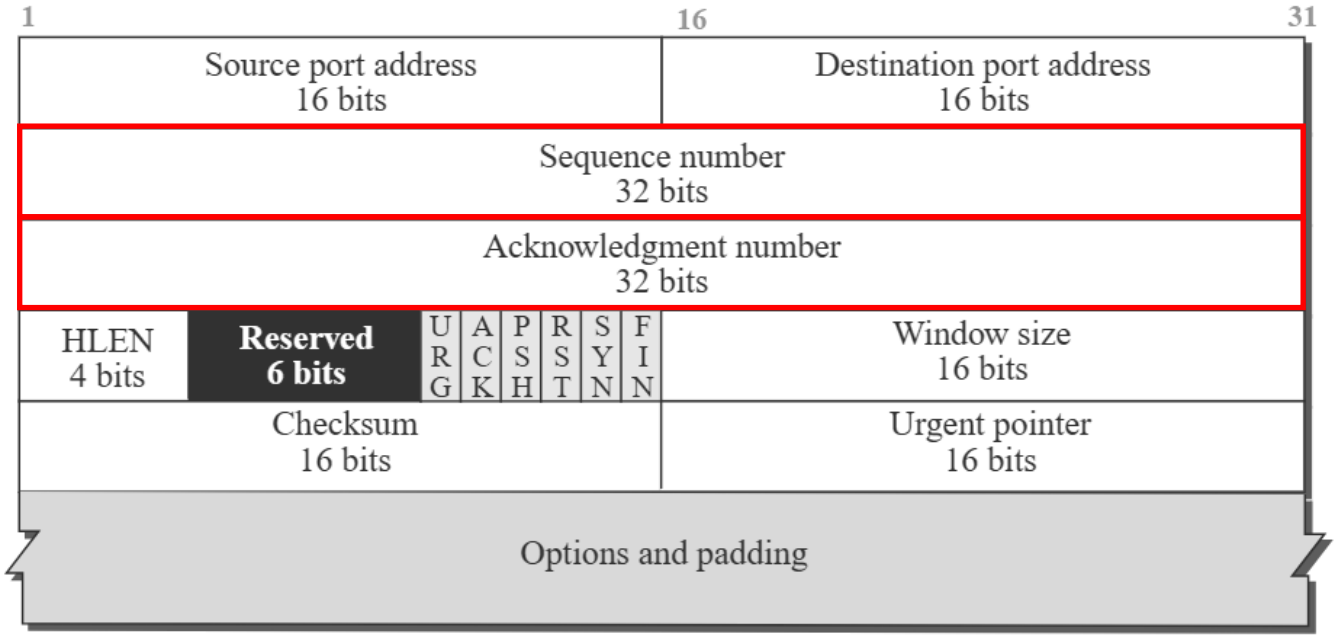
TCP 개요

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



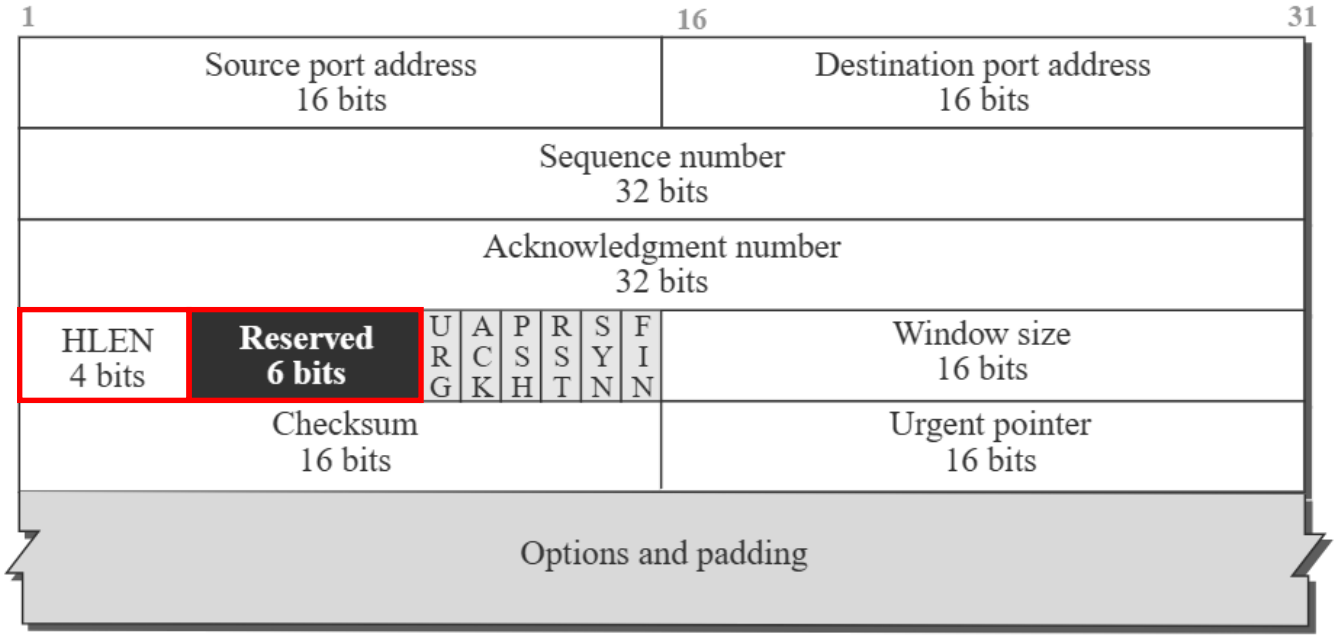
TCP 개요

- 세그먼트 (3/8)
  - 순서 번호
    - 세그먼트에 포함된 데이터의 첫 번째 바이트 번호
  - 확인응답 번호
    - 수신 노드가 송신 노드로부터 수신하고자 하는 바이트 번호



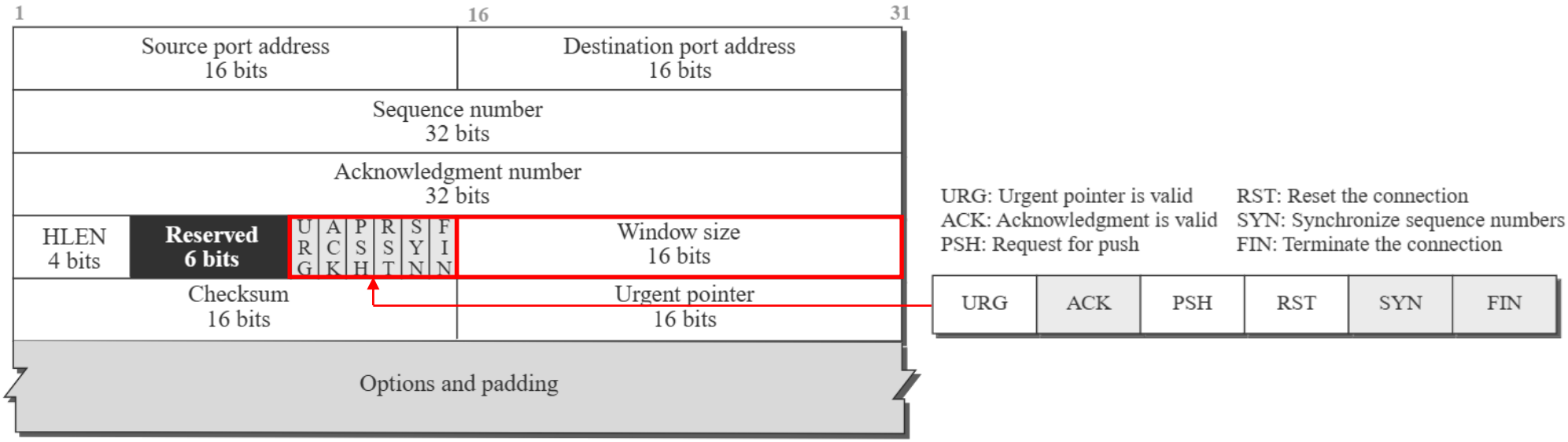
TCP 개요

- 세그먼트 (4/8)
  - 헤더 길이(HLEN, Header Length)
    - 헤더의 길이를 나타내는 4 바이트 단위 4 비트 필드
  - 예약(Reserved)
    - 차후 사용을 위해 예약된 6비트 필드



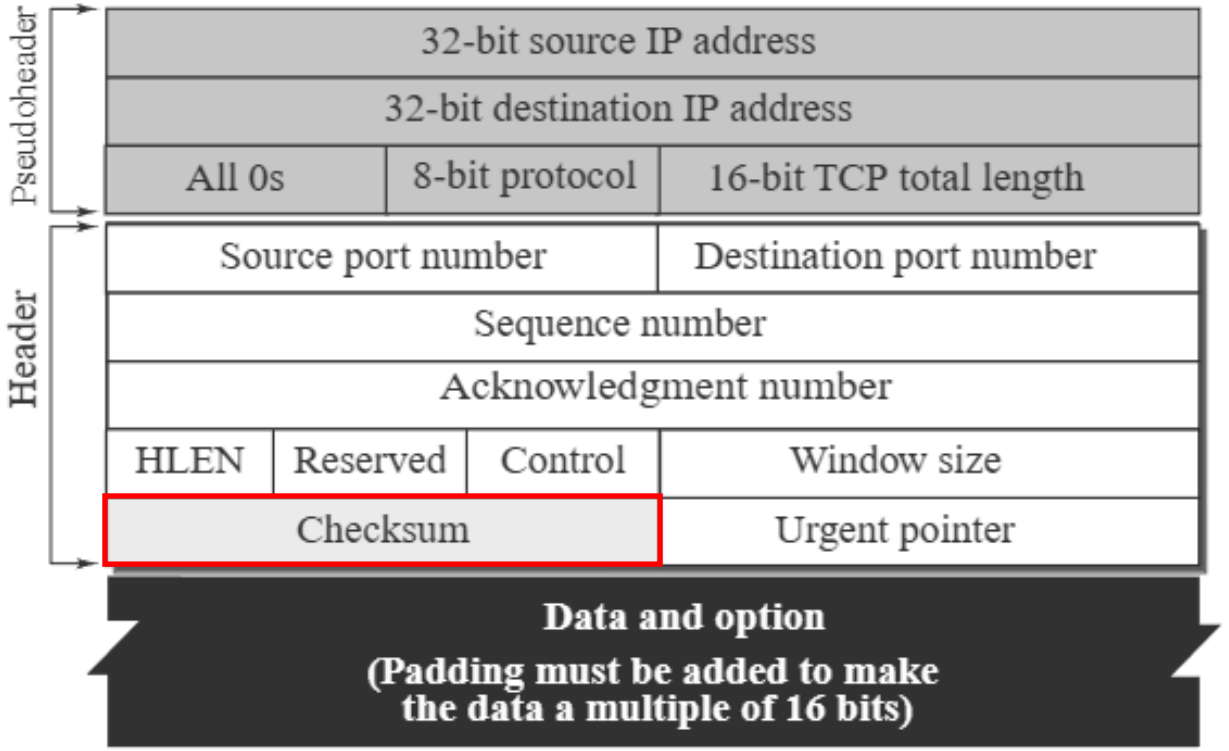
TCP 개요

- 세그먼트 (5/8)
  - 제어(Control)
    - 제어를 위해 사용되는 6개의 플래그 비트
  - 윈도우 크기(Window Size)
    - 수신 측에 의해 결정되는 송신 TCP의 수신 윈도우(rwnd, Receiving Window)의 크기



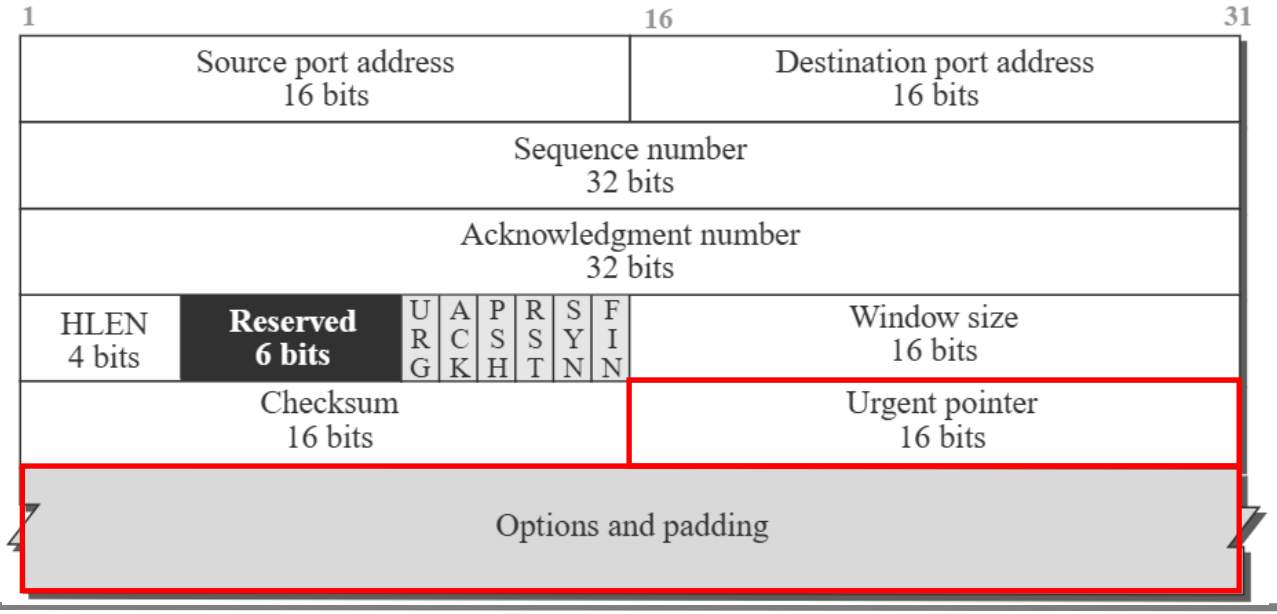
TCP 개요

- 세그먼트 (6/8)
  - 체크섬(Checksum)
    - 오류 검사를 위해 사용되는 필드
    - 검사 및 계산 시에는 IP 주소를 포함한 의사 헤더를 추가하여 수행함



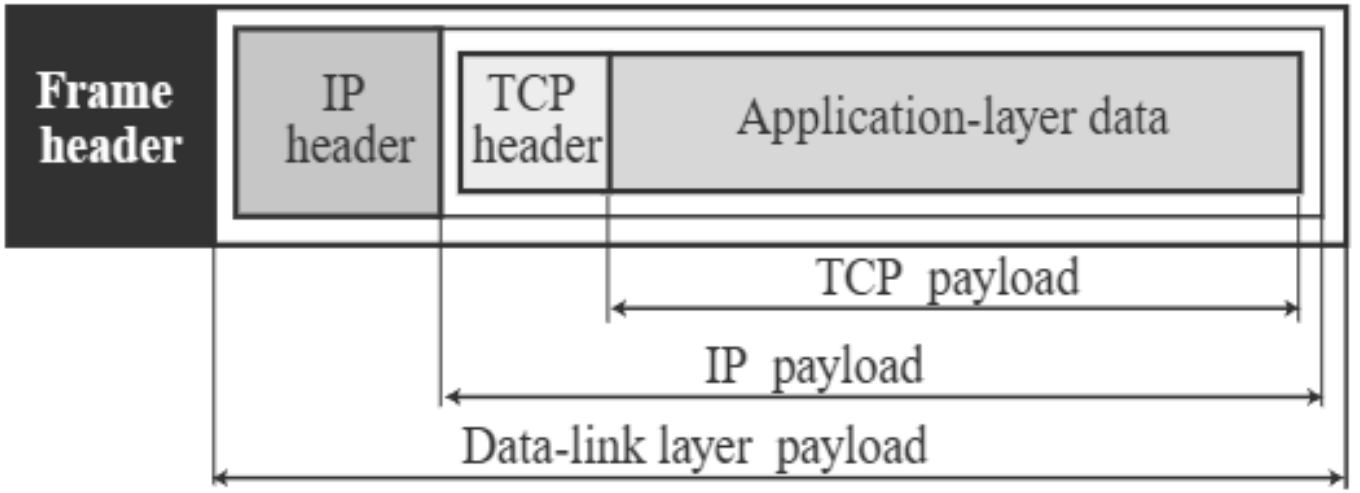
TCP 개요

- 세그먼트 (7/8)
  - 긴급 포인터(Urgent Pointer)
    - 세그먼트 내 긴급 데이터의 마지막 바이트가 순서 번호로부터 몇 바이트 떨어져 있는지 나타내는 필드
  - 옵션(Option)
    - 최대 40 바이트의 추가적인 옵션 정보



TCP 개요

- 세그먼트 (8/8)
  - 캡슐화
    - TCP 세그먼트는 응용 계층으로부터 들어온 데이터를 캡슐화함
    - TCP 세그먼트는 IP 데이터그램에 캡슐화 되고 IP 데이터그램은 데이터링크 계층의 프레임에 캡슐화됨



목 차

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

TCP 연결 및 전송

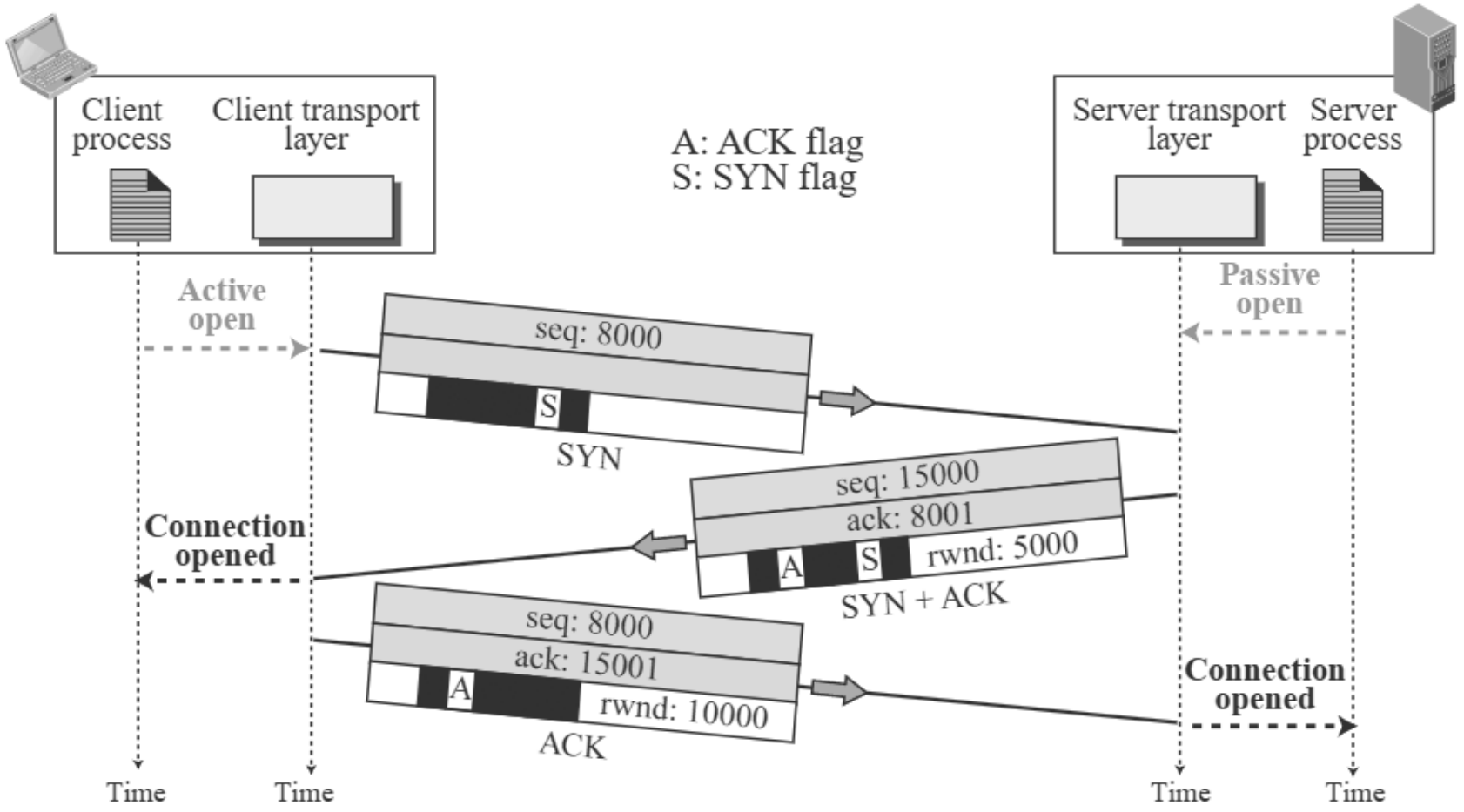
- 개요
  - TCP는 연결 지향 프로토콜로서, 통신 전 발신지와 목적지 간에 가상 경로를 설정함
  - TCP에서 연결 지향 전송은 연결 설정, 데이터 전송, 연결 종료를 통해 이루어짐

TCP 연결 및 전송

- 연결 설정 (1/2)
  - 개방 요청
    - 서버 프로그램이 자신의 TCP에게 Passive Open 요청을 통해 연결을 수락할 준비가 됐음을 알림
    - 클라이언트 프로그램이 자신의 TCP에게 Active Open 요청을 통해 특정 서버와 연결을 설정할 것임을 알림
  - 3단계 핸드셰이크(Three-way Handshaking) (1/2)
    1. 클라이언트가 순서 번호 동기화를 위해 초기 순서 번호(ISN, Initial Sequence Number)를 포함한 SYN 세그먼트를 전송함
    2. 서버는 자신이 사용할 ISN과 클라이언트의 SYN 세그먼트에 대한 확인응답 번호를 포함한 SYN+ACK 세그먼트를 전송함
    3. 클라이언트는 서버의 SYN에 응답하기 위해 ACK 세그먼트를 전송함

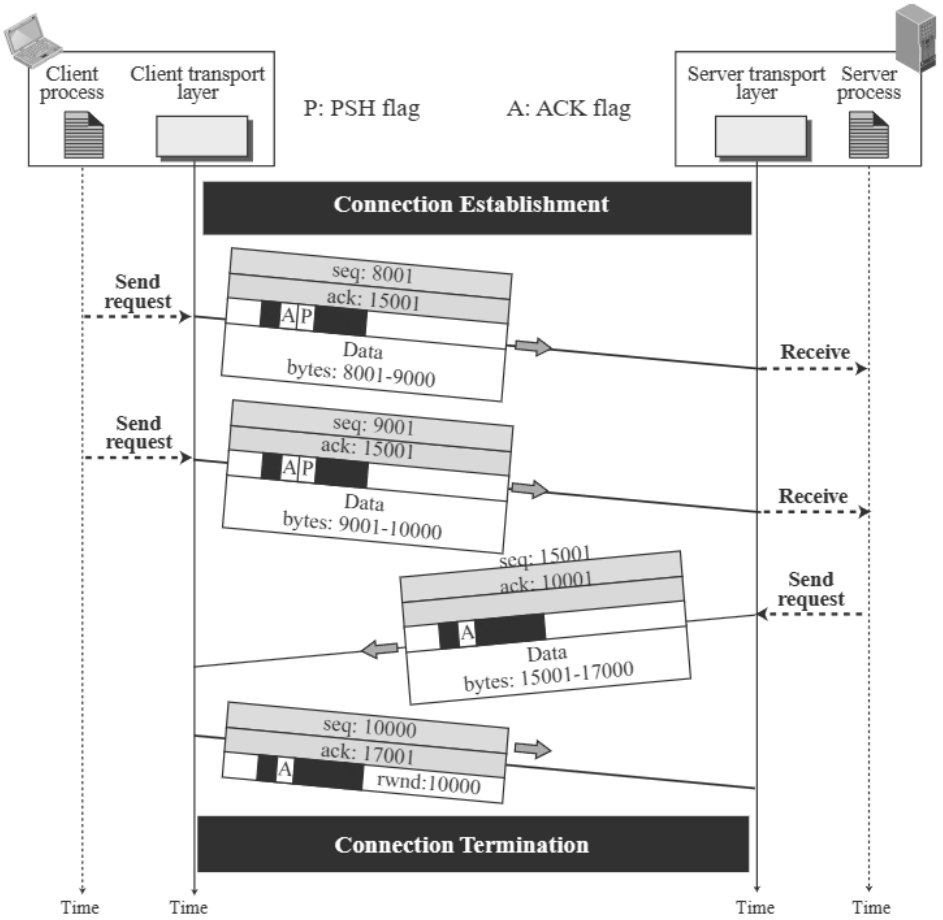
TCP 연결 및 전송

- 연결 설정 (2/2)
  - 3단계 핸드셰이크(Three-way Handshaking) (2/2)



TCP 연결 및 전송

- 데이터 전송 (1/2)
  - 동일한 방향으로 전송되는 데이터와 확인응답은 하나의 세그먼트로 전달될 수 있음



TCP 연결 및 전송

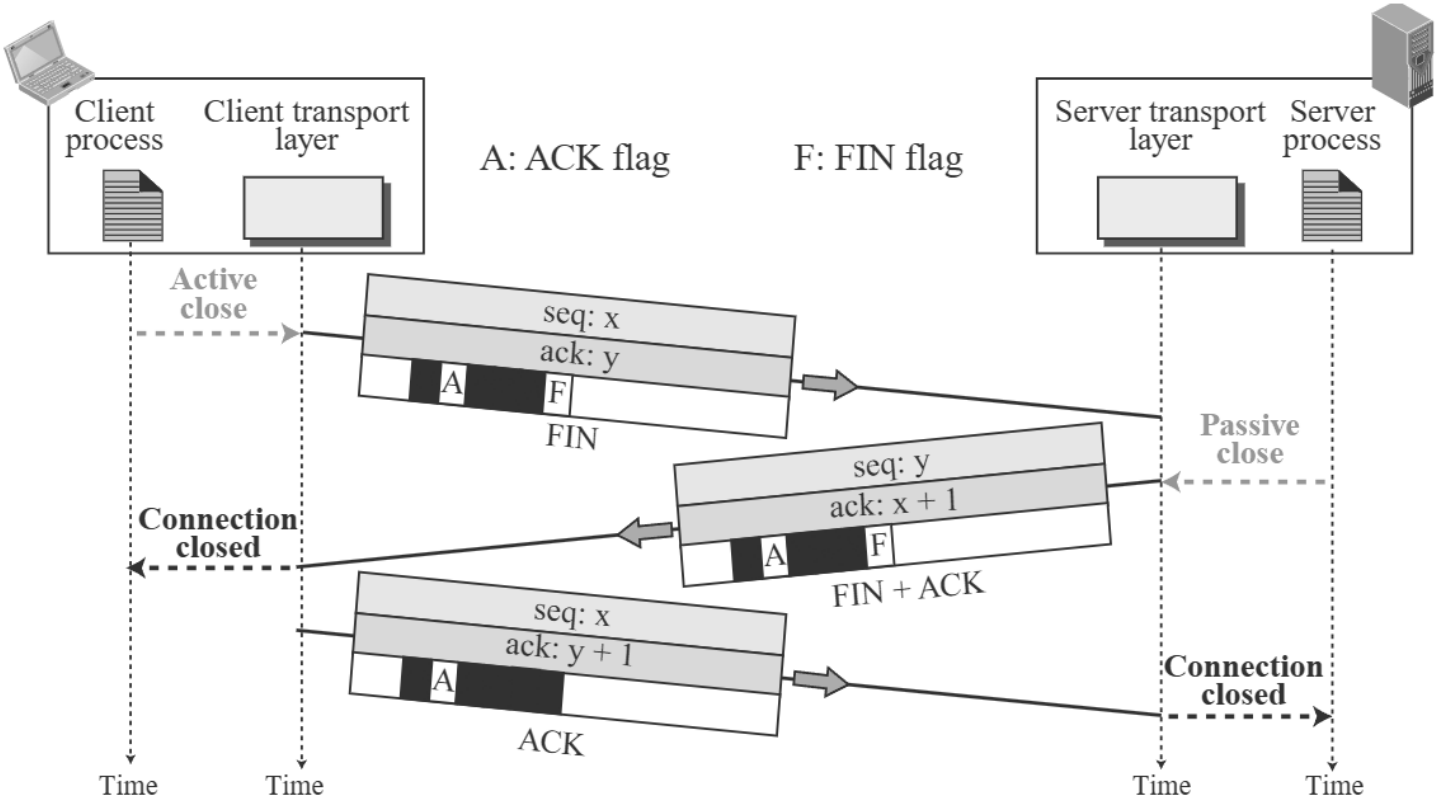
- 데이터 전송 (2/2)
  - 푸싱 데이터(Pushing Data)
    - PSH 플래그 비트를 1로 설정하여 전송함
    - 낮은 지연이 필요한 상황에서 사용하여, 송신 TCP는 즉시 해당 데이터를 세그먼트로 만들어 전송하고 수신 TCP는 수신한 데이터를 즉시 수신 프로세스로 전달함
  - 긴급 데이터(Urgent Data)
    - URG 플래그 비트를 1로 설정하고 긴급 포인터 필드를 채워 전송함
    - 수신 프로세스가 긴급하게 읽었으면 하는 데이터가 존재하면, 긴급 포인터를 통해 표시하여 해당 바이트까지의 데이터를 바로 수신 프로세스로 전달하도록 함

TCP 연결 및 전송

- 연결 종료 (1/4)
  - 종료 요청
    - 클라이언트 프로그램이 자신의 TCP에게 Active Close 요청을 통해 연결중이던 서버와 연결을 종료할 것임을 알림
    - 서버 프로그램이 자신의 TCP에게 Passive Close 요청을 통해 해당 클라이언트와의 연결을 종료할 것임을 알림
  - 3단계 핸드셰이크 (1/2)
    - 클라이언트 프로세스로부터 Close 요청을 받은 클라이언트 TCP가 FIN 세그먼트를 전송함
    - 서버가 FIN 세그먼트에 대한 확인응답과 반대 방향의 연결 종료를 알리기 위해 FIN+ACK 세그먼트를 전송함
    - 클라이언트가 서버의 FIN 세그먼트에 대한 확인응답을 위해 ACK 세그먼트를 전송함

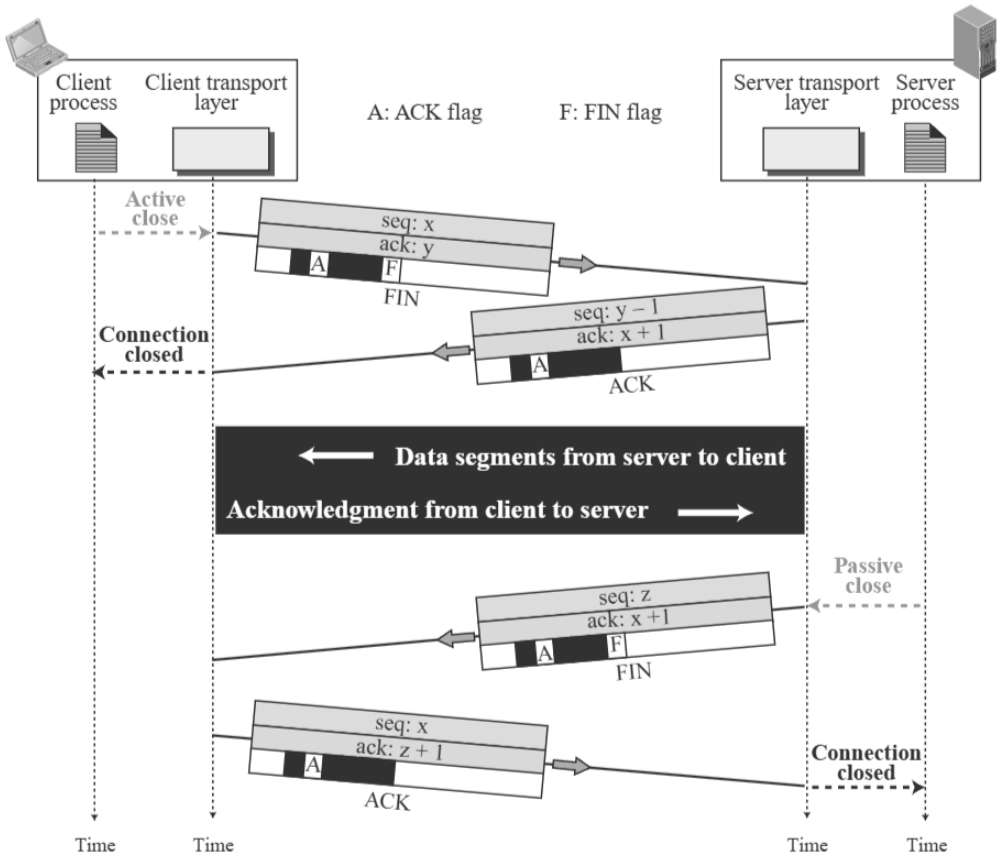
TCP 연결 및 전송

- 연결 종료 (2/4)
  - 3단계 핸드셰이크 (2/2)
    - 데이터를 포함하지 않는 FIN 세그먼트는 순서 번호 하나를 소비함



TCP 연결 및 전송

- 연결 종료 (3/4)
  - 반-닫기(Half-close)
    - 한 쪽에서 데이터를 수신하면서 데이터 전송을 종료함

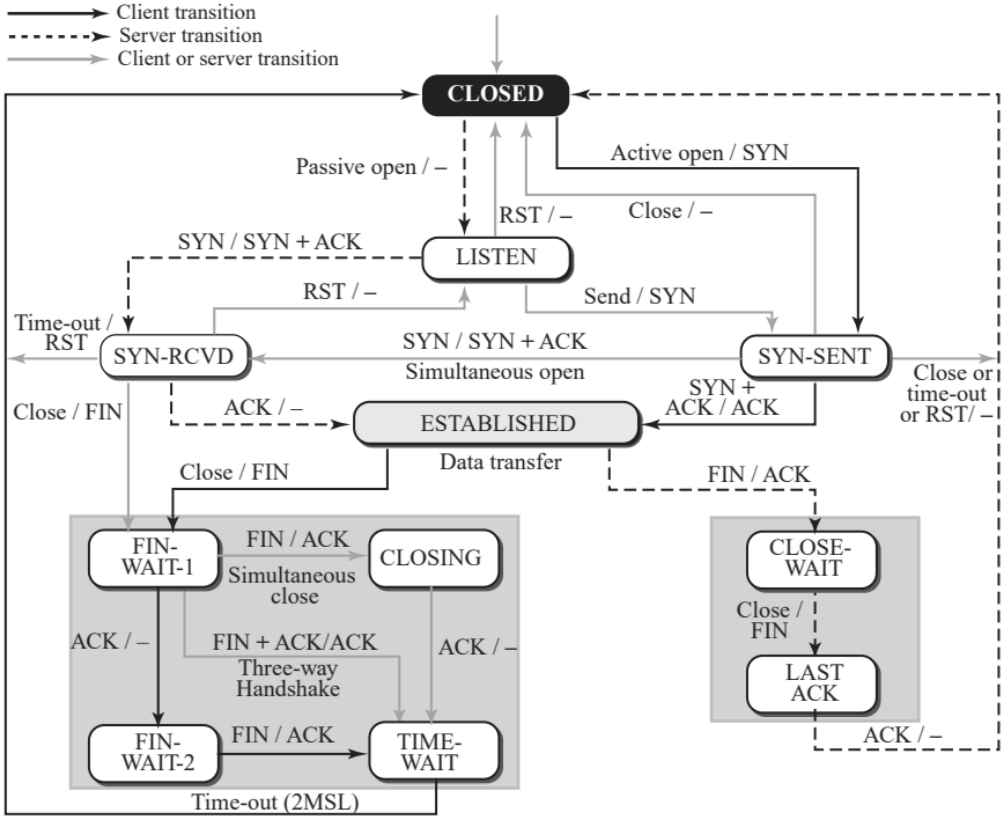


TCP 연결 및 전송

- 연결 종료 (4/4)
  - 연결 리셋
    - 연결 거절
      - 한쪽의 TCP가 존재하지 않은 포트로 연결을 요청하는 경우, 상대 TCP는 요청을 거절하기 위해 RST 세그먼트를 전송함
    - 연결 중단
      - 한 TCP는 비정상적인 상황의 경우, RST 세그먼트를 전송하여 설정되어있는 연결을 중단할 수 있음
    - 휴지 연결의 종료
      - 한 TCP는 상대 TCP가 오랜 시간동안 휴지 상태에 있다는 것을 확인한 후 연결을 파기하기 위해 RST 세그먼트를 전송함

TCP 연결 및 전송

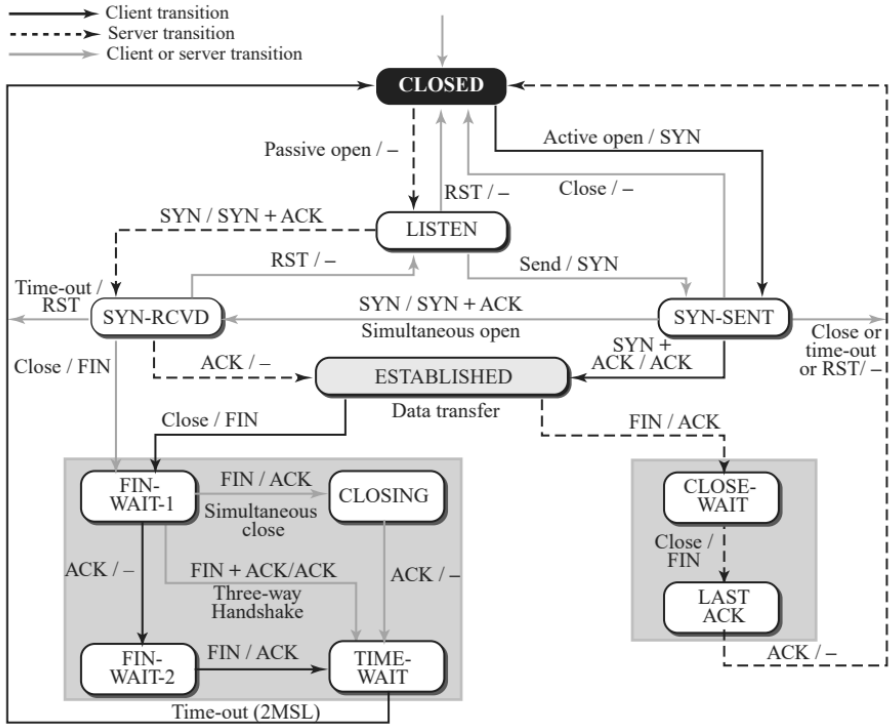
- 상태 천이 다이어그램(State Transition Diagram) (1/3)
  - 정의
    - FSM (Finite State Machine)에서 상태(State)와 천이(Transition)를 표현하기 위한 도식



TCP 연결 및 전송

- 상태 천이 다이어그램(State Transition Diagram) (2/3)
  - 표현 (1/2)
    - 도형 내 단어는 상태를 나타냄

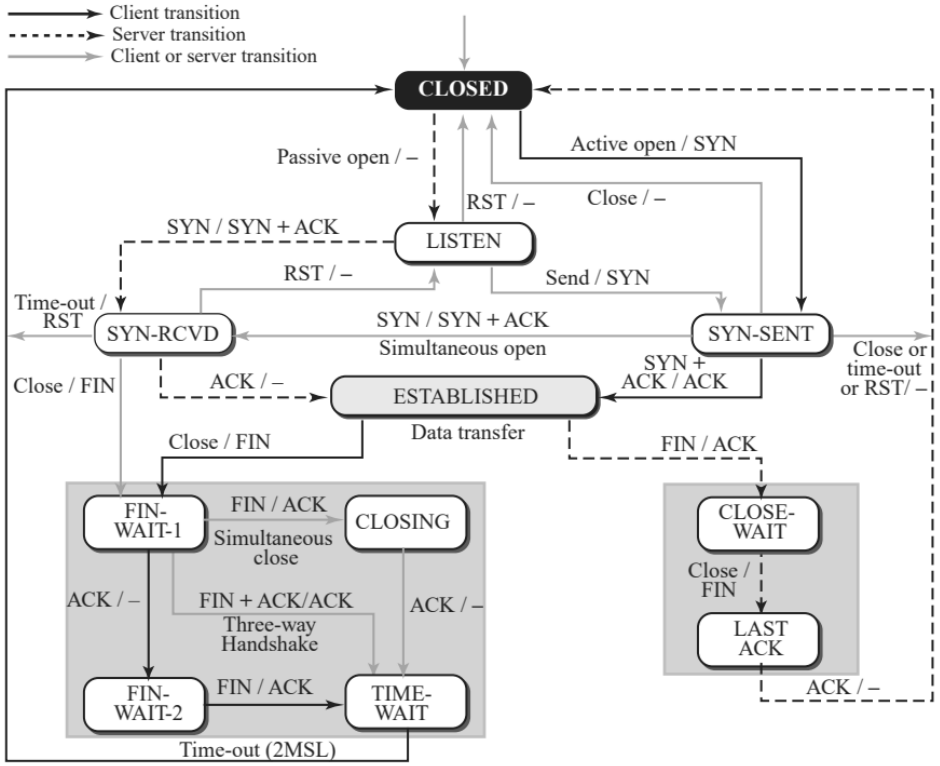
*최대 세그먼트 수명(MSL, Maximum Segment Lifetime)
네트워크 상에서 TCP 세그먼트가 살아남을 수 있는 최대 시간



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, 연결종료를 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

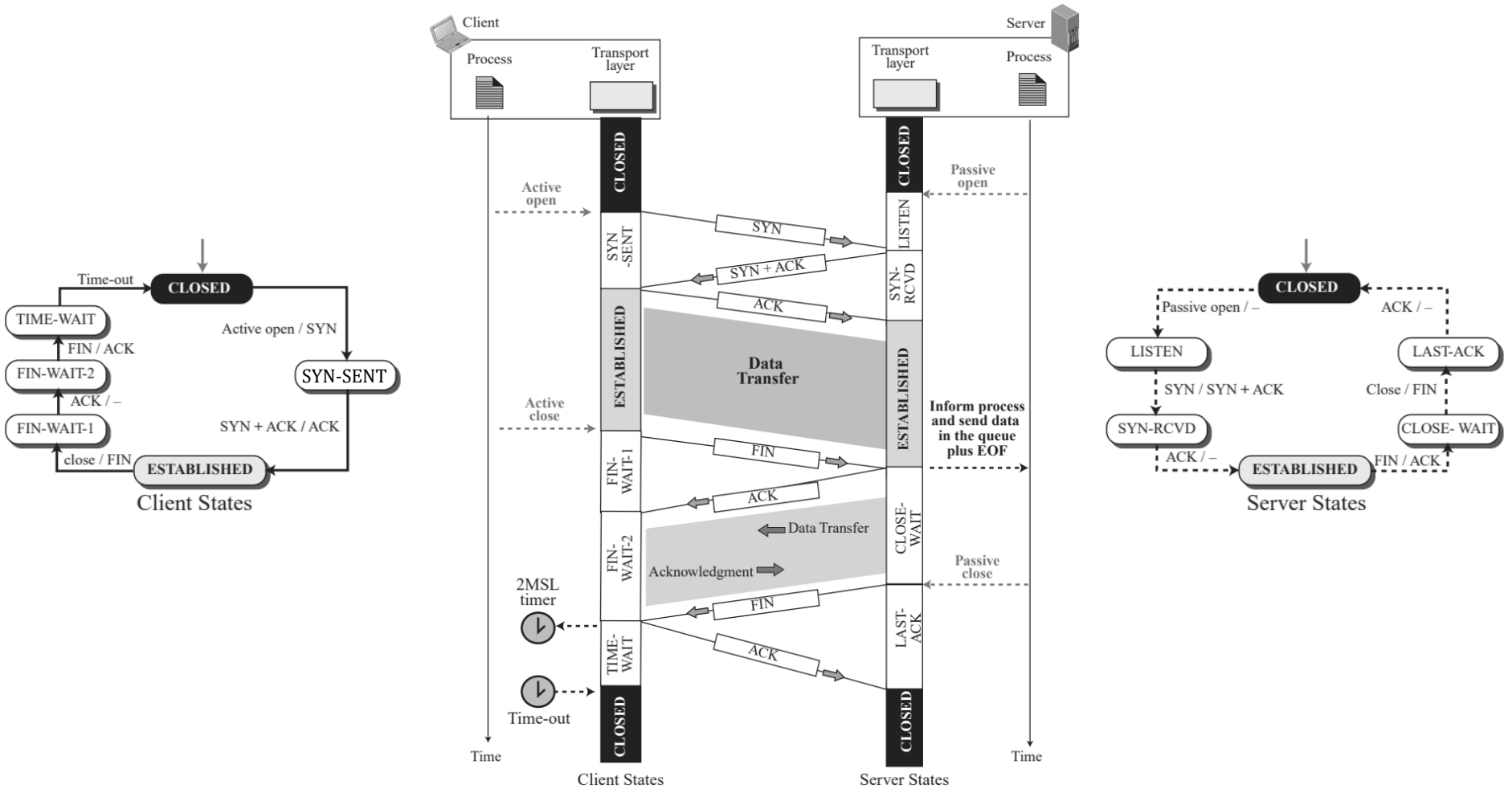
TCP 연결 및 전송

- 상태 천이 다이어그램(State Transition Diagram) (3/3)
  - 표현 (2/2)
    - 화살표는 천이를 나타내고 'TCP가 수신하는 입력 / TCP가 전송하는 출력'을 함께 작성함



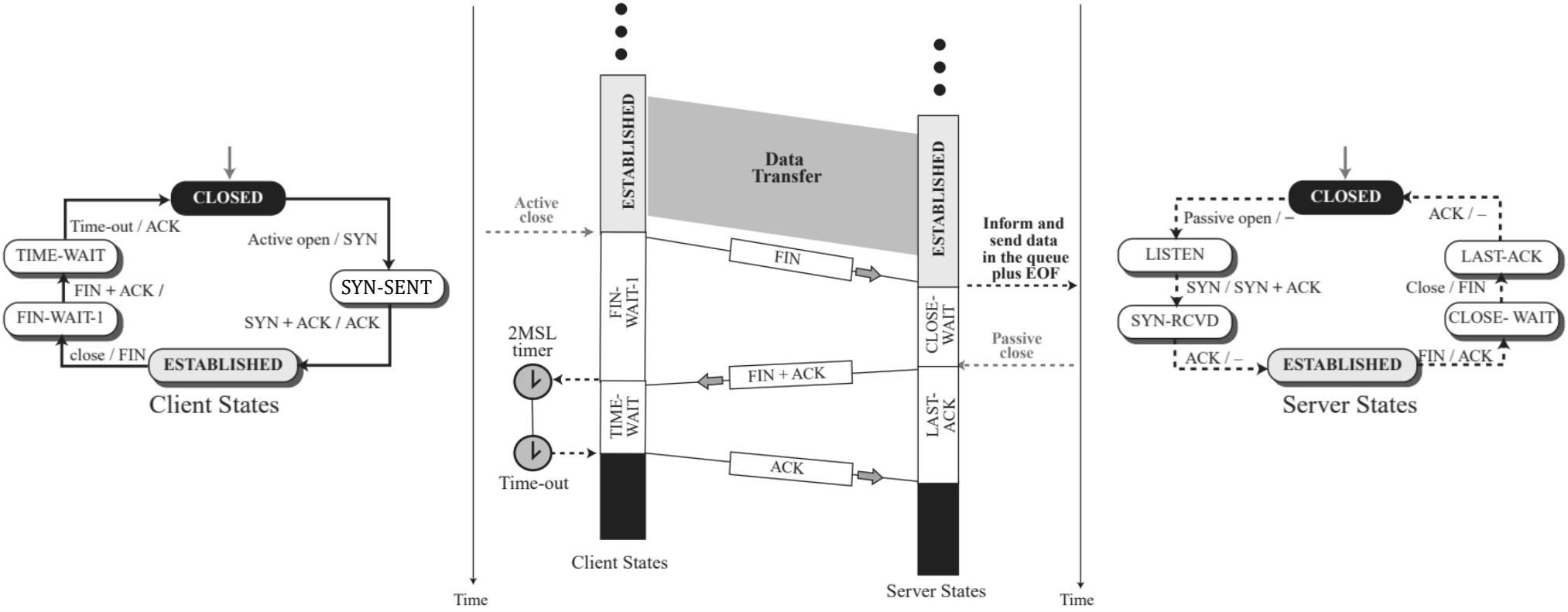
TCP 연결 및 전송

- 연결 및 전송 시나리오 (1/6)
  - 연결 설정 및 반-닫기



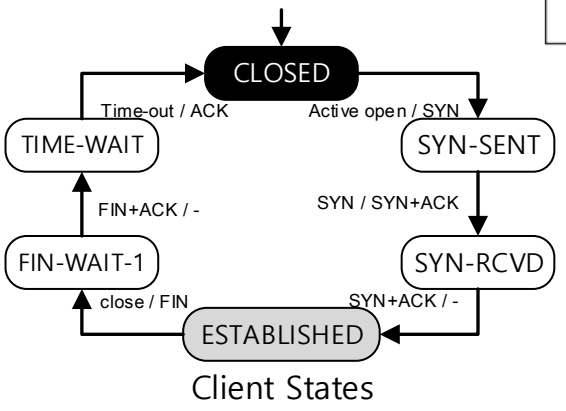
TCP 연결 및 전송

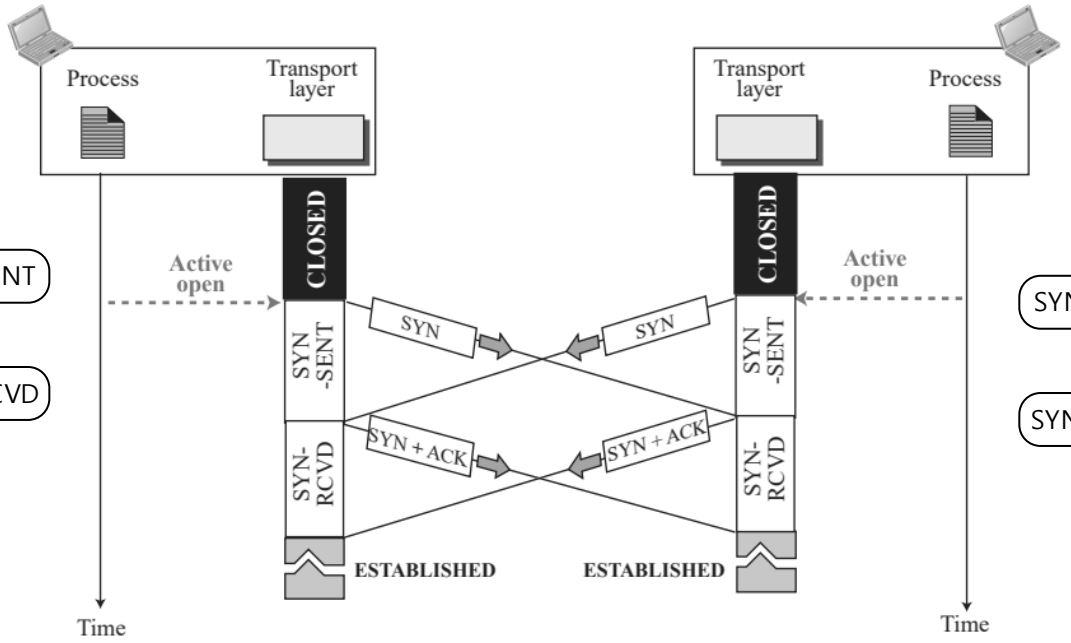
- 연결 및 전송 시나리오 (2/6)
  - 연결 설정 및 3단계 핸드셰이크 종료

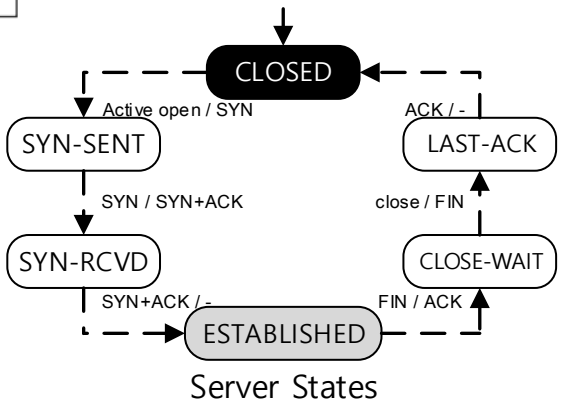


TCP 연결 및 전송

- 연결 및 전송 시나리오 (3/6)
  - 동시 개방
    - 두 호스트가 동시에 연결 설정을 시도하여 발생함

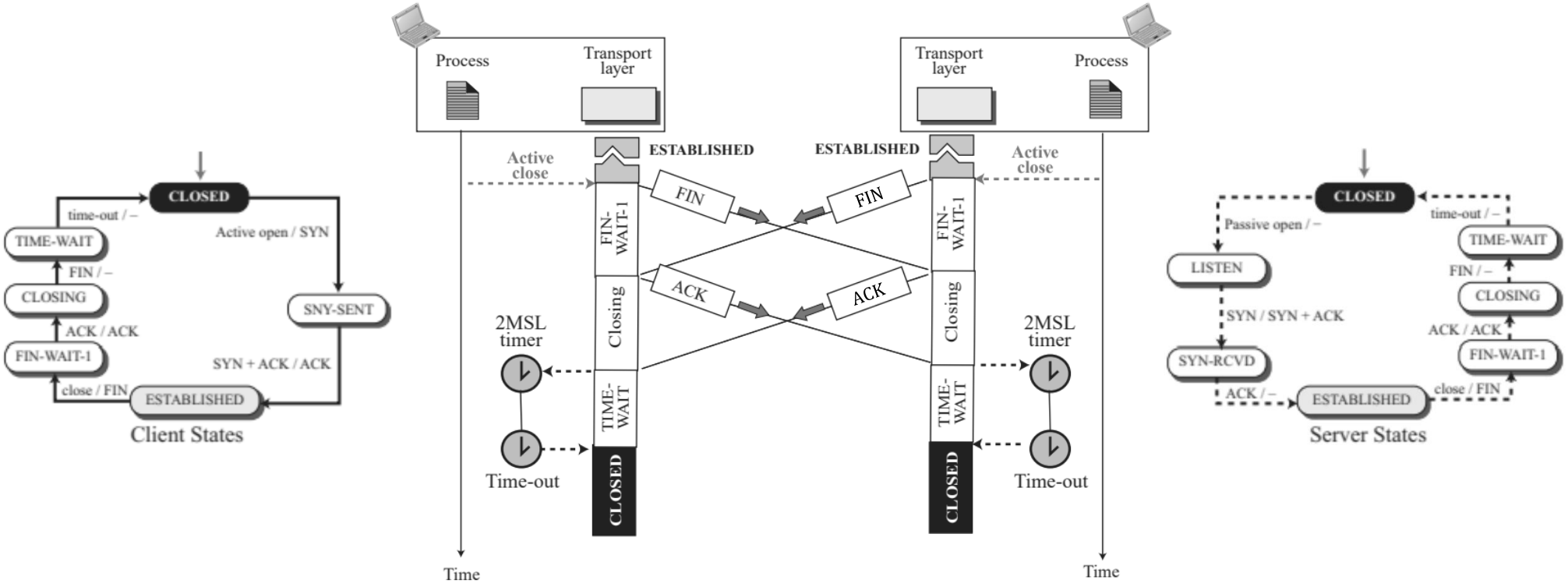






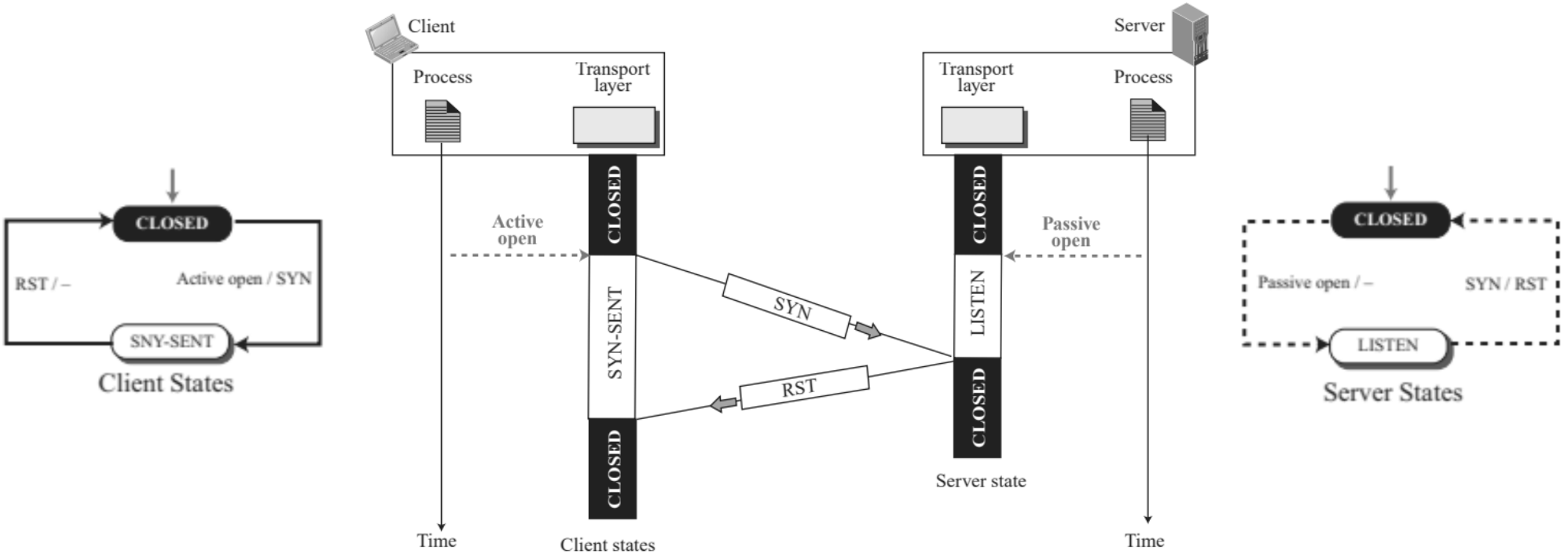
TCP 연결 및 전송

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



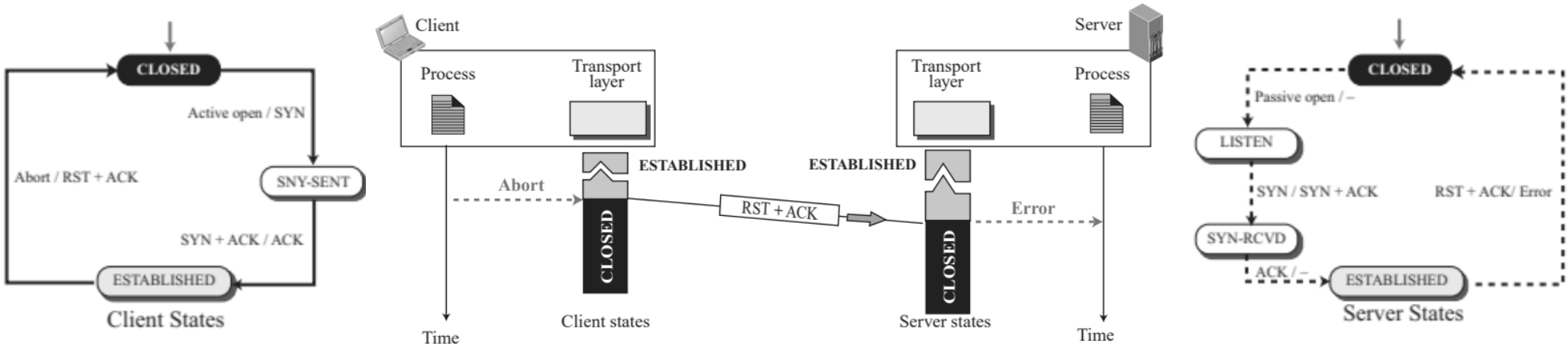
TCP 연결 및 전송

- 연결 및 전송 시나리오 (5/6)
  - 연결 거절
    - SYN 세그먼트에 있는 목적지 포트 번호가 LISTEN 상태에 있지 않은 서버를 가리키는 경우에 발생함



TCP 연결 및 전송

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



목 차

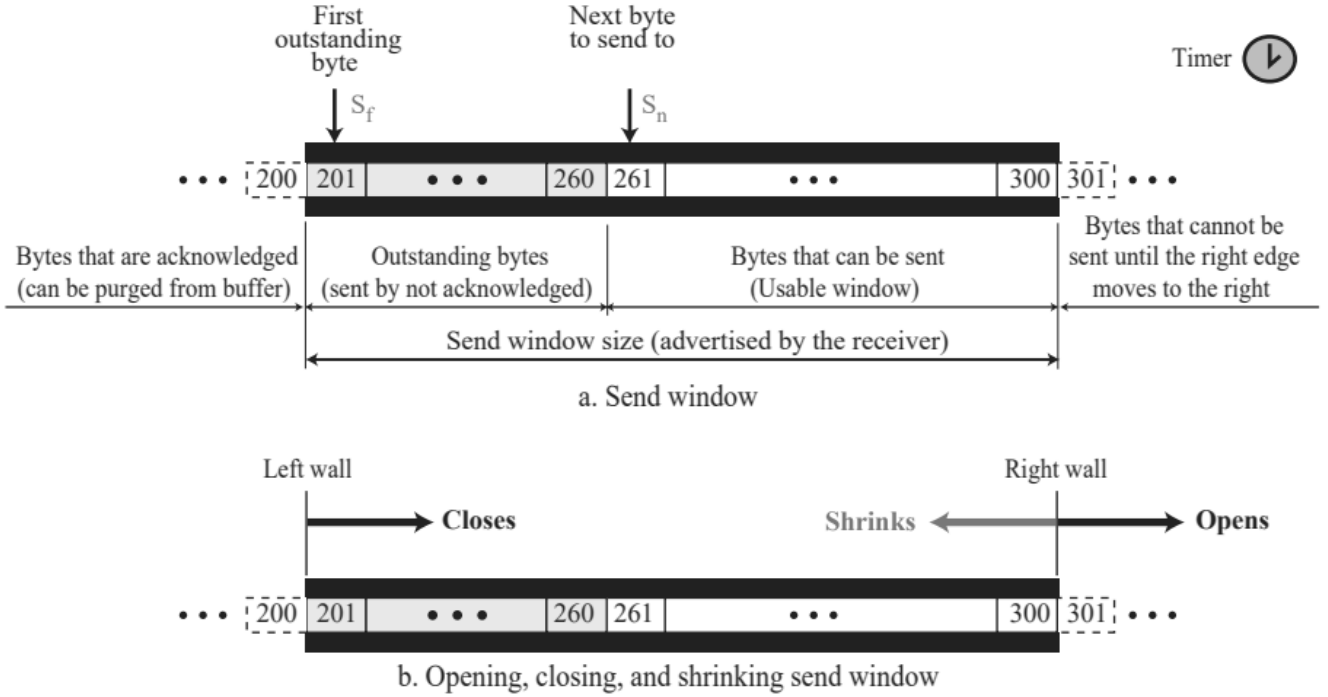
- TCP 개요
- TCP 연결 및 전송
- TCP 제어
- TCP 동작 지원 및 구현

TCP 제어

- TCP 윈도우 (1/3)
  - 정의
    - 송수신 측 버퍼에 대해 한 번에 전송하거나 수신할 수 있는 데이터의 범위
  - 특징
    - 클라이언트와 서버가 각각 송신 윈도우와 수신 윈도우를 가짐
    - 세그먼트가 아닌 바이트 단위로 동작함
    - 흐름 제어와 혼잡 제어의 핵심 메커니즘으로 사용됨

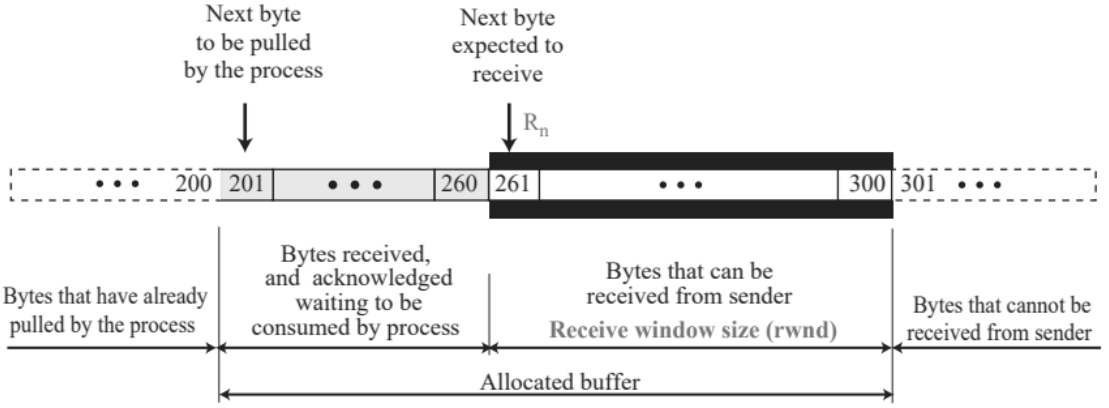
TCP 제어

- TCP 윈도우 (2/3)
  - 송신 윈도우
    - 흐름 제어에 의해 열기(Open), 닫기(Close), 축소(Shrink)가 발생할 수 있음
    - 타임아웃을 탐지하기 위해 하나의 타이머를 사용함

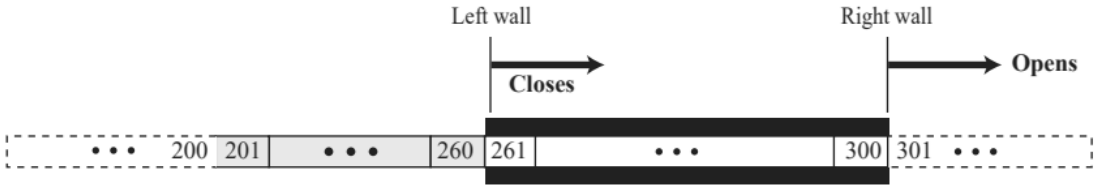


TCP 제어

- TCP 윈도우 (3/3)
  - 수신 윈도우
    - 송신 측의 전송 혼란을 방지하기 위해 축소는 허용되지 않음
    - 수신 버퍼 내 데이터는 프로세스가 읽기 전까지 저장되어 있어야 하므로 수신 버퍼를 넘치지 않고 수신할 수 있는 크기로 설정됨



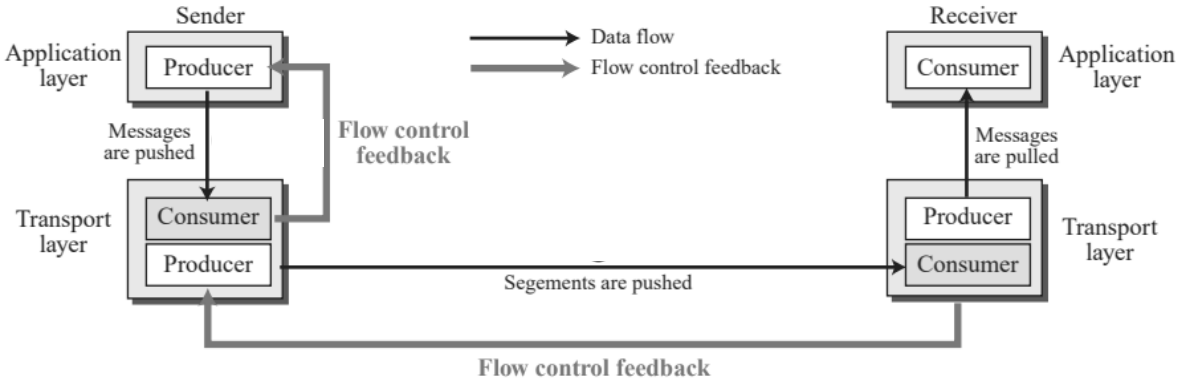
a. Receive window and allocated buffer



b. Opening and closing of receive window

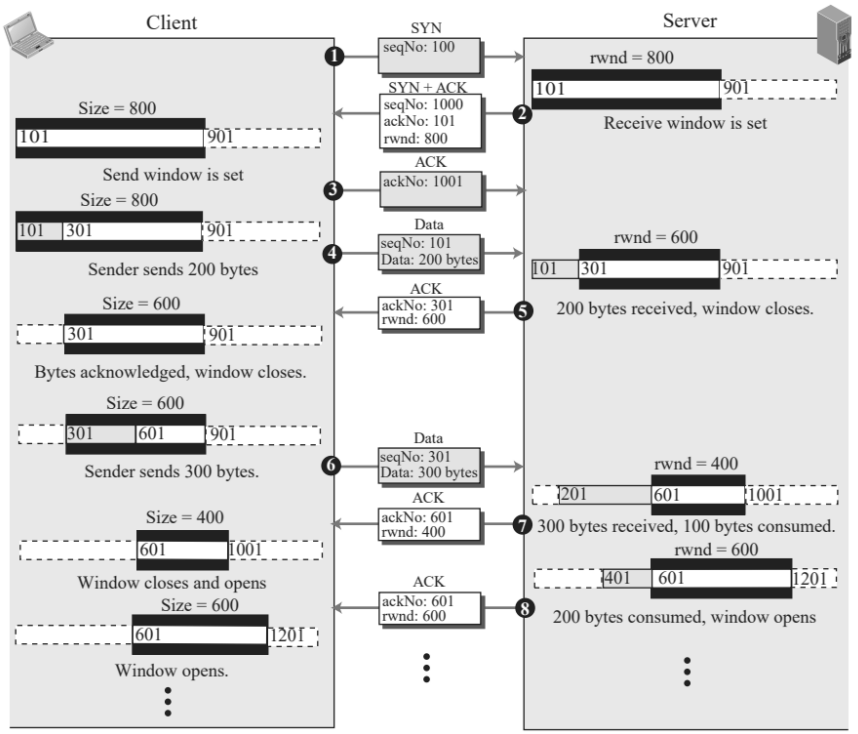
TCP 제어

- 흐름 제어 (1/7)
  - 정의
    - 송신 측이 수신 측의 처리 능력을 초과하는 속도로 데이터를 전송하지 않도록 제어하는 메커니즘
  - 동작
    - 수신 TCP는 송신 TCP에게 ACK 세그먼트를 보낼 때 rwnd를 함께 보냄으로써 송신 TCP가 전송을 제어하도록 함
    - 송신 TCP는 송신 버퍼가 가득 차면 송신 프로세스로부터 더 이상 데이터를 받지않음으로써 송신 프로세스가 전송을 제어하도록 함



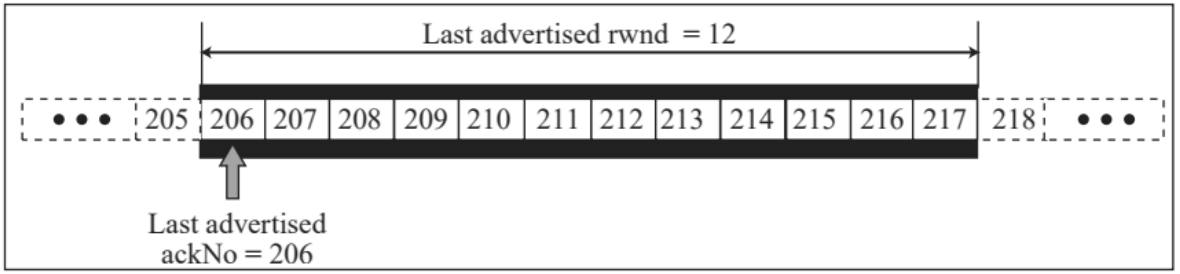
TCP 제어

- 흐름 제어 (2/7)
  - 윈도우 열기 및 닫기
    - 송신 윈도우는 새 확인응답에 의해 닫히고 rwnd에 의해 열림
    - 수신 윈도우는 송신측으로부터 바이트를 수신받으면 닫히고 수신 프로세스가 바이트를 읽으면 열림

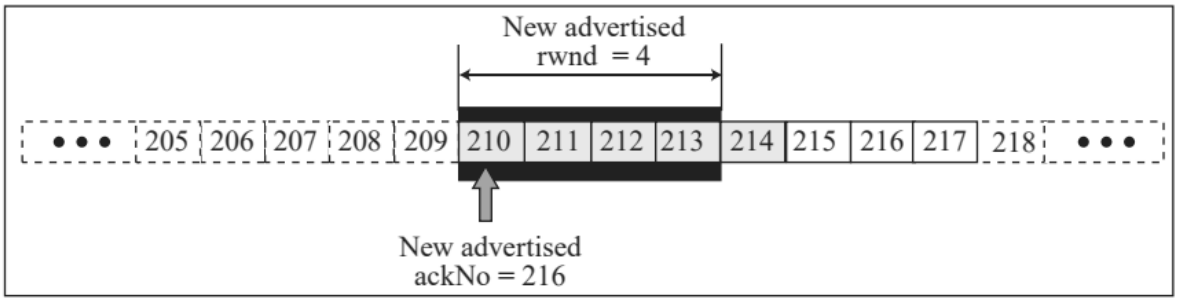


TCP 제어

- 흐름 제어 (3/7)
  - 윈도우 축소 (1/2)
    - 수신 윈도우는 축소될 수 없으나 송신 윈도우는 축소를 야기하는 rwnd 값에 의해 축소될 수 있음
    - 일반적으로 흐름 제어로 인한 축소는 송신 측의 전송 혼란을 방지하기 위해 허용되지 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)
    - 예외적으로 수신 버퍼가 가득 차는 경우, rwnd 값을 0으로 하여 일시적으로 폐쇄하는 것은 가능함
    - 수신 측에서 윈도우를 폐쇄하는 경우, 송신 측은 실제 윈도우 크기를 축소하지는 않지만 새로운 rwnd 값이 광고되기 전까지 데이터 전송을 중단함
    - 윈도우가 폐쇄된 경우에도 송신 측은 교착상태(Deadlock)을 방지하기 위해 탐침(Probe) 세그먼트를 보낼 수 있음

TCP 제어

- 흐름 제어 (5/7)
  - 어리석은 윈도우 신드롬(SWS, Silly Window Syndrome) (1/3)
    - 정의
      - 흐름 제어 과정에서 송신자와 수신자 간에 작은 단위의 세그먼트가 불필요하게 자주 오가는 비효율적인 상황
    - 특징
      - 송신 응용 프로그램이 데이터를 느리게 발생시키거나 수신 응용 프로그램이 데이터를 천천히 소비하는 경우 발생함
      - 적은 수의 데이터를 포함하는 세그먼트의 전송으로 인해 동작의 효율이 감소하게 됨

TCP 제어

- 흐름 제어 (6/7)
  - 어리석은 윈도우 신드롬(SWS, Silly Window Syndrome) (2/3)
    - 송신 측에서 발생하는 신드롬
      - 송신 응용 프로그램의 바이트 생성 속도가 느릴 경우, 적은 데이터를 포함하는 세그먼트를 전송하게 되면서 발생함
      - 해당 문제를 해결하기 위해서는 충분한 데이터가 버퍼에 쌓일 때까지 기다린 후 가능한 큰 블록으로 데이터를 전송하도록 해야함
    - 해결 방안 – Nagle 알고리즘
      1. 송신 TCP는 한 바이트라도 송신 응용 프로그램으로부터 수신하는 첫 데이터를 세그먼트로 만든 후 전송함
      2. 첫 번째 세그먼트 전송 후 수신 TCP로부터 ACK를 수신하거나 최대 크기의 세그먼트를 구성할 수 있을 정도로 충분한 데이터가 출력 버퍼에 저장되면 세그먼트로 만들어 전송함
      3. 나머지 전송 기간 동안 2번 단계를 반복함

TCP 제어

- 흐름 제어 (7/7)
  - 어리석은 윈도우 신드롬(SWS, Silly Window Syndrome) (3/3)
    - 수신 측에서 발생하는 신드롬
      - 수신 응용 프로그램의 바이트 소비 속도가 느릴 경우, 작은 rwnd를 광고하여 송신자가 적은 데이터를 포함하는 세그먼트를 전송하게 되면서 발생함
    - 해결 방안 – Clark 해결 방법
      - 수신받은 데이터에 대해 바로 ACK를 전송함
      - 수신 버퍼에 최대 크기의 세그먼트를 수용할 수 있는 공간이 있거나 수신 버퍼가 반 이상 비어있기 전까지 rwnd를 0으로 통보함
    - 해결 방안 – 지연된 확인응답(Delayed Acknowledgement)
      - 수신 버퍼에 충분한 공간이 있을 때까지 ACK를 보류함
      - 송신 측에서 ACK를 받지 못한 세그먼트를 재전송할 가능성이 존재하기 때문에 ACK는 0.5초 이상 지연되지 않도록 함

TCP 제어

- 오류 제어 (1/13)
  - 정의
    - 송신 측이 전체 스트림을 순서에 맞고 오류 없이 수신 응용 프로그램에게 전달하는 것을 보장하는 메커니즘
  - 체크섬
    - 세그먼트와 의사 헤더를 사용하여 세그먼트가 훼손되었는지 검사함
    - 세그먼트가 무효한 검사합으로 인해 훼손되면 목적지 TCP는 해당 세그먼트를 폐기하고 손실로 간주함

TCP 제어

- 오류 제어 (2/13)
  - 확인응답
    - 누적 확인응답(Accumulative Acknowledgement)
      - 순서대로 도착한 마지막 세그먼트에 대해서 확인응답함
      - TCP 헤더 내 ACK 플래그 비트가 1로 설정된 경우에 유효함
    - 선택 확인응답(SACK, Selective Acknowledgement)
      - 순서에 어긋나게 들어온 데이터 블록과 중복 세그먼트 블록을 알려줌
      - TCP 헤더 끝에 옵션의 형태로 구현됨

TCP 제어

- 오류 제어 (3/13)
  - 확인응답 – 전송 규칙 (1/2)
    1. 수신 측에서 데이터를 전송할 경우, 데이터 세그먼트에 ACK 번호를 함께 포함하여 전송함
    2. 수신 측에서 보낼 데이터가 없고 올바른 순서의 세그먼트를 받았으며, 미확인된 세그먼트가 없는 경우, 새 세그먼트를 수신하거나 일정 시간이 지날 때까지 ACK 전송을 보류함
       - e.g., 1~4의 세그먼트를 수신 받았고, 3까지의 세그먼트에 ACK를 전송한 경우, 세그먼트 4에 대한 ACK를 잠시 보류함
    3. 올바른 순서의 세그먼트를 수신하고 이전 세그먼트들이 확인응답되지 않은 경우, 즉시 ACK를 전송함
       - e.g., 세그먼트 4를 수신 받았을 때, 세그먼트 3에 대한 ACK가 전송되지 않았다면 즉시 세그먼트 4에 대한 ACK를 전송함

TCP 제어

- 오류 제어 (4/13)
  - 확인응답 – 전송 규칙 (2/2)
    4. 기대한 것보다 큰 순서번호의 세그먼트를 수신하는 경우, 자신이 수신을 기대하는 순서 번호를 ACK 번호로 하여 전송함
       - e.g., 세그먼트 2에 대한 ACK를 전송하였으나 세그먼트 4를 수신받은 경우, 세그먼트 2에 대한 ACK를 재전송함
    5. 누락된 세그먼트를 수신하면 즉시 ACK를 전송함
       - e.g., 세그먼트 2,4를 수신한 후 세그먼트 2에 대한 ACK를 전송하고 세그먼트 3을 수신받은 경우, 세그먼트 4에 대한 ACK를 전송함
    6. 중복된 세그먼트를 수신하는 경우, 해당 세그먼트를 폐기하고 ACK를 전송함
       - e.g., 1~4의 세그먼트를 수신하고 해당 세그먼트를 중복으로 수신받은 경우, 세그먼트 4에 대한 ACK를 전송함

TCP 제어

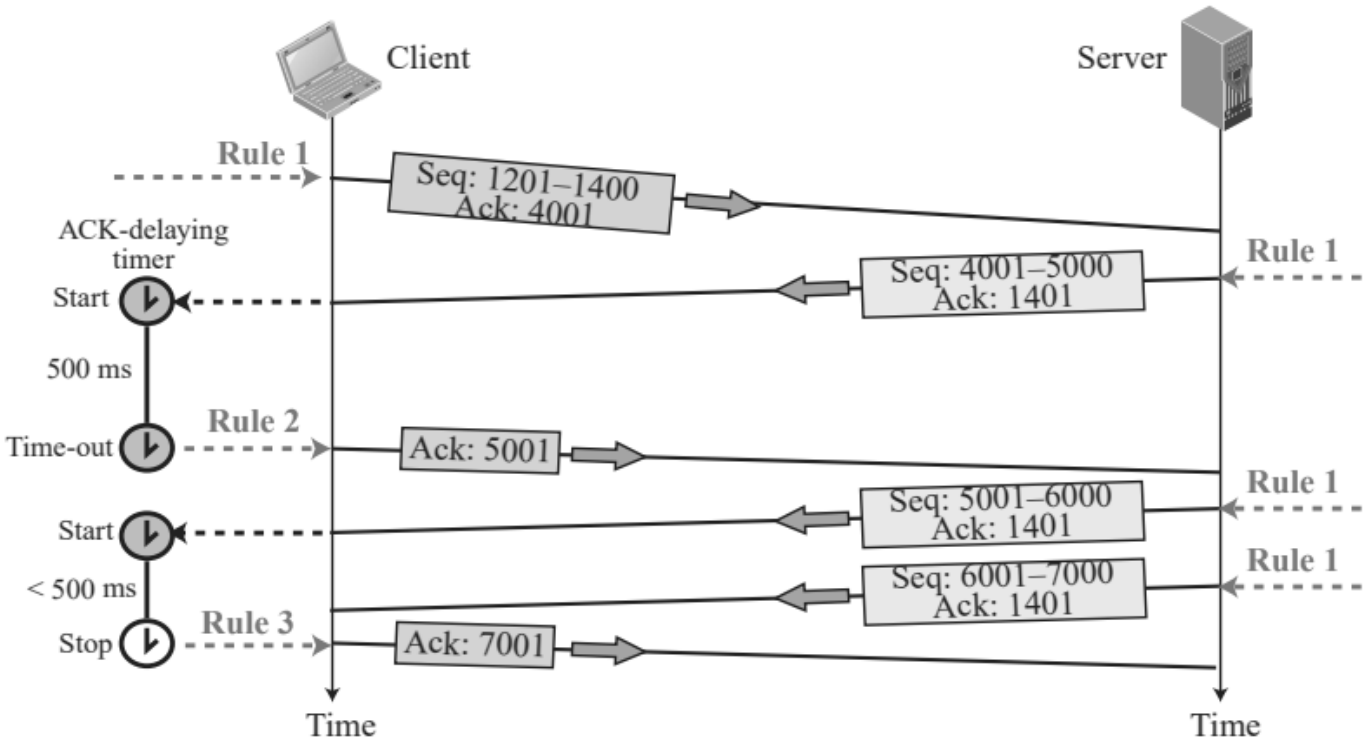
- 오류 제어 (5/13)
  - 재전송
    - 재전송 타임아웃(RTO, Retransmission Time-out) 재전송
      - 송신 TCP는 각각의 연결마다 하나의 RTO 타이머를 구동함
      - 타이머가 만료되어 타임아웃이 발생하면 TCP는 버퍼 앞에 있는 세그먼트를 전송하고 타이머를 재구동함
      - RTO의 값은 가변적이며 세그먼트의 왕복 시간(RTT, Round Trip Time)을 기반으로 업데이트됨
      - RTT는 세그먼트를 목적지로 전송하고 그 세그먼트에 대한 ACK를 수신하는데 걸리는 시간을 나타냄
    - 빠른 재전송(Fast Retransmit)
      - 하나의 세그먼트에 대한 세 개의 중복된 확인응답이 수신되면 타임아웃을 기다리지 않고 다음 세그먼트가 즉시 재전송됨

TCP 제어

- 오류 제어 (6/13)
  - 순서가 어긋난 세그먼트
    - 순서에 어긋난 세그먼트는 버려지지 않고 일시적으로 저장되며, 손실된 세그먼트가 도착하기 전까지 이 세그먼트들을 순서가 어긋난 세그먼트로 표시함
    - 순서가 어긋난 세그먼트들은 누락된 세그먼트가 도착하기 전까지 프로세스로 전달되지 않음

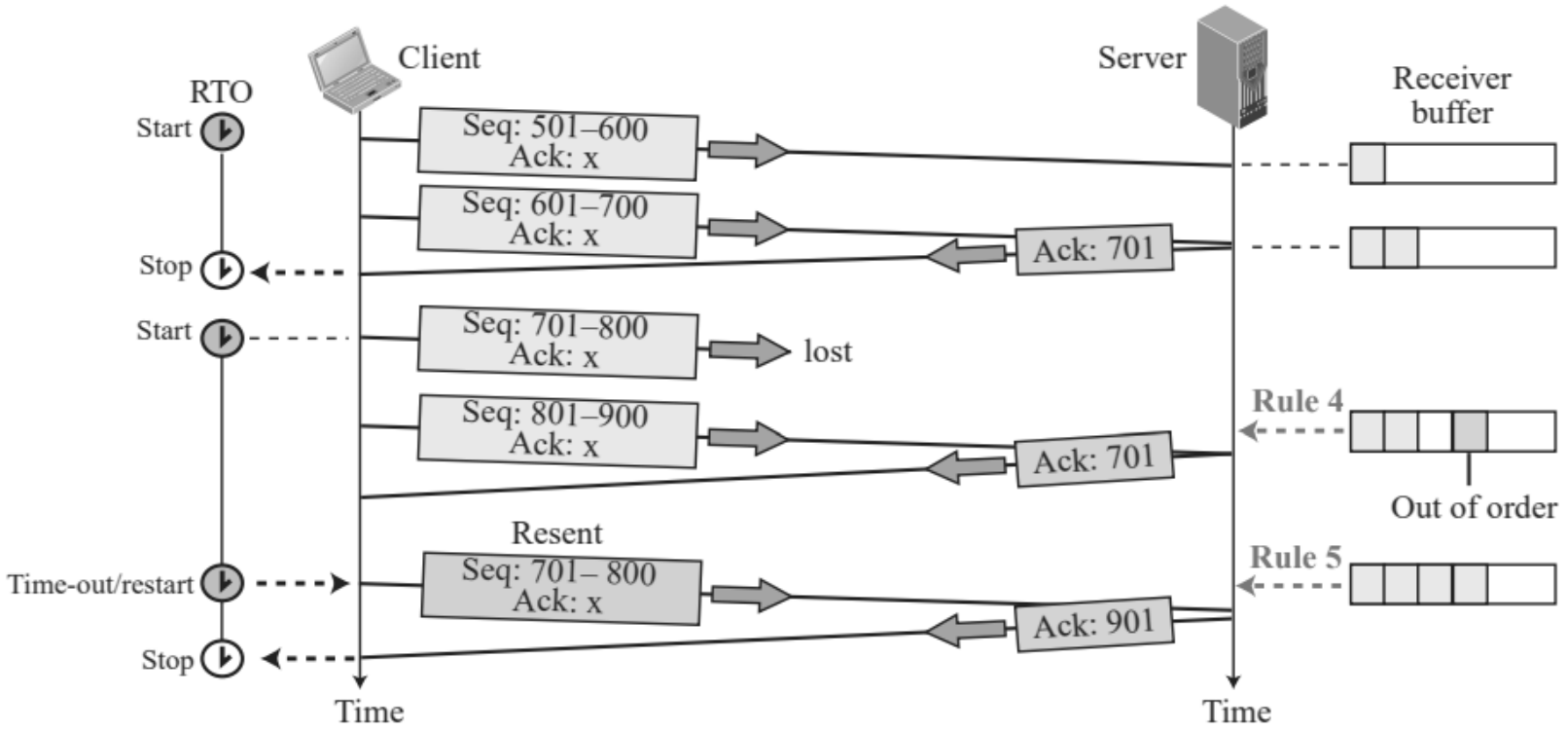
TCP 제어

- 오류 제어 (7/13)
  - 시나리오 (1/7)
    - 정상 동작
      - 전송 규칙 1에 의해 보낼 데이터가 있을 때 ACK를 함께 전송함



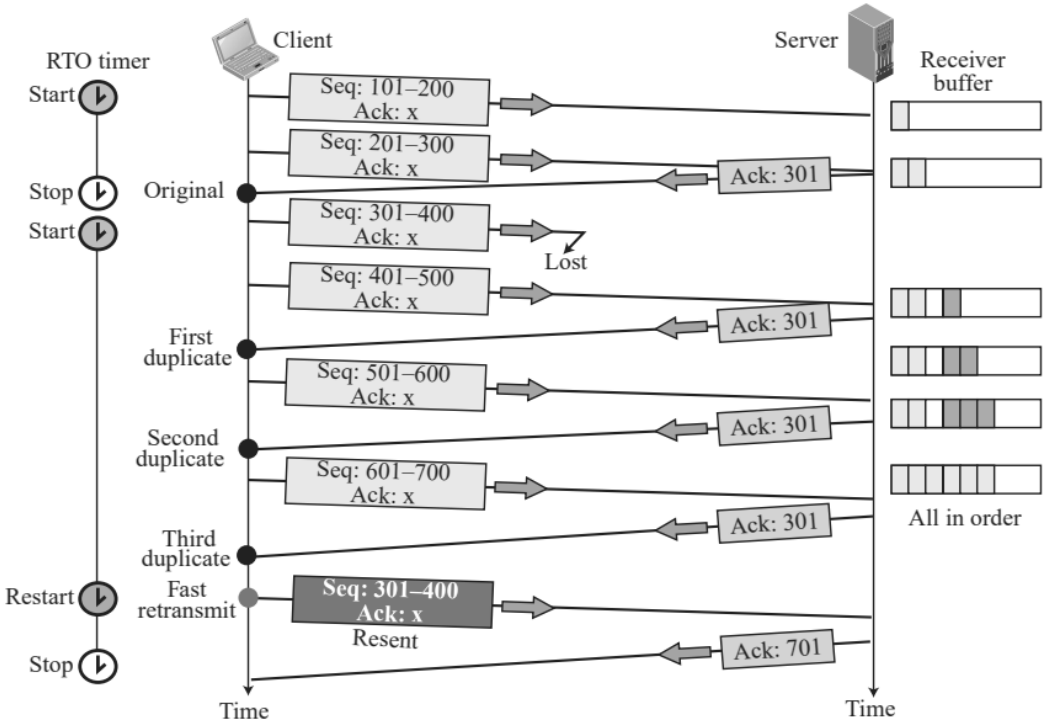
TCP 제어

- 오류 제어 (8/13)
  - 시나리오 (2/7)
    - 손실 세그먼트
      - 규칙 4에 의해 순서가 어긋난 세그먼트에 대해 즉시 ACK를 전송함
      - 규칙 5에 의해 누락된 세그먼트 수신 후 알맞은 ACK를 전송함



TCP 제어

- 오류 제어 (9/13)
  - 시나리오 (3/7)
    - 빠른 재전송
      - 송신측은 동일한 세그먼트에 대한 중복 ACK를 세 번 수신할 경우, 즉시 해당 세그먼트를 재전송함

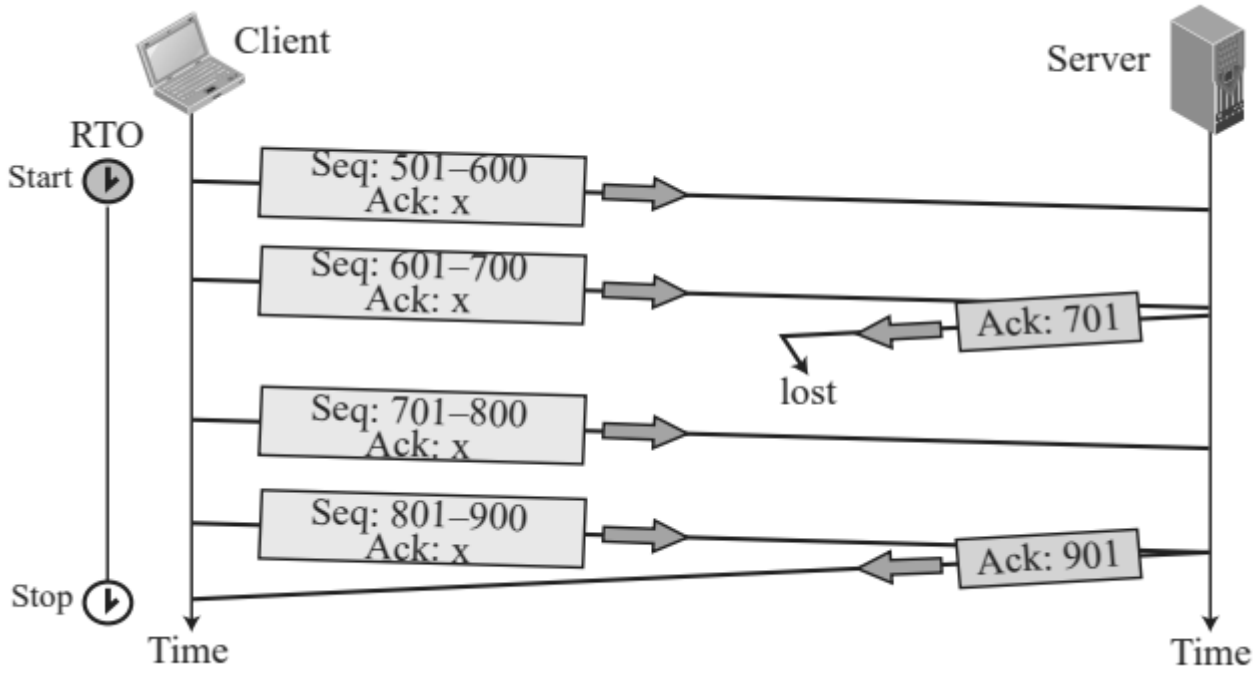


TCP 제어

- 오류 제어 (10/13)
  - 시나리오 (4/7)
    - 지연 세그먼트
      - TCP 세그먼트를 캡슐화하는 IP 데이터그램은 서로 다른 지연을 가진 서로 다른 경로로 목적지에 도착할 수 있음
      - 해당 과정에서 지연 세그먼트가 발생할 수 있고, 재전송된 세그먼트 이후에 도착한 지연 세그먼트는 중복 세그먼트로 간주되어 폐기됨
    - 중복 세그먼트
      - 세그먼트가 지연으로 인해 수신측에서 손실로 처리된 경우, 재전송에 의해 중복 세그먼트가 발생할 수 있음
      - 수신측은 중복 세그먼트를 수신하면 해당 세그먼트를 폐기하고 자신이 기대하는 ACK 번호를 가진 ACK를 전송함

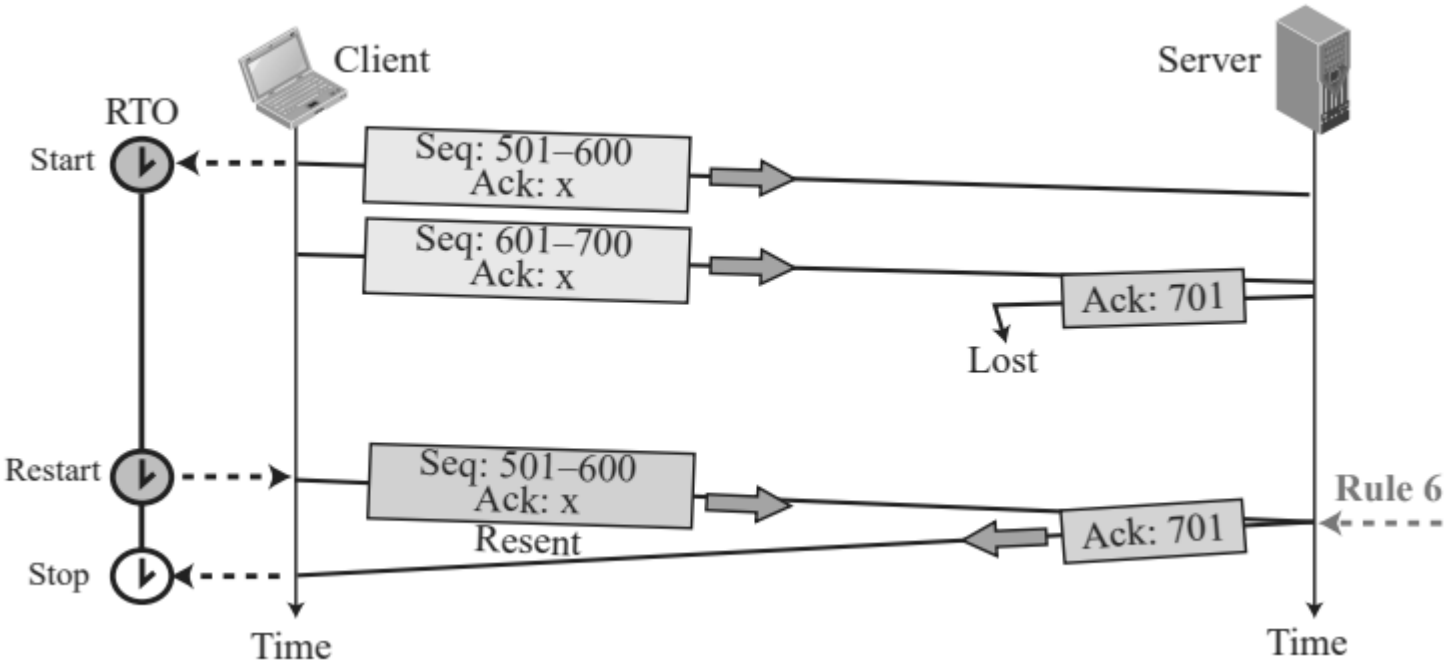
TCP 제어

- 오류 제어 (11/13)
  - 시나리오 (5/7)
    - 확인응답의 손실
      - ACK가 손실되어도 송신측으로부터 지속적으로 세그먼트를 수신 받고 있다면 자동으로 ACK 번호를 교정하여 전송함



TCP 제어

- 오류 제어 (12/13)
  - 시나리오 (6/7)
    - 확인응답의 손실
      - ACK가 오랜 시간 전달되지 못 하면 RTO에 의해 세그먼트가 재전송됨
      - 규칙 6에 의해 중복된 세그먼트를 폐기하고 즉시 알맞은 ACK를 전송함



TCP 제어

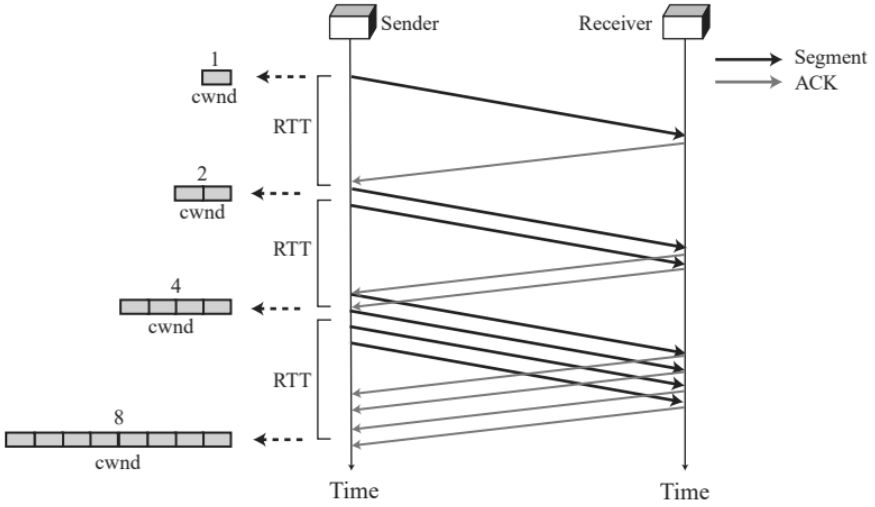
- 오류 제어 (13/13)
  - 시나리오 (7/7)
    - 확인응답의 손실로 발생하는 교착 상태
      - 수신측에서 윈도우 폐쇄 후 다시 윈도우를 여는 경우, 이를 위해 전송한 ACK가 손실되면 교착 상태에 빠질 수 있음
        - 수신측은 ACK를 전송하였고 ACK에는 RTO가 존재하지 않으므로 수신측은 송신측이 세그먼트를 보낼 때까지 기다림
        - 송신측은 윈도우 폐쇄에 의해 전송을 억제하고 있으므로 수신측이 윈도우를 열 때까지 세그먼트를 전송하지 않음
      - 이를 해결하기 위해 영속 타이머(Persistence Timer)와 탐침(Probe) 세그먼트를 활용함
        - 영속 타이머가 만료될 때마다 ACK를 받기 위해 1 바이트의 데이터를 담은 탐침 세그먼트를 전송함

TCP 제어

- 혼잡 제어 (1/4)
  - 정의
    - 네트워크 혼잡을 방지하기 위해 송신 측 전송 속도를 조절하는 메커니즘
  - 혼잡 윈도우
    - 네트워크가 송신측에서 데이터를 발생하는 속도보다 늦게 데이터를 전달한다면 네트워크는 송신측의 데이터 발생 속도를 늦추도록 해야 함
    - 송신자는 수신자에 의해 광고되는 윈도우 크기(rwnd)와 혼잡 윈도우 크기의 두 가지 정보를 가지고 있으며, 윈도우의 실제 크기는 두 정보의 최소값임

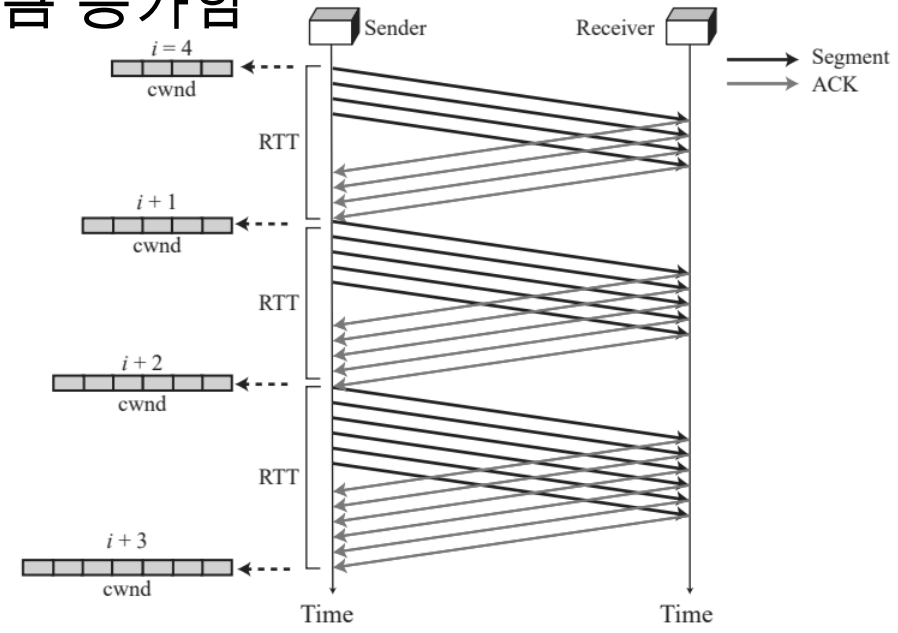
TCP 제어

- 혼잡 제어 (2/4)
  - 혼잡 제어 원칙 (1/3)
    - 느린 시작(Slow Start) : 지수 증가(Exponential Increase)
      - 혼잡 윈도우(cwnd)의 크기는 하나의 최대 세그먼트 크기인 MSS (Maximum Segment Size)에서부터 시작함
      - 윈도우의 크기는 하나의 확인응답이 도착할 때마다 하나의 MSS 만큼 증가하여 지수적으로 증가함
      - 느린 시작은 무한정 계속될 수 없고 이 단계를 중지할 느린 시작 임계치(ssthresh, Slow Start Threshold)라는 변수가 존재함



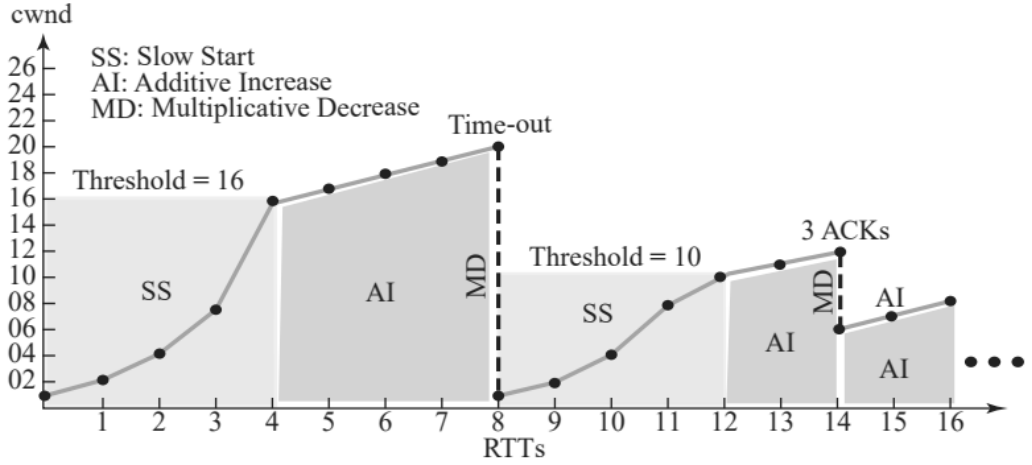
TCP 제어

- 혼잡 제어 (3/4)
  - 혼잡 제어 원칙 (2/3)
    - 혼잡 회피(Congestion avoidance) : 가산 증가(Additive Increase)
      - 혼잡 윈도우의 크기가 ssthresh에 도달하면 느린 시작 단계는 종료되고 가산 단계가 시작됨
      - 전체 윈도우만큼의 세그먼트가 확인응답될 때마다 혼잡 윈도우의 값은 1만큼 증가함



TCP 제어

- 혼잡 제어 (4/4)
  - 혼잡 제어 원칙 (3/3)
    - 혼잡 감지(Congestion Detection) : 지수 감소(Multiplicative Decrease)
      - RTO가 발생했다면 혼잡이 일어났을 가능성이 높으므로, 임계치 값을 현재 윈도우 크기의 반으로 설정하고 cwnd를 하나의 세그먼트 크기로 줄인 후 느린 시작 단계부터 다시 시작함
      - 빠른 재전송이 발생했다면 혼잡이 일어날 가능성이 적으므로, 빠른 복구(Fast Recovery)를 수행하여 임계치 값을 현재 윈도우 크기의 반으로 설정하고 cwnd 값을 임계치 값으로 설정한 후 혼잡 회피 단계를 시작함

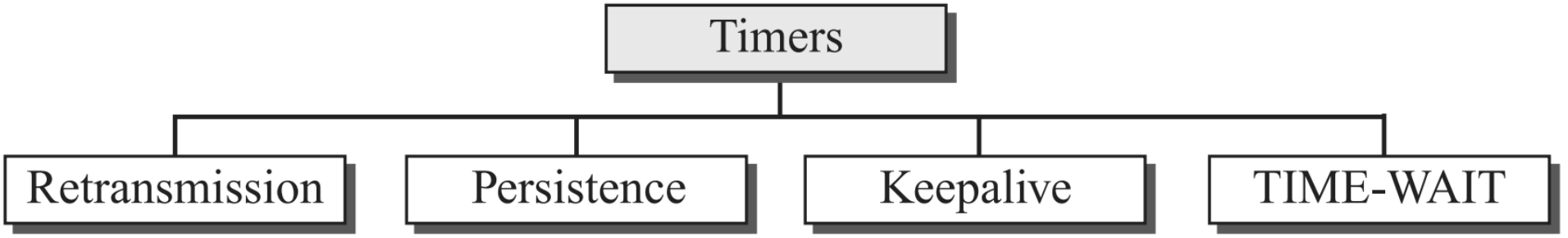


목 차

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

TCP 동작 지원 및 구현

- 타이머 (1/10)
  - 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 지원 및 구현

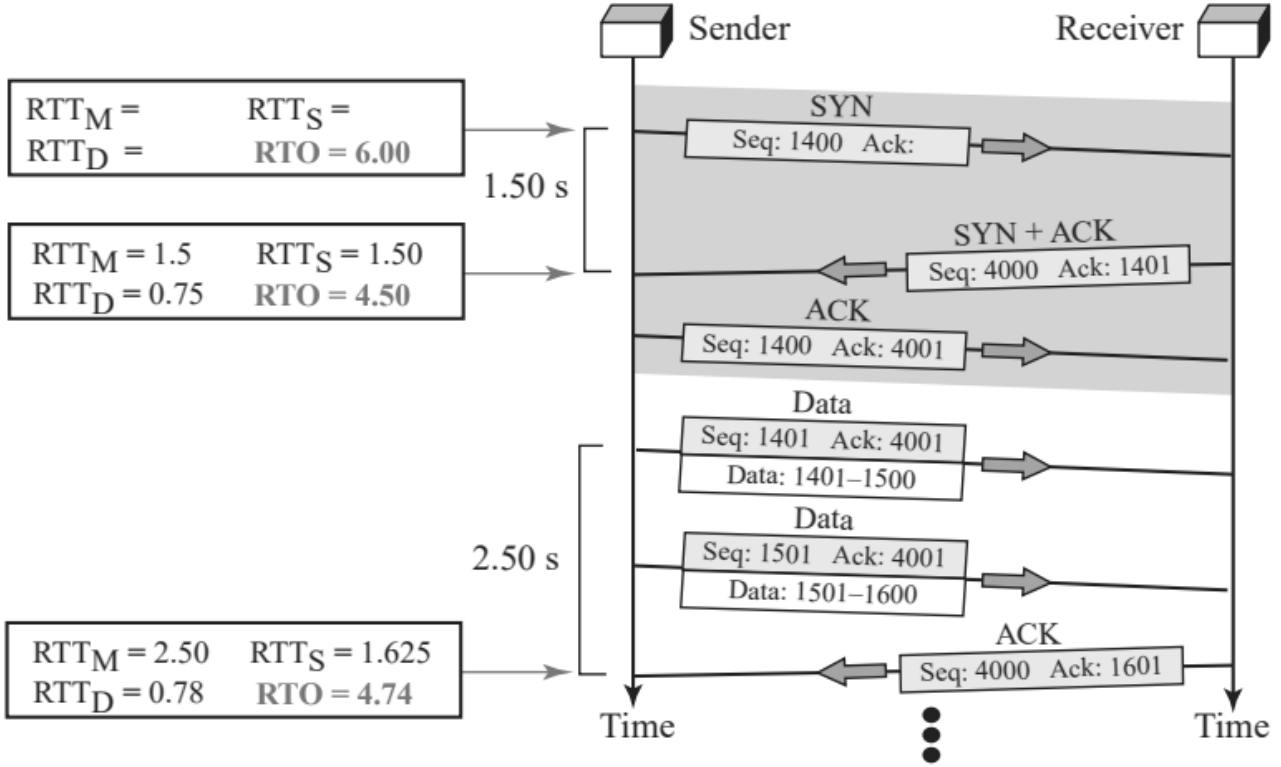
- 타이머 (2/10)
  - 재전송 타이머 (1/5)
    - 손실된 세그먼트의 재전송을 위해 TCP는 한 세그먼트에 대한 확인응답을 기다리는 시간인 RTO 값으로 설정되는 타이머를 가지고 있음
    - 송신 버퍼의 맨 앞에 있는 세그먼트를 전송할 때 타이머를 구동함
    - 타이머가 만료되면 버퍼의 앞에 있는 첫 번째 세그먼트를 재전송하며 타이머를 재구동함
    - 누적 확인응답된 세그먼트들은 버퍼에서 삭제하고 버퍼가 빈 경우에는 타이머를 정지하고 그렇지 않으면 재구동함

TCP 동작 지원 및 구현

- 타이머 (3/10)
  - 재전송 타이머 (2/5)
    - MRTT (Measured Round Trip Time)
      - 실제 측정된 RTT 값으로, 세그먼트를 전송하고 그 세그먼트에 대한 확인응답을 수신하는 데 얼마나 오랜 시간이 걸리는지 측정함
    - SRTT (Smooth Round Trip Time)
      - 측정된 RTT는 네트워크 상태에 따라 크게 변하므로 MRTT와 이전 SRTT의 가중 평균인 $SRTT = (1-\alpha)SRTT + \alpha \times MRTT$를 사용함
        - α는 주로 1/8로 설정됨
        - 첫 번째 측정 시엔 $SRTT = MRTT$로 계산함
    - RTTVAR (RTT Variation)
      - MRTT의 변동성을 반영하기 위해 $RTTVAR = (1-\beta)RTTVAR + \beta \times |SRTT - MRTT|$를 사용함
        - β는 주로 1/4로 설정됨
        - 첫 번째 측정 시엔 $RTTVAR = MRTT/2$로 계산함

TCP 동작 지원 및 구현

- 타이머 (4/10)
  - 재전송 타이머 (3/5)
    - RTO
      - SRTT, RTTVAR을 기반으로 $RTO = SRTT + 4 \times RTTVAR$로 계산함

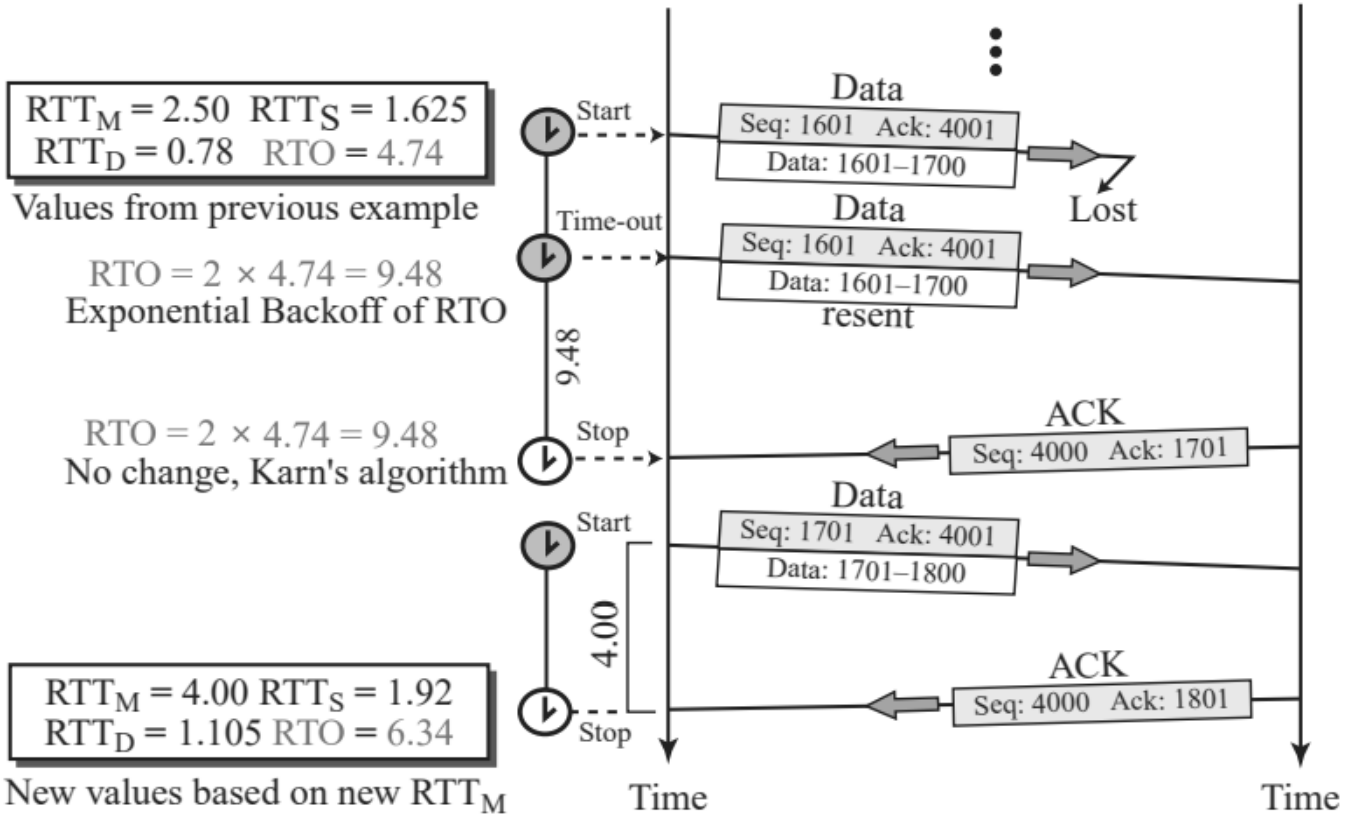


TCP 동작 지원 및 구현

- 타이머 (5/10)
  - 재전송 타이머 (4/5)
    - Karn 알고리즘
      - RTO 이후 재전송된 세그먼트에 대한 ACK를 수신했을 때, 송신 TCP는 이 확인응답이 원래 세그먼트에 대한 ACK인지 재전송 세그먼트에 대한 ACK인지 알 수 없음
      - 원래의 세그먼트가 손실되고 수신한 ACK가 재전송 세그먼트에 대한 확인이면 현재 RTT 값은 세그먼트가 재전송된 시간부터 계산되어야 함
      - Karn은 이런 모호한 부분을 해결하기 위해 새로운 RTT 계산에 재전송 세그먼트의 RTT는 고려하지 않음
      - 세그먼트를 전송하고 재전송 없이 확인응답을 수신하기 전까지는 RTT를 갱신하지 않음

TCP 동작 지원 및 구현

- 타이머 (6/10)
  - 재전송 타이머 (5/5)
    - 지수 백오프(Exponential Backoff)
      - 세그먼트가 재전송 될 때마다 RTO 값을 두 배로 설정함



TCP 동작 지원 및 구현

- 타이머 (7/10)
  - 영속 타이머 (1/2)
    - 개요
      - 수신 TCP가 윈도우 폐쇄를 통보한 경우, 송신 TCP는 윈도우 폐쇄가 끝남을 알리는 ACK를 수신하기 전까지 세그먼트의 전송을 보류함
      - 수신 TCP의 ACK가 손실되면 송신 TCP는 수신 TCP가 윈도우 크기를 알리는 확인응답을 전송하기만을 기다림
      - 두 TCP는 서로의 응답만을 기다리다가 교착 상태에 빠짐

TCP 동작 지원 및 구현

- 타이머 (8/10)
  - 영속 타이머 (2/2)
    - 과정
      1. 송신 TCP가 윈도우 폐쇄를 통보하는 ACK를 수신하면 송신 TCP는 영속 타이머의 값을 재전송 시간의 값으로 하여 구동함
      2. 영속 타이머가 만료되면 송신 TCP는 프로브(Probe) 세그먼트를 전송함
      3. 수신측으로부터 ACK를 수신하지 못하면 다른 프로브 세그먼트가 전송되며 영속 타이머 값을 두 배로 하여 재구동함
      4. 송신측은 프로브 세그먼트의 전송과 영속 타이머 값 설정을 타이머 값이 임계치(Threshold)에 도달할 때까지 반복함
      5. 이후 윈도우가 재개시 될 때까지 매 60 초마다 하나의 프로브 세그먼트를 전송함

TCP 동작 지원 및 구현

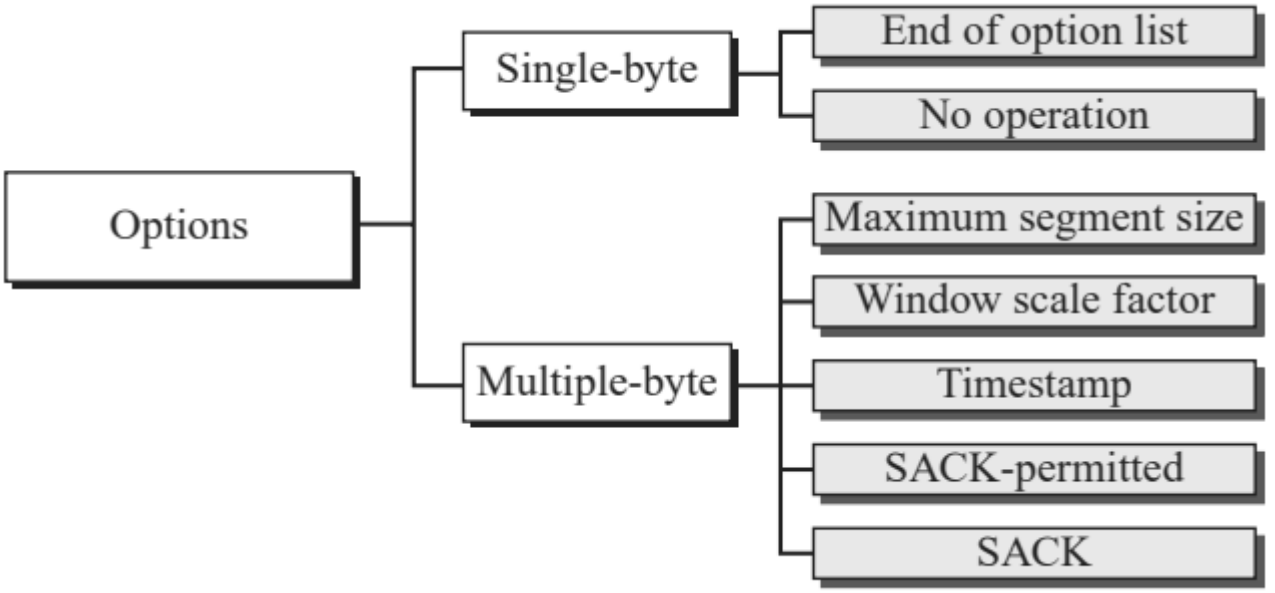
- 타이머 (9/10)
  - 킵얼라이브 타이머
    - 두 TCP 사이에 설정된 연결이 오랜 기간 동안 휴지(Idle) 상태에 있는 것을 방지하기 위해 사용됨
    - 서버가 클라이언트로부터 세그먼트를 받을 때마다 서버는 이 타이머를 초기화함
    - 타임아웃 기간은 일반적으로 2시간임
    - 서버가 2시간이 지나도록 클라이언트로부터 세그먼트를 수신하지 못하면 서버는 프로브 세그먼트를 전송함
    - 각각 75초의 시간 간격으로 10개의 프로브를 전송했는데도 응답을 수신하지 못하면 서버는 클라이언트가 다운되었다고 간주하고 연결을 종료함

TCP 동작 지원 및 구현

- 타이머 (10/10)
  - 시간 대기 타이머(TIME-WAIT Timer, 2MSL Timer)
    - 연결 종료 과정에서 TIME-WAIT 상태일 때 사용함
    - TIME-WAIT 상태 전에 보낸 ACK가 손실되면 서버는 자신이 보낸 FIN 세그먼트의 손실로 판단하고 재전송하는데, 이 때 2 MSL의 시간을 대기하여 서버에게 ACK를 전송할 수 있게 함
    - 이전 연결과 동일한 주소에 새로운 연결을 열었을 때, 이전 연결의 중복 세그먼트가 도착하지 않도록 2 MSL의 시간을 대기하여 이전 연결의 세그먼트가 소멸할 시간을 보장함

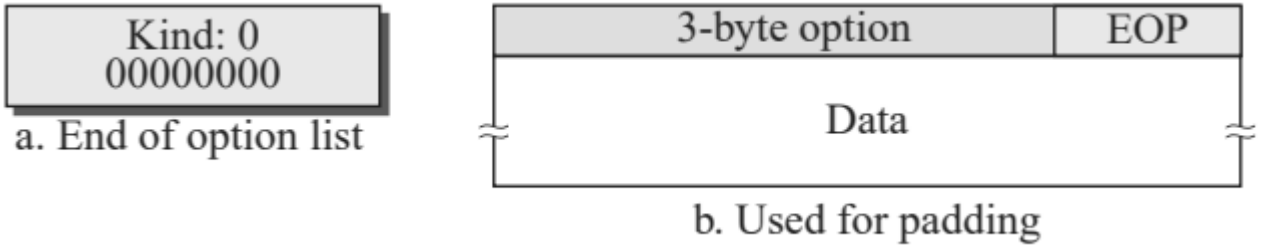
TCP 동작 지원 및 구현

- 옵션 (1/13)
  - 개요
    - TCP 헤더에서는 최대 40 바이트의 옵션 정보가 있을 수 있음
    - 이러한 옵션 정보들은 목적지에게 부가 정보를 전달하거나 다른 옵션의 정렬을 맞추기 위해 사용됨



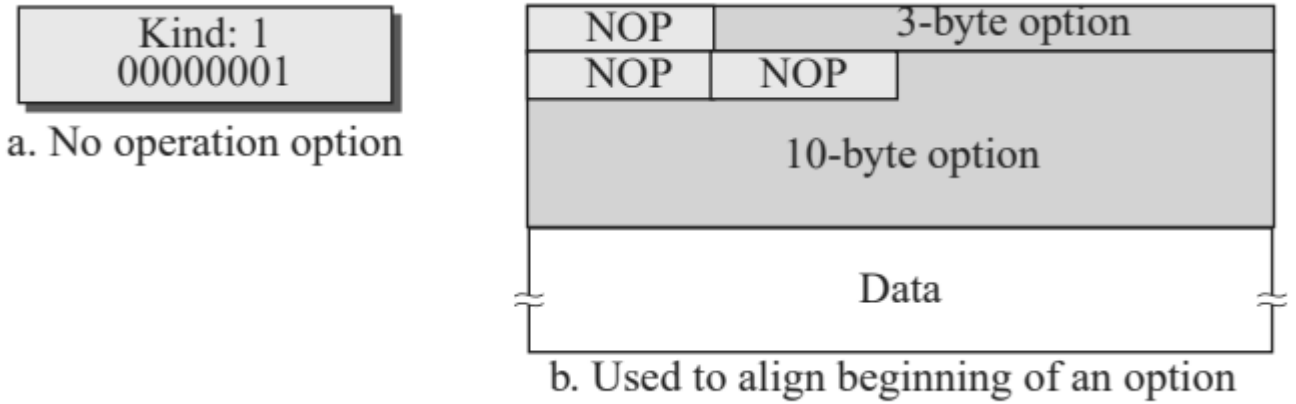
TCP 동작 지원 및 구현

- 옵션 (2/13)
  - 단일 바이트(Single-byte) 옵션 (1/2)
    - 옵션 종료(EOP, End-of-Option)
      - 옵션 구간 끝에 패딩을 위해 사용되는 단일 바이트 옵션
      - 마지막 옵션으로서만 사용될 수 있으며 한 번만 사용될 수 있음
      - 다른 옵션 사용 후 EOP 옵션이 다음 워드 길이의 경계로 데이터를 정렬하기 위해 추가됨



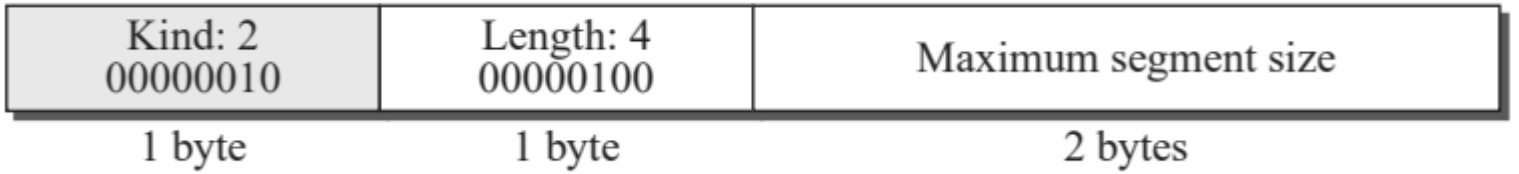
TCP 동작 지원 및 구현

- 옵션 (3/13)
  - 단일 바이트(Single-byte) 옵션 (2/2)
    - 무동작(NOP, No-Operation)
      - 옵션 사이의 채우기(Filler)로서 사용되는 단일 바이트 옵션
      - 일반적으로 다른 옵션 앞에 위치하여 그 옵션이 4 바이트의 배수가 되도록 함



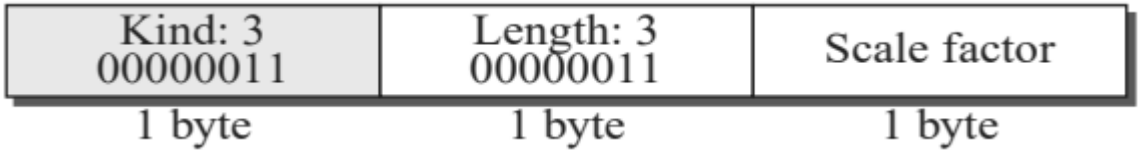
TCP 동작 지원 및 구현

- 옵션 (4/13)
  - 다중 바이트(Multiple-byte) 옵션 (1/10)
    - 최대 세그먼트 크기(MSS, Maximum Segment Size)
      - TCP 세그먼트의 목적지에서 수신할 수 있는 데이터 세그먼트의 최대 크기를 정의함
      - MSS는 세그먼트 내 데이터 부분의 최대 크기임
      - 이 필드는 MSS를 바이트 단위로 나타내는 16 비트 필드이므로 MSS는 0~65535 바이트의 값을 가질 수 있음
      - MSS는 연결 설정 단계에서 결정됨
      - 각 TCP는 자신이 TCP 연결 동안 수신하고자 하는 세그먼트의 MSS를 결정함
      - 크기를 결정하지 않으면 기본값(Default)인 536으로 정의됨



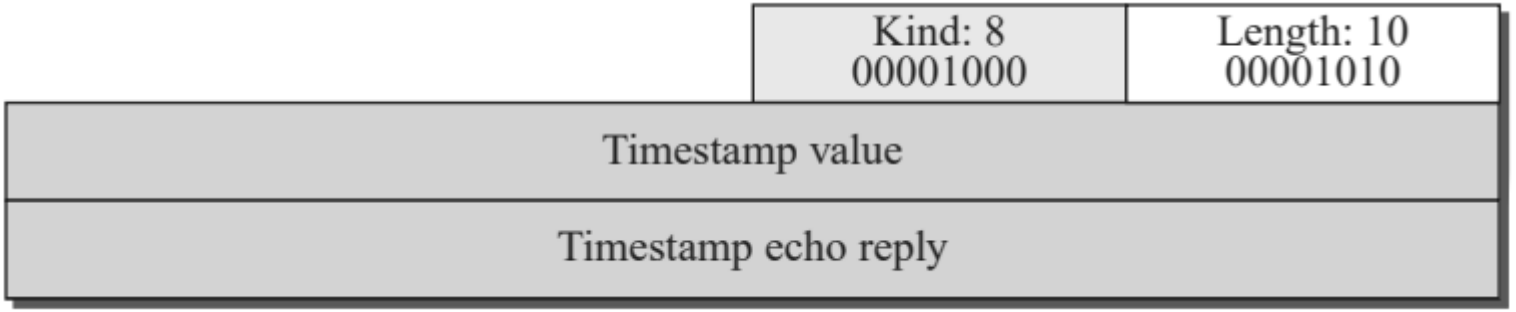
TCP 동작 지원 및 구현

- 옵션 (5/13)
  - 다중 바이트(Multiple-byte) 옵션 (2/10)
    - 윈도우 확장 인자(Window Scale Factor)
      - 헤더에 있는 윈도우 크기 필드는 16 비트 필드로, 0~65535 바이트의 범위를 가질 수 있음
      - 이 크기가 충분하지 않을 때 윈도우 크기를 증가시키기 위해 윈도우 확장 인자 옵션이 사용됨
      - 새로운 윈도우 크기를 구하기 위해 윈도우 확장 인자에 명시된 수의 2 승수를 구하고 해당 결과를 헤더에 있는 윈도우 크기 값에 곱하여 새로운 윈도우 크기를 결정함
      - 새로운 윈도우 크기가 순서 번호 최대값의 절반을 초과하지 않도록 윈도우 확장 인자의 최대값은 14로 제한됨
      - 윈도우 확장 인자는 연결 설정 단계에서만 결정될 수 있음



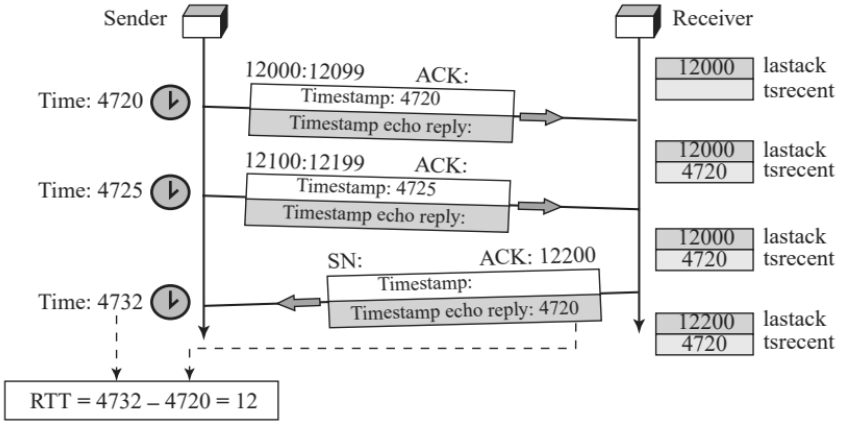
TCP 동작 지원 및 구현

- 옵션 (6/13)
  - 다중 바이트(Multiple-byte) 옵션 (3/10)
    - 타임스탬프(Timestamp) (1/4)
      - 타임스탬프 값과 타임스탬프 에코 응답을 포함하는 10 바이트 옵션
      - 능동 개방을 요청한 호스트는 SYN 세그먼트 내에 타임스탬프를 알림
      - 상대로부터 다음 세그먼트(SYN+ACK)에 있는 타임스탬프를 수신하면 타임스탬프 사용이 허락됐음을 의미함
      - 타임스탬프가 없는 경우, 더 이상 타임스탬프 옵션을 사용할 수 없음



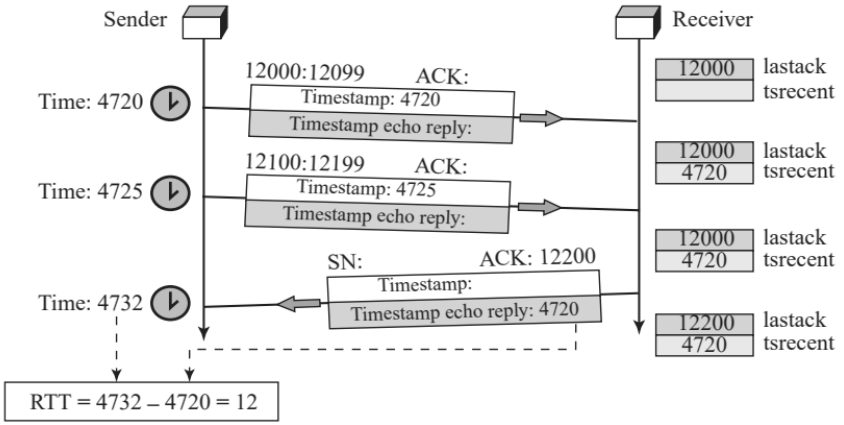
TCP 동작 지원 및 구현

- 옵션 (7/13)
  - 다중 바이트(Multiple-byte) 옵션 (4/10)
    - 타임스탬프(Timestamp) (2/4)
      - RTT 측정
        - 송신 TCP는 시스템 클록의 값을 타임스탬프 값 필드에 삽입함
        - 해당 세그먼트에 대한 ACK나 세그먼트에 있는 바이트를 포함하는 누적 확인 응답을 전송하는 수신측은 수신한 타임스탬프 값을 타임스탬프 에코 응답에 복사함
        - 수신측에서는 마지막으로 전송한 확인응답 값 lastack와 아직 에코되지 않은 최근 타임스탬프 값 tsrecent를 기억하여 lastack 값과 동일한 바이트를 포함하는 세그먼트를 수신하면 타임스탬프 필드에 있는 값을 tsrecent 변수에 넣음



TCP 동작 지원 및 구현

- 옵션 (8/13)
  - 다중 바이트(Multiple-byte) 옵션 (5/10)
    - 타임스탬프(Timestamp) (3/4)
      - RTT 측정
        - ACK를 전송할 때 수신자는 tsrecent 값을 에코 응답 필드에 넣음
        - ACK를 수신한 송신자는 클록 값에서 타임스탬프 에코 응답에 있는 값을 뺀 값을 RTT로 계산함



TCP 동작 지원 및 구현

- 옵션 (9/13)
  - 다중 바이트(Multiple-byte) 옵션 (6/10)
    - 타임스탬프(Timestamp) (4/4)
      - 순서 번호 겹침 방지(PAWS, Protection Against Wrapped Sequence Number)
        - 순서 번호는 32 비트 길이로, 고속 연결에서는 순서가 겹칠 수 있음
        - 세그먼트가 다음 라운드에 도착하면 이전 라운드의 세그먼트가 새로운 라운드에 속하는 세그먼트로 잘못 받아들여질 수 있음
        - 세그먼트 확인자에 타임스탬프를 넣음으로써 세그먼트의 신원은 타임스탬프와 순서 번호의 합에 의해 확인될 수 있음

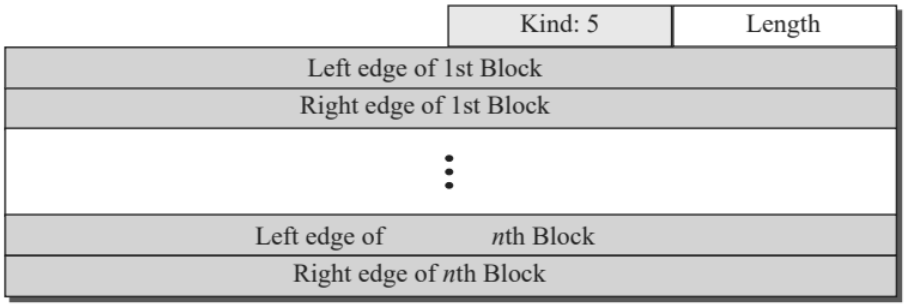
TCP 동작 지원 및 구현

- 옵션 (10/13)
  - 다중 바이트(Multiple-byte) 옵션 (7/10)
    - SACK-허용 및 SACK 옵션 (1/4)
      - 누적 확인응답 방식에서 수신측은 순서에 어긋나게 들어오는 바이트와 세그먼트 중복에 대해서 알려주지 않음
      - 일부 세그먼트가 손실되면 송신측에서는 타임아웃이 발생할 때까지 기다린 후 확인응답 되지 않은 모든 세그먼트들을 재전송하며 수신측에서는 중복 세그먼트를 받을 수 있음
      - SACK은 송신측이 어느 세그먼트가 없어지거나 순서에 어긋나게 도착했는지에 대해 더 자세히 알 수 있도록 함
      - 송신측은 중복 세그먼트 리스트를 이용하여 어떤 세그먼트가 짧은 타임아웃에 의해 재전송 됐는지를 알 수 있음

TCP 동작 지원 및 구현

- 옵션 (11/13)
  - 다중 바이트(Multiple-byte) 옵션 (8/10)
    - SACK-허용 및 SACK 옵션 (2/4)
      - SACK-허용 옵션은 2 바이트의 길이를 가지며 연결 설정 과정에서만 사용할 수 있음
        - SYN 세그먼트를 전송하는 호스트는 자신이 SACK 옵션을 제공할 수 있음을 알리기 위해 해당 옵션을 SYN 세그먼트에 추가함
        - 상대가 SYN+ACK 세그먼트에 해당 옵션은 추가하면 양쪽 모두 데이터 전송 기간 동안 SACK 옵션을 사용 가능함
      - SACK 옵션은 가변 길이를 가지며 양쪽 모두 사용하기로 했을 때 데이터 전송 단계에서만 사용 가능함
        - 해당 옵션은 순서에 어긋나게 들어온 블록 리스트를 포함함

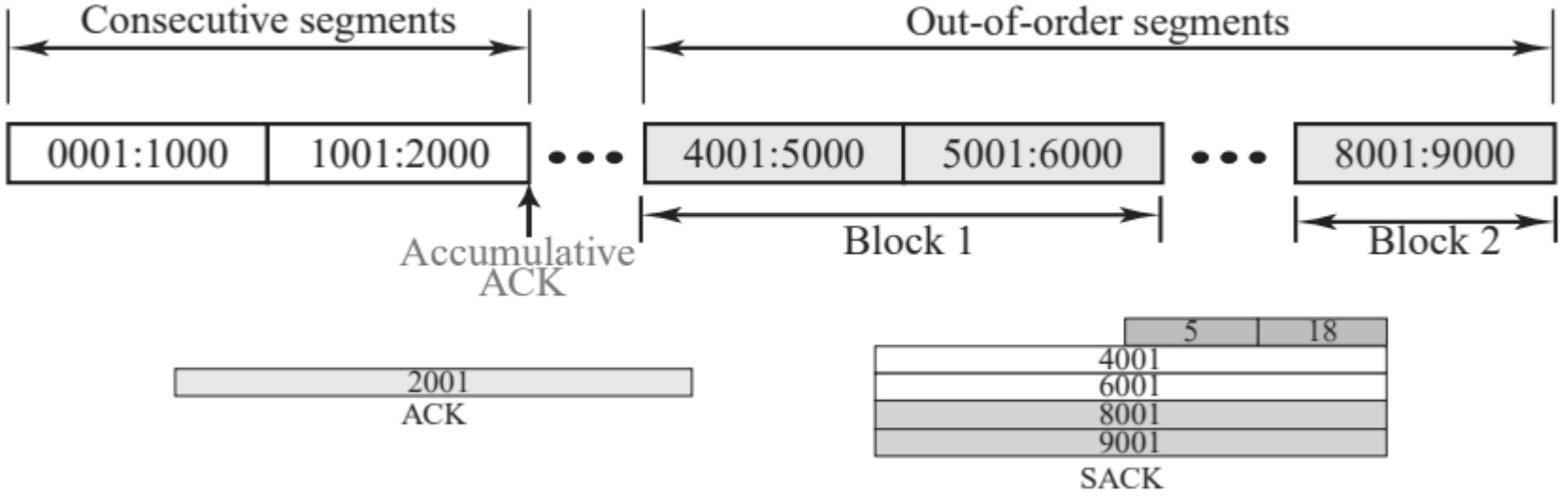
SACK-permitted option



SACK option

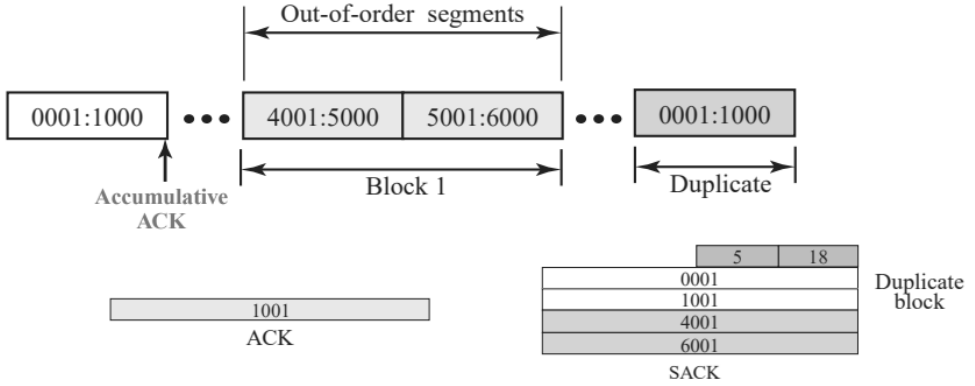
TCP 동작 지원 및 구현

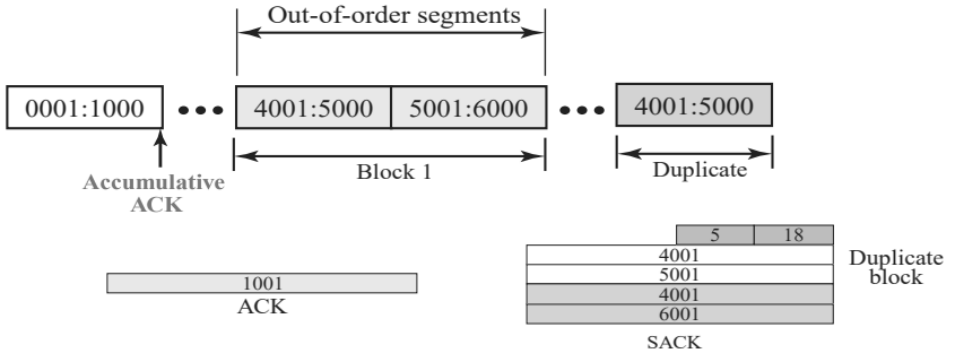
- 옵션 (12/13)
  - 다중 바이트(Multiple-byte) 옵션 (9/10)
    - SACK-허용 및 SACK 옵션 (3/4)
      - 각 블록은 시작과 끝을 나타내는 두 개의 32 비트 번호를 사용함
      - TCP에서 허용되는 옵션의 최대 길이는 40 바이트이므로 SACK 옵션은 최대 4 블록까지만 정의될 수 있음



TCP 동작 지원 및 구현

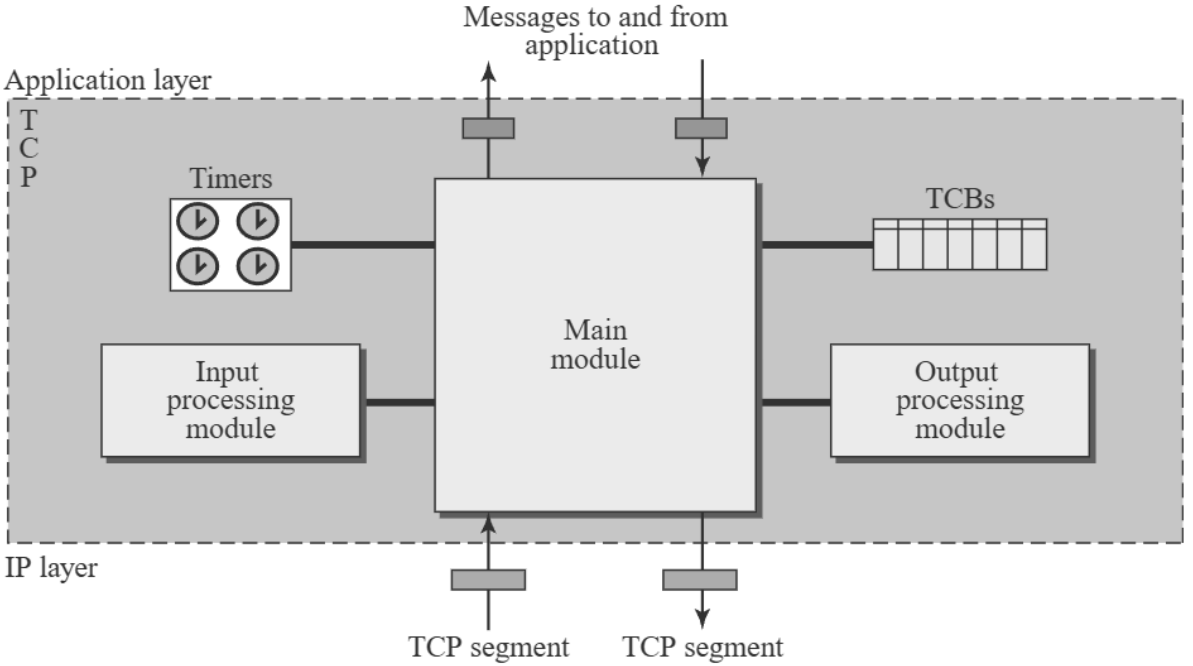
- 옵션 (13/13)
  - 다중 바이트(Multiple-byte) 옵션 (10/10)
    - SACK-허용 및 SACK 옵션 (4/4)
      - 첫 번째 블록의 바이트 번호가 이미 ACK를 전송한 바이트 번호라면 해당 블록은 중복 수신한 세그먼트를 나타냄
      - 블록 범위의 중복을 통해 순서 번호가 어긋난 세그먼트의 중복을 알 수 있음





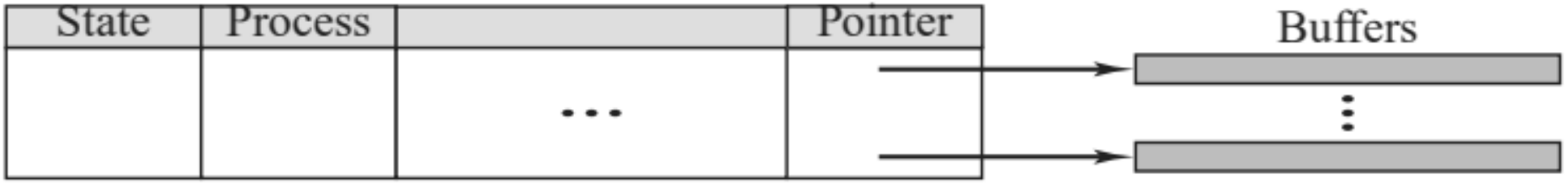
TCP 동작 지원 및 구현

- TCP 패키지 (1/6)
  - 정의
    - TCP 구현을 위한 메커니즘의 집합
  - 구성요소
    - 전송 제어 블록
    - 타이머
    - 메인 모듈
    - 입력 처리 모듈
    - 출력 처리 모듈



TCP 동작 지원 및 구현

- TCP 패키지 (2/6)
  - 전송 제어 블록(TCB, Transmission Control Block) (1/2)
    - 정의
      - 연결 제어를 위해 각 연결에 대한 정보를 보관하는 구조
    - 구성 요소
      - 상태 정보
      - 식별자 정보
      - 송수신 윈도우 관리 정보
      - 순서 번호 정보
      - 타이머 정보



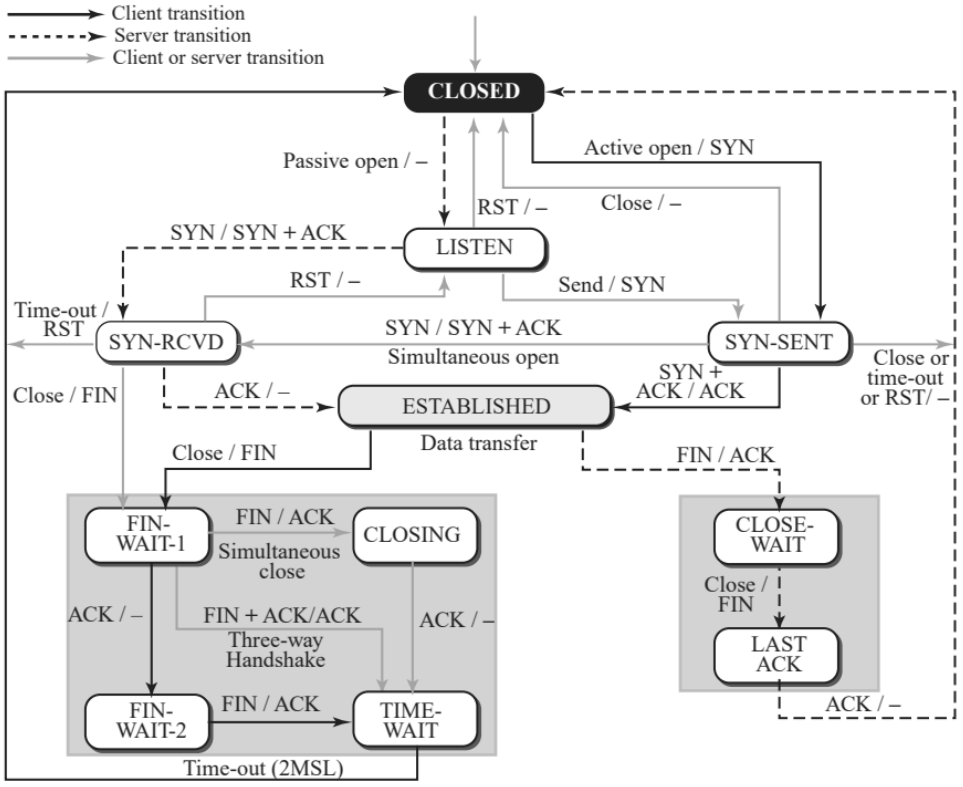
TCP 동작 지원 및 구현

- TCP 패키지 (3/6)
  - 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

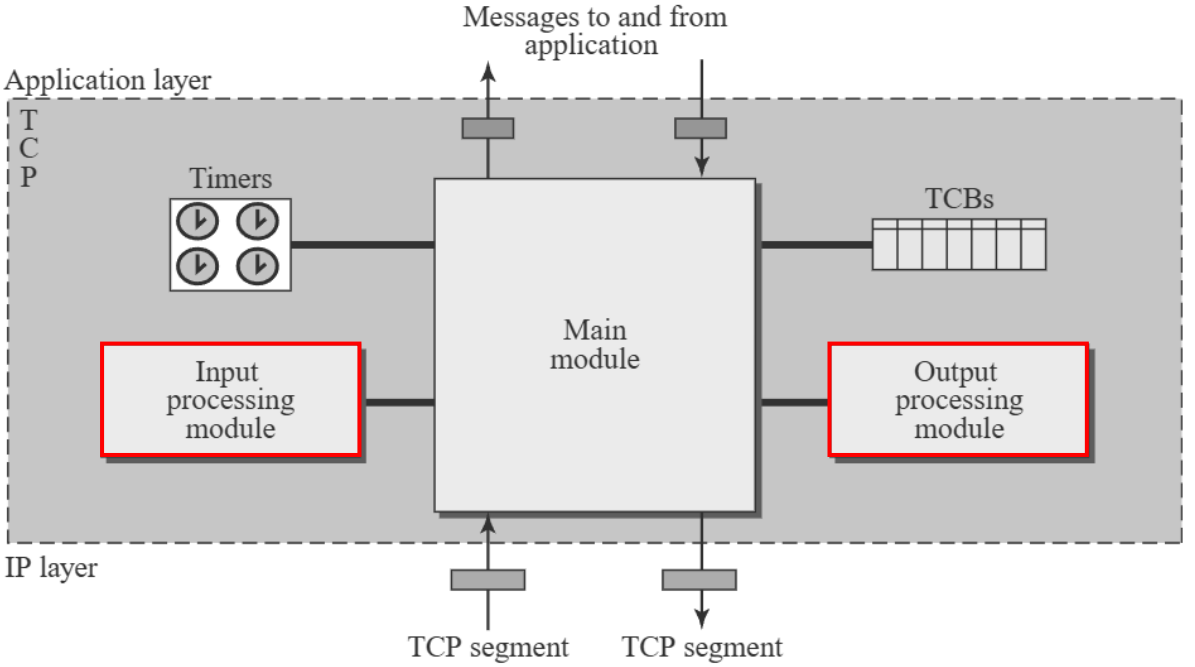
TCP 동작 지원 및 구현

- TCP 패키지 (4/6)
  - 메인 모듈 (1/2)
    - 세그먼트가 도착하거나, 타임아웃 이벤트가 발생하거나, 응용 프로그램으로부터 메시지가 들어오면 실행됨
    - 메인 모듈은 실행될 때마다 취해야 하는 행동이 TCP의 현재 상태에 따라 다름
    - 11개의 상태에 대해 서로 다른 Case를 사용함



TCP 동작 지원 및 구현

- TCP 패키지 (5/6)
  - 메인 모듈 (2/2)
    - ESTABLISHED 상태에 있고 데이터나 확인 세그먼트가 도착하면 입력 처리 모듈이 해당 상황을 처리하기 위해 호출됨
    - ESTABLISHED 상태에 있고 응용 프로그램으로부터 데이터 전송 메시지가 발생하면 출력 처리 모듈이 해당 상황을 처리하기 위해 호출됨



TCP 동작 지원 및 구현

- TCP 패키지 (6/6)
  - 입력 처리 모듈
    - ESTABLISHED 상태에 있을 때 수신하는 데이터나 ACK를 처리하기 위해 필요한 모든 기능을 담당함
    - 필요한 경우 ACK를 전송하고 윈도우 크기 알림을 처리하고 오류 확인 등을 담당함
  - 출력 처리 모듈
    - ESTABLISHED 상태에 있을 때 응용 프로그램으로부터 수신한 데이터를 전송하기 위해 필요한 모든 기능을 담당함
    - 재전송 타임아웃, 영속 타임아웃 등과 같은 기능을 담당함
  - 타이머
    - 재전송 타이머, 영속 타이머, 킵얼라이브 타이머, 시간 대기 타이머를 구동하고 타이머 만료 시 필요한 모듈에게 알림

Thanks!

이 성 현(seonghyeon@pel.sejong.ac.kr)