멀티플렉싱

1. 멀티플렉싱 개념
2. 멀티플렉서?
3. 멀티플렉싱 종류별 설명
4. 퀴즈

안효성

멀티플렉싱이란?

멀티플렉싱(Multiplexing)은 **여러 개의 입력 데이터를 하나의 경로로 통합**하는 것
디멀티플렉싱(Demultiplexing)은 **하나의 입력 데이터를 여러 대상으로 분리하여 전달**하는 것

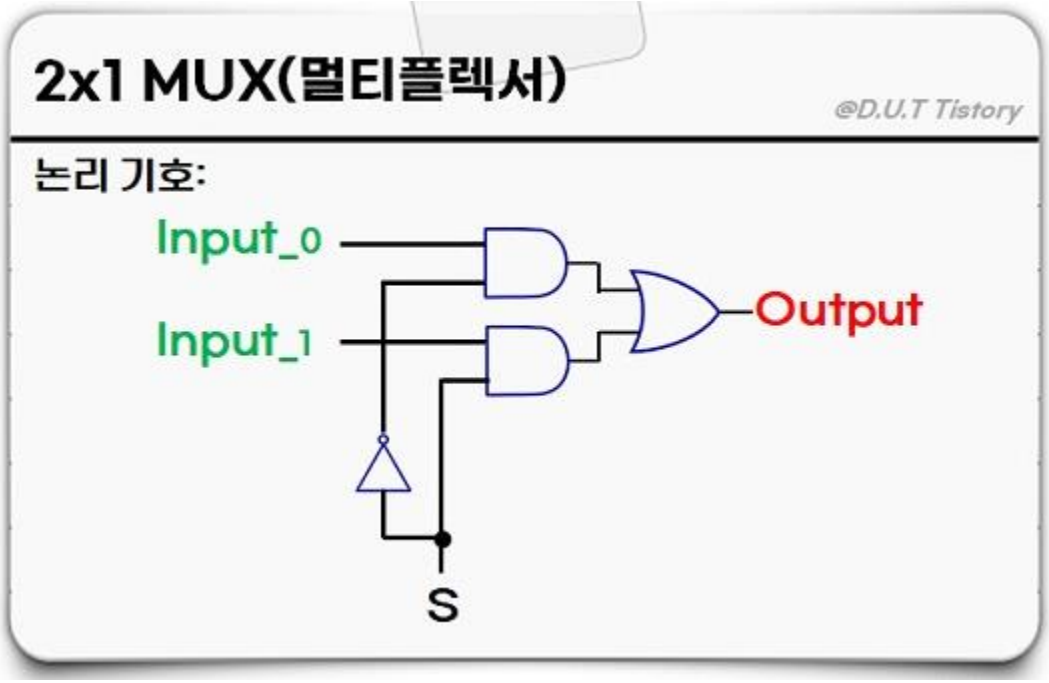
- 멀티플렉싱: N개의 입력 → 1개의 출력 (N → 1)
- 디멀티플렉싱: 1개의 입력 → N개의 출력 (1 → N)

멀티플렉싱의 개념은

1. FDM/TDM과 같은 물리적인 계층
2. 전송계층
3. 서버 프로그래밍(I/O 멀티플렉싱)
4. 웹통신 (HTTP2.0에서) 등 다양한 곳에서 활용되고 있는 개념입니다

컴구에서는 MUX

멀티플렉서



S가 0이면 Output은 Input_0과 동일하고, S가 1이면 Input_1과 동일

select

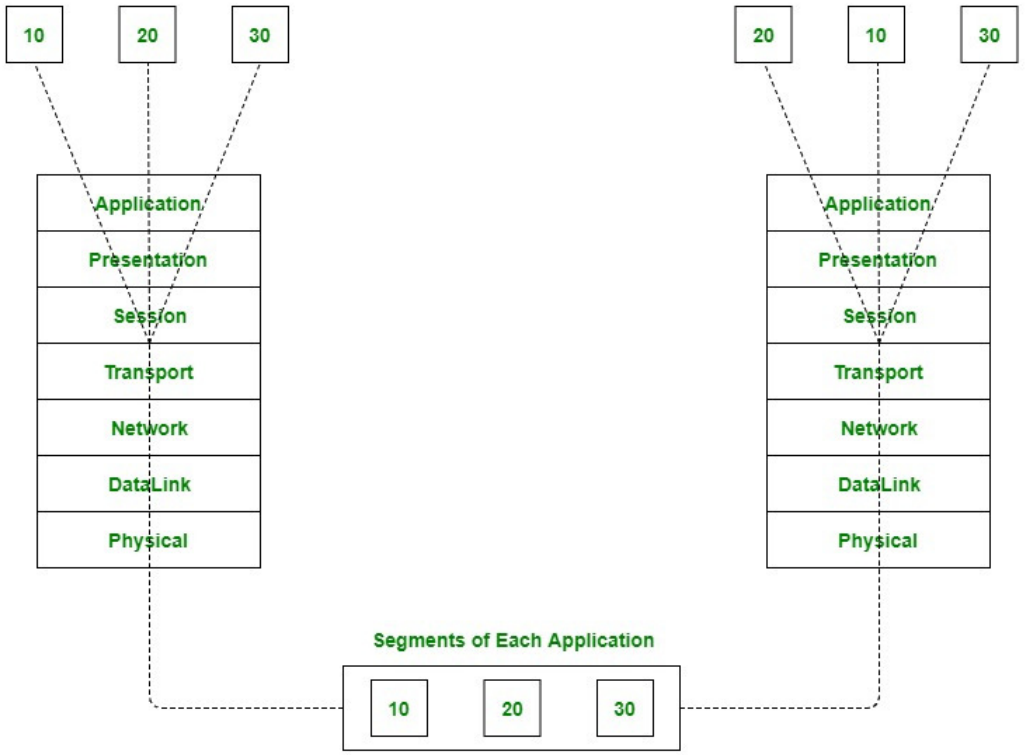
MUX는 selector를 통해 Input을 선별

combine

일반적으로 네트워크에서 말하는 멀티 플렉싱은 select보다는 combine에 가까움

1. 전송계층에서 멀티플렉싱 포트 번호가 존재 하는 이유!

전송 계층의 가장 기본적인 역할은 네트워크 계층이 제공한 호스트 간의 연결을 프로세스 간의 연결로 확장시키는 것.
송신 측에서 여러 소켓의 데이터를 '한 IP'를 통해 보낼 수 있게 하는 것



Sender의 multiplexing: 여러 개의 소켓을 통해 넘어온 데이터를 TCP의 경우 segment, UDP는 datagram로 만들고 포트번호 등 헤더를 붙여 네트워크 계층으로 전달

Receiver의 demultiplexing: 헤더의 내용을 떼서, 이걸 참조해서 맞는 소켓에 갖다 넣어줌.

1. 전송계층에서 멀티플렉싱 포트 번호가 존재 하는 이유!

```
C:₩Users₩Administrator>netstat -n

활성 연결

  프로토콜  로컬 주소          외부 주소          상태
  TCP    127.0.0.1:3452         127.0.0.1:49902        ESTABLISHED
  TCP    127.0.0.1:49670        127.0.0.1:49671        ESTABLISHED
  TCP    127.0.0.1:49671        127.0.0.1:49670        ESTABLISHED
  TCP    127.0.0.1:49672        127.0.0.1:49673        ESTABLISHED
  TCP    127.0.0.1:49673        127.0.0.1:49672        ESTABLISHED
  TCP    127.0.0.1:49675        127.0.0.1:49676        ESTABLISHED
  TCP    127.0.0.1:49676        127.0.0.1:49675        ESTABLISHED
  TCP    127.0.0.1:49677        127.0.0.1:49678        ESTABLISHED
  TCP    127.0.0.1:49678        127.0.0.1:49677        ESTABLISHED
  TCP    127.0.0.1:49902        127.0.0.1:3452         ESTABLISHED
  TCP    192.168.0.5:49790      4.213.25.241:443       ESTABLISHED
  TCP    192.168.0.5:50076      108.177.125.188:5228   ESTABLISHED
  TCP    192.168.0.5:50080      211.239.236.89:443     LAST_ACK
  TCP    192.168.0.5:50081      211.239.236.40:443     LAST_ACK
  TCP    192.168.0.5:50108      13.107.246.254:443     CLOSE_WAIT
  TCP    192.168.0.5:50113      23.219.19.250:80       LAST_ACK
  TCP    192.168.0.5:50151      52.231.108.127:443     ESTABLISHED
  TCP    192.168.0.5:50536      52.231.108.127:443     ESTABLISHED
  TCP    192.168.0.5:50610      172.64.155.209:443     ESTABLISHED
  TCP    192.168.0.5:50617      104.18.32.47:443       ESTABLISHED
  TCP    192.168.0.5:50620      172.64.148.235:443     ESTABLISHED
  TCP    192.168.0.5:50646      52.231.108.127:443     ESTABLISHED
  TCP    192.168.0.5:50671      104.18.31.173:443      ESTABLISHED
  TCP    192.168.0.5:50695      104.208.16.95:443      TIME_WAIT
  TCP    192.168.0.5:50696      52.231.108.127:443     ESTABLISHED
  TCP    192.168.0.5:50697      52.231.108.127:443     ESTABLISHED
  TCP    192.168.0.5:50698      52.231.108.127:443     CLOSE_WAIT
```

상태	의미
ESTABLISHED	연결이 정상적으로 유지 중
FIN_WAIT_1 / FIN_WAIT_2	내가 먼저 연결 끊는 과정
CLOSE_WAIT	상대방이 먼저 FIN 보낸 상태, 내가 아직 닫지 않음
LAST_ACK	내가 FIN 보내고 마지막 ACK 기다리는 상태
TIME_WAIT	최종 연결 종료 후 안전하게 남아있는 상태

연결 종료에서의 4-way handshake 과정

A → B : FIN(연결 종료 요청)
A ← B : ACK(확인)
A ← B : FIN
A → B : ACK
연결 종료 완료

2. I/O 멀티플렉싱

**관심있는 I/O 작업들을 동시에 모니터링하고,
그 중에 완료된 I/O 작업들을 한번에 알려주는 방법!**

다수의 클라이언트 요청을 적은 리소스로 처리해야 함

↓

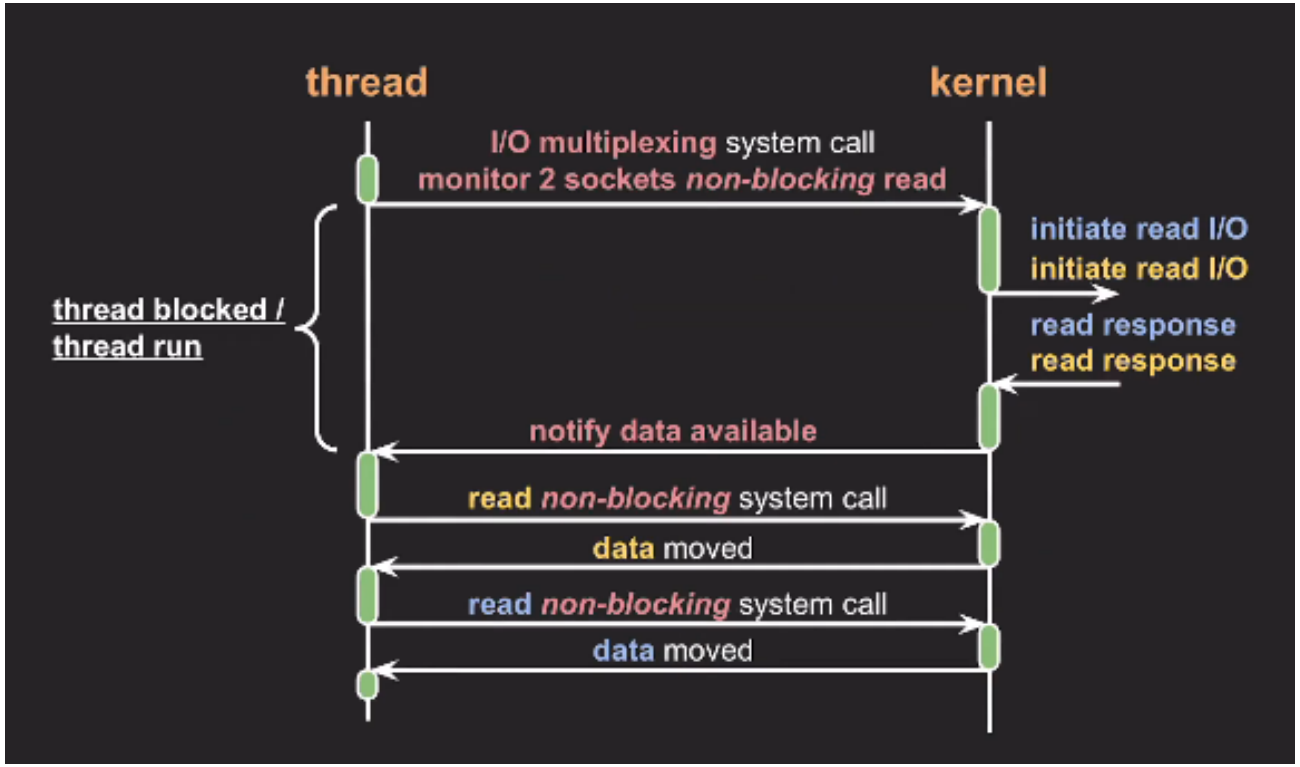
멀티프로세스와 멀티 스레드로 병렬처리 해서 처리 할 수 있지만, 자원 문제와 확장성이 낮다는 단점 존재

↓

I/O 멀티플렉싱
하나의 서버 프로세스가 여러 클라이언트 소켓의 상태를 감시하면서 이벤트가 발생한 소켓만 선택적으로 처리

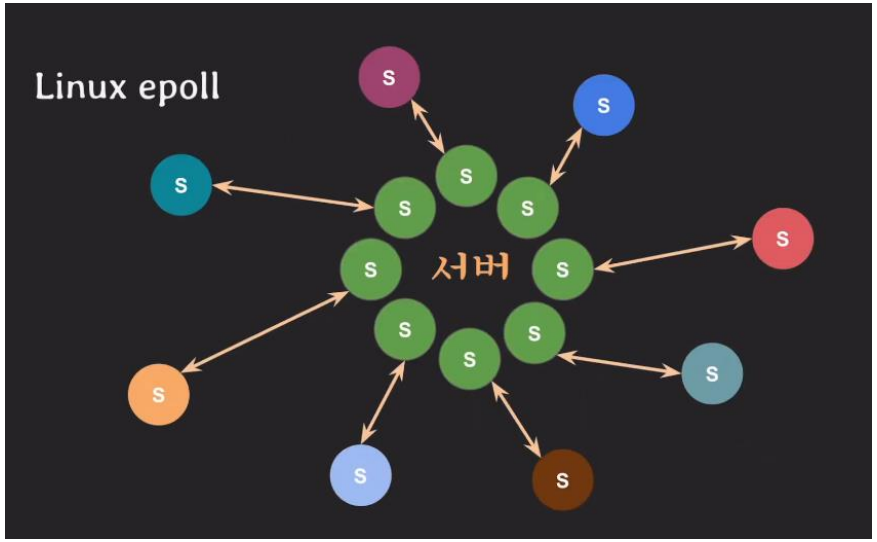
사용되는 대표적 system call
-select-성능 별로
-poll-성능 별로
-epoll (Linux)
-kqueue (mac OS)
-IOCP(윈도우나 솔라리스)

2. I/O 멀티플렉싱



I/O 멀티플렉싱은 한 번의 시스템 콜로 여러 이벤트(read,write 들)에 대해서 여러 소켓들로부터 그 이벤트가 발생하면 알려달라는 **요청을 한번에 처리하는 방식**

2. I/O 멀티플렉싱



서버에 8개의 소켓이 열려있다고 가정

↓

Epoll을 통해 8개의 소켓에 대해서 "하나라도 Read 이벤트가 발생하면 알려줘"라고 등록

↓

여기서 파랑, 빨강, 갈색 클라이언트 소켓에서 요청을 보냈다고 하면, 거기에 맞는 서버 소켓에 들어오게 됨

↓

Epoll은 이 3개의 서버 소켓에 event가 있다는 것을 알려주고, 이 정보를 가지고 3개의 read에 대해서만 처리하게 됨

https://gemini.google.com/share/434a93fa0c4d

PS: 소켓마다 하나의 Thread에 1대1 매핑되니까, 만약 스레드 풀을 만들어 놨다면 요청에 대해 하나씩 처리하게 됨

하지만 원래 알던 연결당 스레드랑은 다르게 요청당 스레드로 요청이 끝나면 다시 반환됨

"전송계층의 멀티플렉싱"은 **데이터를 포트 단위로 묶어 보내는 네트워크 개념**

"I/O 멀티플렉싱"은 **여러 소켓을 동시에 감시하는 운영체제 프로그래밍 기법**

3. HTTP2.0 멀티플렉싱

전송계층의 멀티 플렉싱은 누가 포트번호로 누가 보냈는지 구분하는 거라면, 여기서의 멀티플렉싱은 무엇을 보냈는지 구분하는 것

HTTP/2 멀티플렉싱은 응용계층임

하나의 TCP 연결에 대해서 여러 데이터를 동시에 전송할 수 있는 기법

브라우저가 여러 요청을 생성(A는 이미지, B는 CSS, C는 JS)

↓

각 요청 별로 Stream ID를 부여

↓

각 요청을 작은 프레임 단위로 분할

↓

모든 프레임을 한 TCP 연결로 섞어서 전송

↓

수신측에서는 Stream ID 별로 각 요청을 재조립 해서 처리

퀴즈1

OSI 7계층 중 트랜스포트 계층(Transport Layer)에서, 단일 호스트(IP 주소)가 여러 애플리케이션(웹 브라우저, 게임 등)의 데이터를 동시에 송수신할 수 있도록 해주는 멀티플렉싱/디멀티플렉싱의 핵심 식별자는 무엇인가요?

A. IP 주소 (IP Address)

B. 포트 번호 (Port Number)

C. 시퀀스 번호 (Sequence Number)

D. MAC 주소 (MAC Address)

B. 포트 번호 (Port Number)

✓ **정답입니다!**

포트 번호는 TCP/UDP 헤더에 포함되어, 운영체제가 수신된 패킷을 올바른 애플리케이션(프로세스)에게 전달(demultiplexing)하도록 하는 '문' 역할을 합니다.

퀴즈2

Nginx와 같은 현대적인 웹 서버가 'Connection-per-Thread' 모델 대신, 단일 또는 소수의 스레드로 수천 개의 동시 연결을 효율적으로 처리하는 핵심 기술은 무엇인가요?

A. I/O 멀티플렉싱 (I/O Multiplexing)

B. HTTP 캐싱 (HTTP Caching)

C. 데이터베이스 커넥션 풀링 (DB Connection Pooling)

D. 부하 분산 (Load Balancing)

A. I/O 멀티플렉싱 (I/O Multiplexing)

✓ **정답입니다!**

`select()`, `poll()`, `epoll()` (Linux), `kqueue()` (BSD)와 같은 시스템 콜을 사용하여, 여러 소켓(연결)의 이벤트를 한 번에 감시하고 준비된 소켓만 처리하는 이벤트 기반 방식입니다.

퀴즈3

HTTP/2는 단일 TCP 연결 상에서 '스트림 멀티플렉싱'을 제공하여 *애플리케이션 레벨*의 HOL(Head-of-Line) 블로킹을 해결했습니다. 하지만 HTTP/3 (QUIC)가 등장한 이유는 HTTP/2가 여전히 해결하지 못한 근본적인 문제가 있기 때문입니다. 그 문제는 무엇인가요?

A. *TCP 레벨*의 HOL 블로킹 (TCP Head-of-Line Blocking)

B. TLS 암호화로 인한 CPU 부하 증가

C. 서버 푸시(Server Push) 기능의 비효율성

D. TCP의 3-way-handshake로 인한 연결 수립 지연 시간

A. *TCP 레벨*의 HOL 블로킹 (TCP Head-of-Line Blocking)

✓ **정답**

HTTP/2에서는 모든 스트림이 *하나의* TCP 흐름제어에 묶입니다. 만약 스트림 A의 패킷 하나가 유실(packet loss)되면, TCP는 재전송 및 순서 보장을 위해 스트림 B, C의 멀쩡한 패킷들까지 버퍼에 쌓아두고 애플리케이션으로 전달하지 않습니다. QUIC(UDP 기반)은 스트림별로 독립적인 흐름제어를 하므로 이 문제가 없습니다.

퀴즈4

OSI 7계층에서, 트랜스포트 계층(TCP/UDP)은 포트 번호를 사용하여 '프로세스 대 프로세스' 멀티플렉싱을 수행합니다. 반면, 데이터링크 계층(Ethernet)은 MAC 주소를 사용합니다. 이 두 계층의 멀티플렉싱(주소 지정)에 대한 설명으로 가장 정확한 것은 무엇인가요?

A. 라우터는 패킷 전달 시 포트 번호를 확인하지만, 스위치는 MAC 주소를 확인한다.

B. 포트 번호는 전 세계적으로 고유하지만, MAC 주소는 로컬 네트워크 내에서만 고유하다.

C. TCP는 MAC 주소를, UDP는 포트 번호를 사용하여 멀티

D. MAC 주소는 *동일 네트워크(LAN)* 내의 다음 홉(hop) 장치를 찾는 데 사용되고, 포트 번호는 *최종 목적지 호스트* 내의 특정 애플리케이션을 찾는 데 사용된다.

✓ **정답**

정확한 설명입니다. MAC 주소(L2)는 프레임이 라우터나 목적지 PC 등 '다음' 장치로 가는 데 사용됩니다. 반면 포트 번호(L4)는 IP 주소(L3)와 함께 최종 목적지 컴퓨터에 도착한 패킷이 '어떤 프로그램(소켓)'으로 가야 할지(디멀티플렉싱)를 결정하는 데 사용됩니다.

https://kwj1270.tistory.com/entry/Reactive-%EC%97%AC%EC%A0%95%EA%B8%B01%EB%B6%80-IO-Multiplexing-%EA%B3%BC-Asynchronous-IO-%EA%B7%B8%EB%A6%AC%EA%B3%A0-Event-Loop

https://m.blog.naver.com/whdgml1996/222148507992

https://velog.io/@leo4study/%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EB%A9%80%ED%8B%B0%ED%94%8C%EB%A0%89%EC%8B%B1Multiplexing%EC%9D%B4%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80

https://globalman96.tistory.com/8

https://m.blog.naver.com/no5100/220722643762