네트워크 프로그래밍

1. 네트워크 프로그래밍과 소켓의 이해

2

소켓 프로그래밍의 이해

네트워크 프로그래밍과 소켓

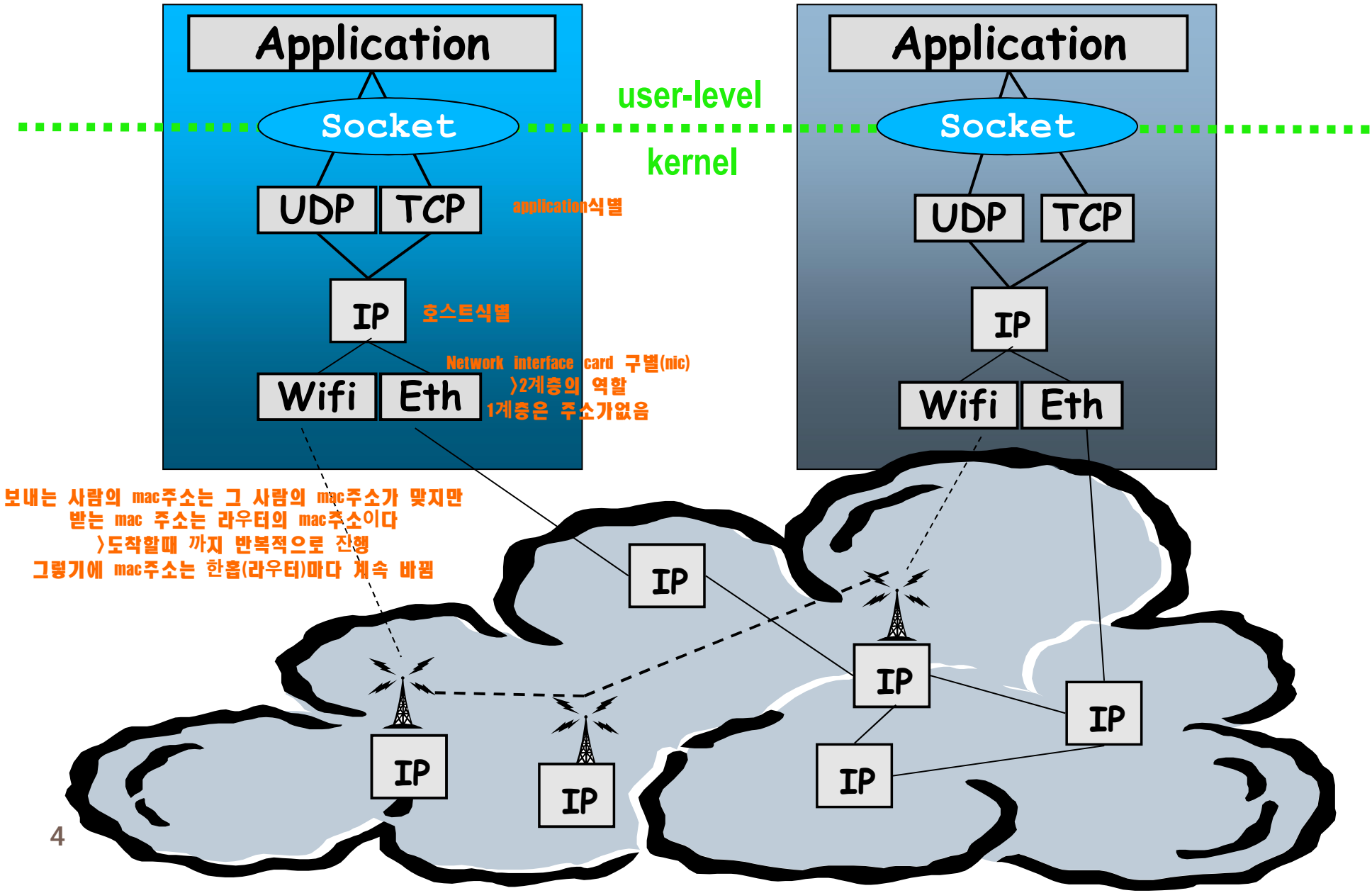
OS: 우리가 디바이스 종속적으로 구현하지 않도록 여러기능을 지원해 줌

- 네트워크 프로그래밍이란?
  - 소켓 API를 이용하여 컴퓨터 네트워크를 사용하는 프로그램을 작성하는 것
    - 컴퓨터 네트워크를 이용한 입출력 프로그래밍
    - 소켓 API를 이용한 입출력 프로그래밍

- 소켓이란 무엇인가?
  - 네트워크 플러그인 인터페이스 (통신 접점)
  - 응용 프로그램이 데이터를 주고 받는 추상화 개념(자료 구조)
  - TCP/IP를 포함하여 다양한 프로토콜 인터페이스 지원

파일 IO에서 네트워크로 바뀌었을 뿐, 즉 본질적으로 파일로 간주.

네트워크 플러그인 인터페이스



응용 프로그램이 데이터를 주고 받는 추상화 개념(자료 구조)

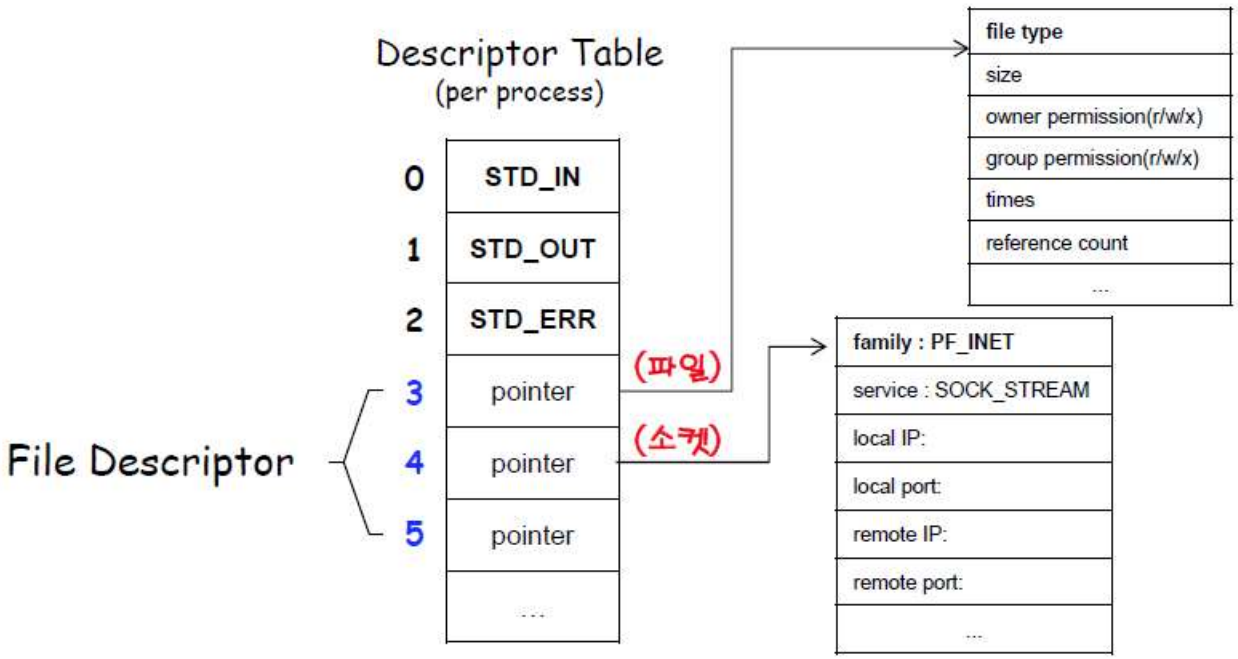
5

- 유닉스/리눅스 계열에서는 **소켓도 파일로 간주**하기 때문에, 저 수준 파일 입출력 함수를 사용하여 소켓 기반의 데이터 송수신이 가능하다.

유닉스는 모든 리소스를 파일로 본다 (디바이스, 파일, 소켓등)
=>so,어떤상황에서든 경우의 수는 파일 입출력으로 단순화됨

파일 또는 소켓 정보 구조체

메타데이터(os 입장에서 파일의 정보를 알아야 관리 할 수 있음)



파일 열기와 소켓의 생성

OS에서 제공하는 함수

파일 열기

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *path, int flag);

    ➜ 성공 시 파일 디스크립터, 실패 시 -1 반환
```

- path　파일 이름을 나타내는 문자열의 주소 값 전달.
- flag　파일의 오픈 모드 정보 전달.

API와 Library의 차이
〉os 레벨에서 제공 = API
〉사용자가 지정 = Library
*만약 library지만 api라고 부는 것은
플랫폼으로서의 역할을 할 수 있을때

소켓의 생성

소켓은 오픈없이 socket()이라는 함수로 만든다.

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);

    ➜ 성공 시 파일 디스크립터, 실패 시 -1 반환
```

TCP/IP를 포함하여 다양한 프로토콜 인터페이스 지원

- 프로토콜 체계(Protocol Family)
  - 프로토콜도 그 종류에 따라서 부류가 나뉘는데, 그 부류를 가리켜 프로토콜 체계라 한다.
  - 프로토콜의 체계 PF_INET은 IPv4 인터넷 프로토콜 체계를 의미한다. 우리는 이를 기반으로 소켓 프로그래밍을 학습한다.

원래는 그냥 메세지를 보냄, law소켓의 경우 transport layer의 헤더까지 직접정의 한 것.

이름	프로토콜 체계(Protocol Family)
PF_INET	IPv4 인터넷 프로토콜 체계
PF_INET6	IPv6 인터넷 프로토콜 체계
PF_LOCAL	로컬 통신을 위한 UNIX 프로토콜 체계
PF_PACKET	Low Level 소켓을 위한 프로토콜 체계
PF_IPX	IPX 노벨 프로토콜 체계

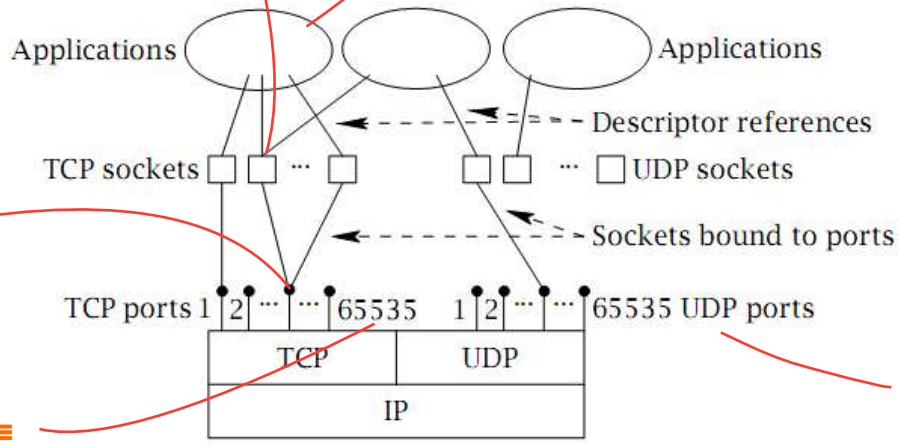
대표적인 프로토콜 체계 정보

소켓함수의 첫번째 인자

소켓의 구분

- 하나의 응용프로그램이 여러 개의 소켓을 가질 수 있으며 반대로 여러 개의 프로그램이 하나의 소켓의 공유가 가능

한 파일을 여러프로세스가 사용가능하지만
문제발생가능
>이때 문제는 프로그래머가 책임져야함



질문: 필요없는 포트를 자를거면 왜 복사를 사용?

mux와 demux때문에 있는것
(demux후 위로 보낸다는 것은 여가에
나오지 않은 추가정보를 이용한다는것)

16비트 = = 2바이트

전송계층 프로토콜마다 포트가 있다

소켓의 구분

- 프로토콜(TCP, UDP혹은 기타 프로토콜), 주소, 포트 별로 구분

```
struct sockaddr_in serv_addr;
. . . .
if(bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error");
. . . .
```

```
struct sockaddr_in
{
    sa_family_t     sin_family;
    uint16_t        sin_port;
    struct in_addr  sin_addr;
    char            sin_zero[8];
};
```

주소체계

PORT번호

32비트 IP주소

사용되지 않음

10

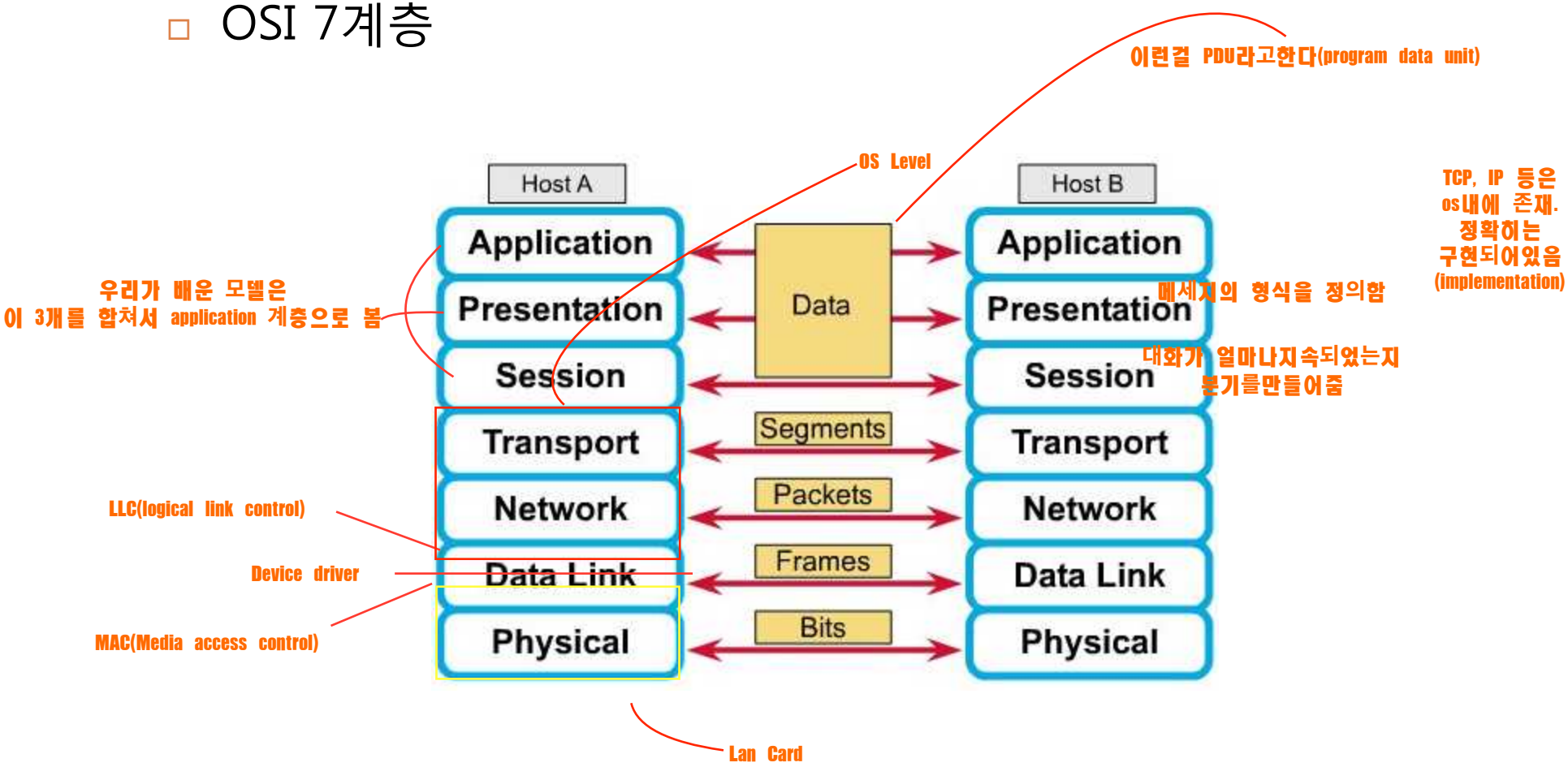
부록

네트워크 기초

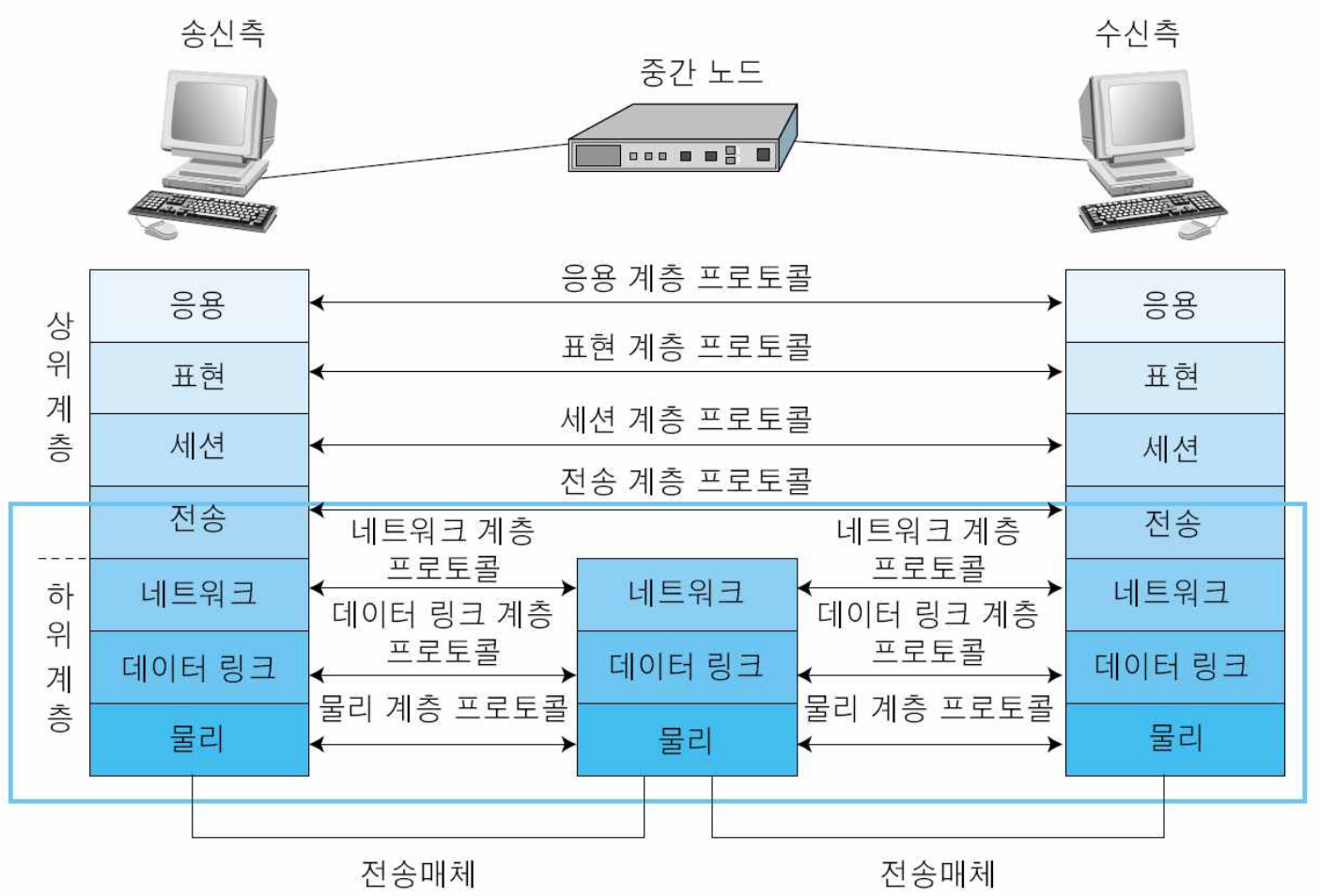
OSI 참조 모델

11

- 국제 표준화 기구(ISO)에서 제정한 네트워크 통신 모델
- OSI 7계층

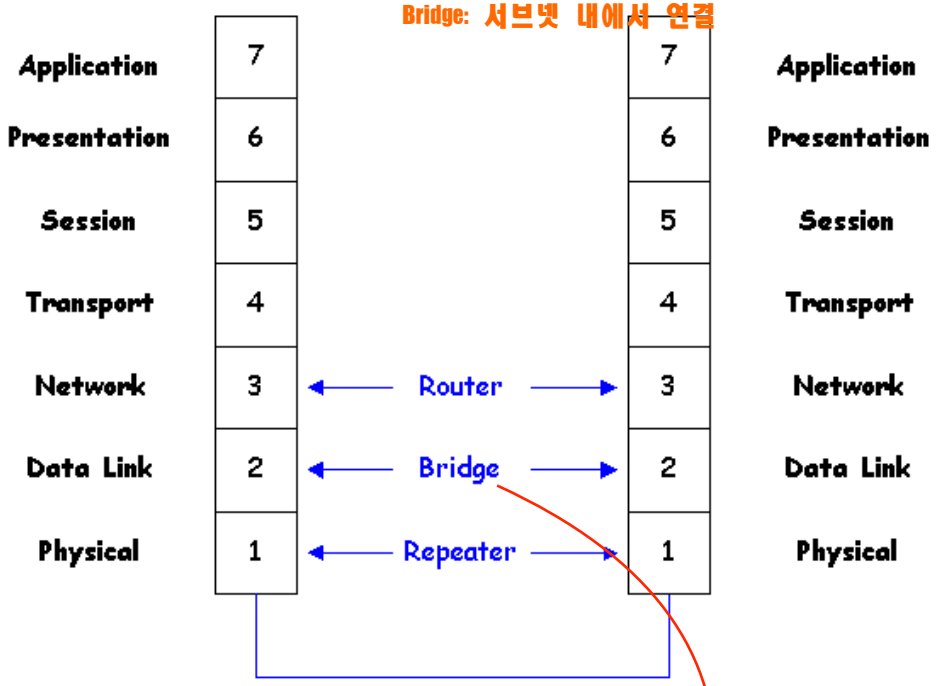


OSI 참조 모델


송신측
중간 노드
수신측
상위 계층
하위 계층
응용
표현
세션
전송
네트워크
데이터 링크
물리
응용 계층 프로토콜
표현 계층 프로토콜
세션 계층 프로토콜
전송 계층 프로토콜
네트워크 계층 프로토콜
데이터 링크 계층 프로토콜
물리 계층 프로토콜
네트워크 계층 프로토콜
데이터 링크 계층 프로토콜
물리 계층 프로토콜
응용
표현
세션
전송
네트워크
데이터 링크
물리
전송매체
전송매체

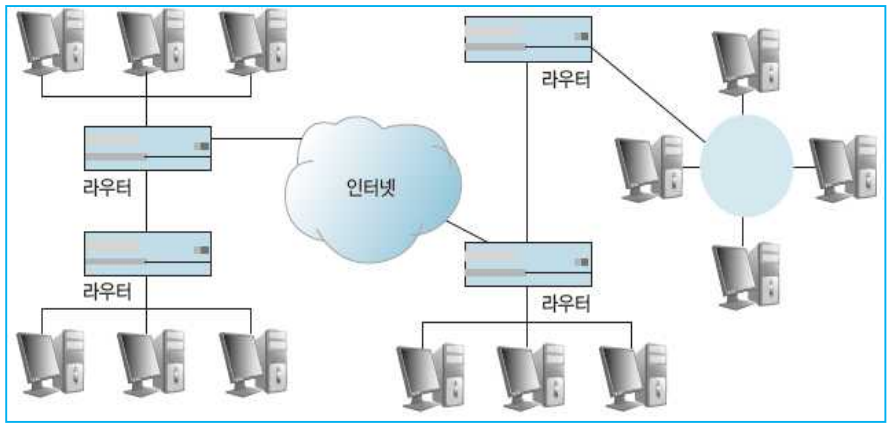
OSI 참조 모델

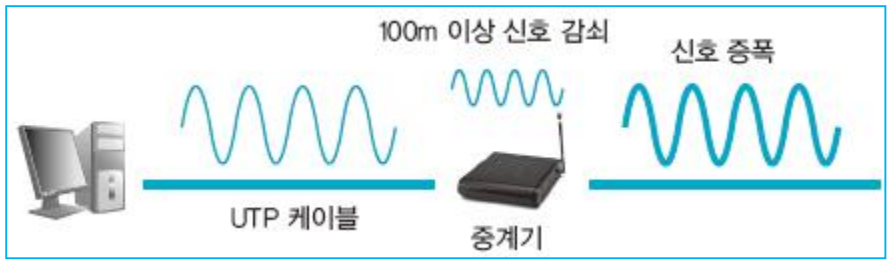
라우터랑 브릿지의 정확한 차이?
〉
Router: 서브넷 끼리 연결
Bridge: 서브넷 내에서 연결



어느계층까지 구현되었느냐에 따라 이름이 바뀐다

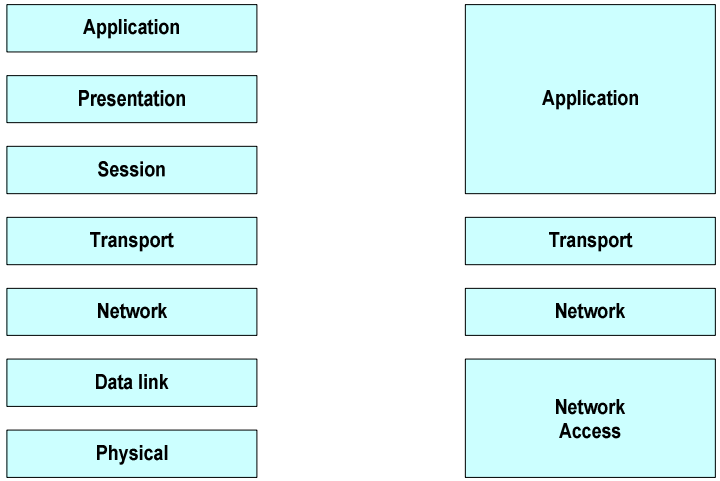
스위칭개념까지 있음
〉신호가 가야하는 방향으로만 보냄



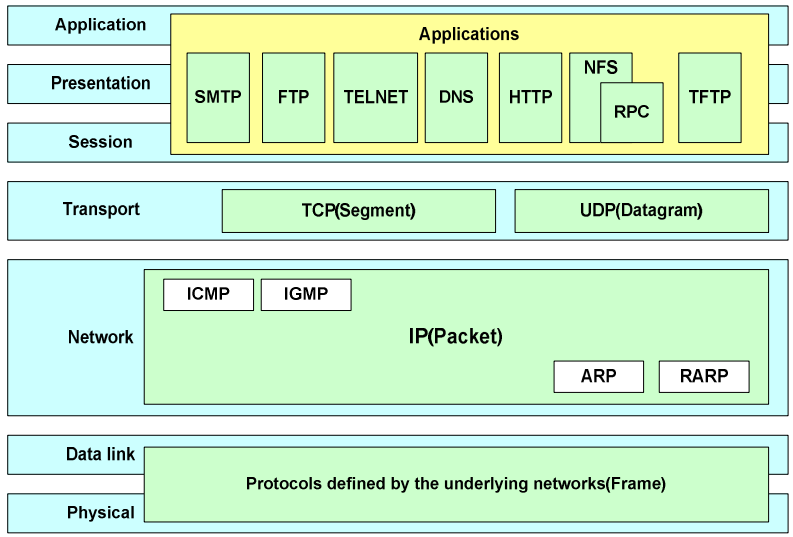


OSI 7 계층과 TCP/IP

OSI 모델과 TCP/IP

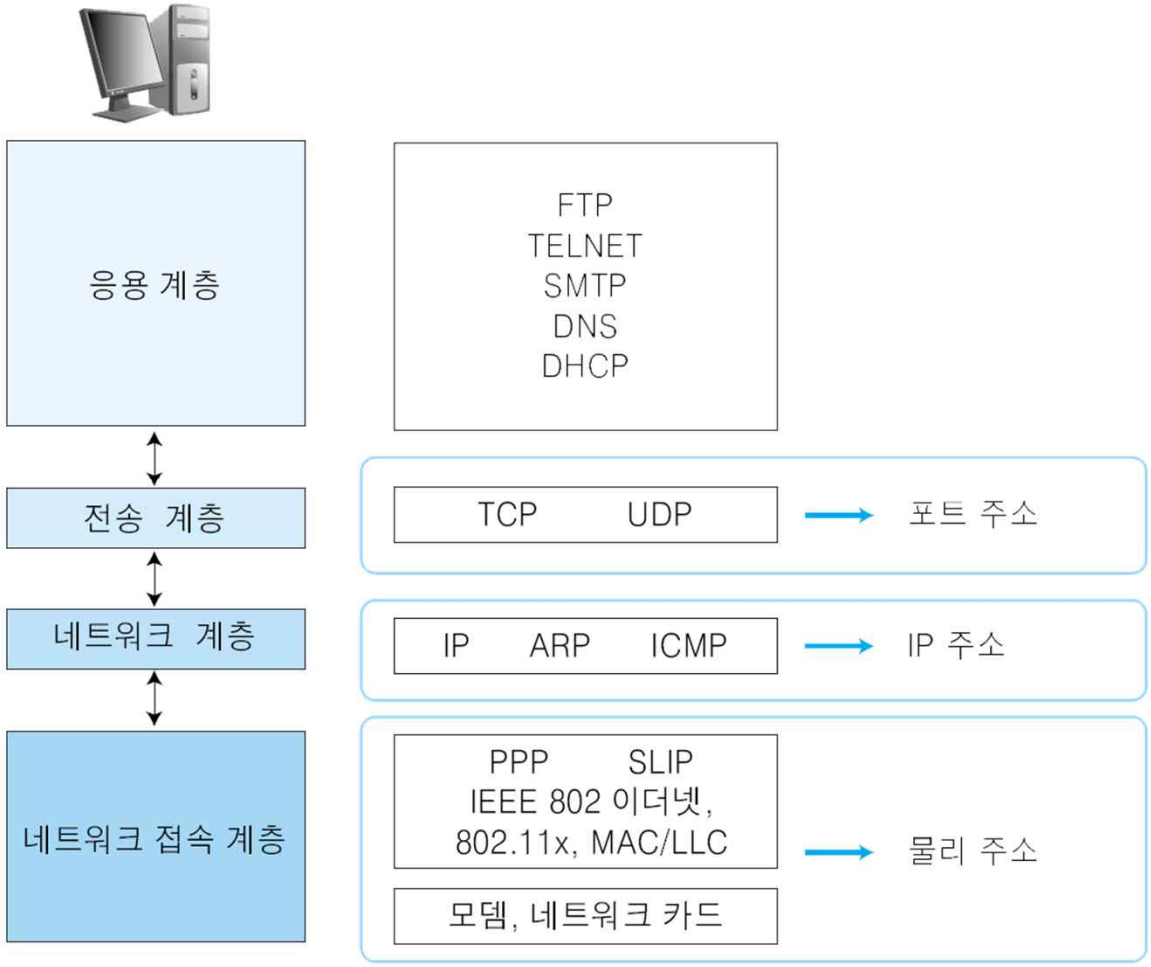


OSI 모델에 적용시켜본 TCP/IP



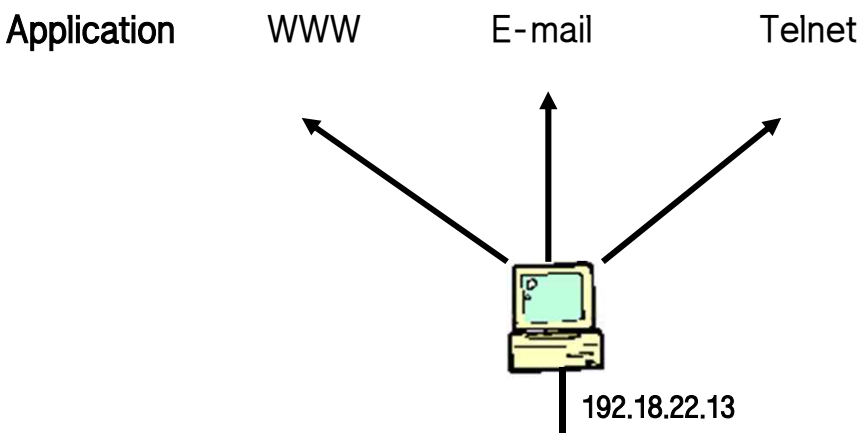
TCP/IP

- 계층과 주소와의 관계

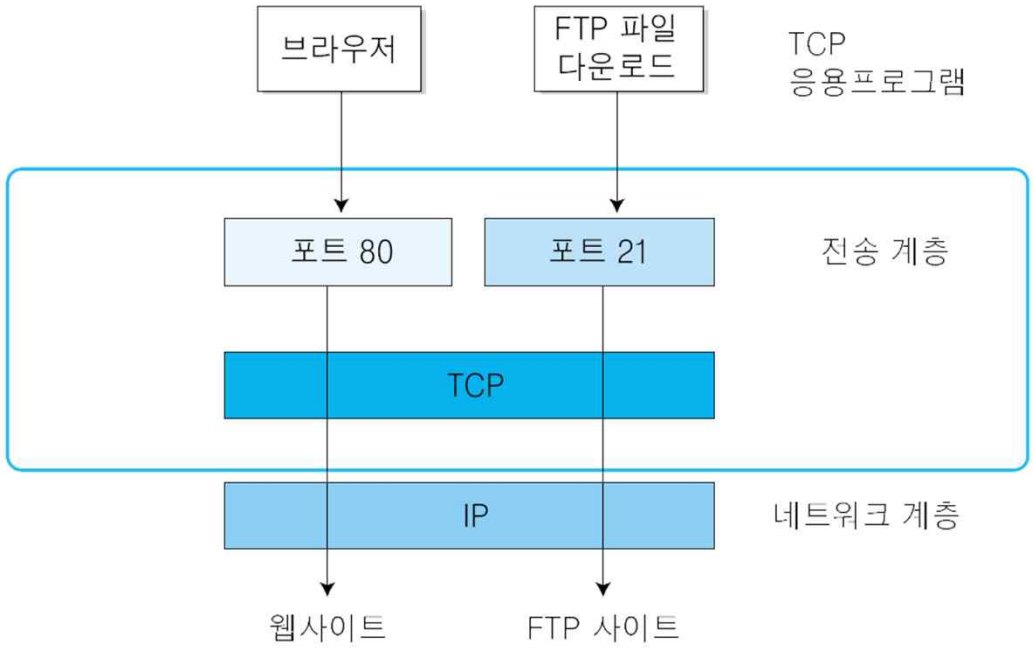


전송 프로토콜의 주소 : Port

- Port는 최종 목적지를 구분
  - IP는 호스트를 구분
  - 호스트는 하나 이상의 응용 프로그램이 탑재 가능 따라서 IP로 응용 프로그램의 구분이 불가능
  - TCP혹은 UDP의 포트는 개별 응용 프로그램을 구분(종단간)
  - IP가 대표 번호라면 Port는 내선번호와 유사
- 결국 인터넷 종단간 응용 프로그램을 구분하기 위해서는
  - IP와 Port의 쌍(Pair)정보가 필요



전송 프로토콜의 주소 : Port



포트의 범주

- 0부터 65535번까지 존재
- 잘 알려진 포트
  - 0~1023번까지의 포트 — well-known port
  - ftp(21번), ssh(22번), telnet(23번), mail(25번), http(80번)…
- 등록된 포트 — 상용회사에서 사용하는 포트
  - 1024~49151번까지의 포트
- 동적인 또는 사적인 포트
  - 49152~66535번까지의 포트
  - 클라이언트가 여는 소켓에 자동적으로 할당되는 포트