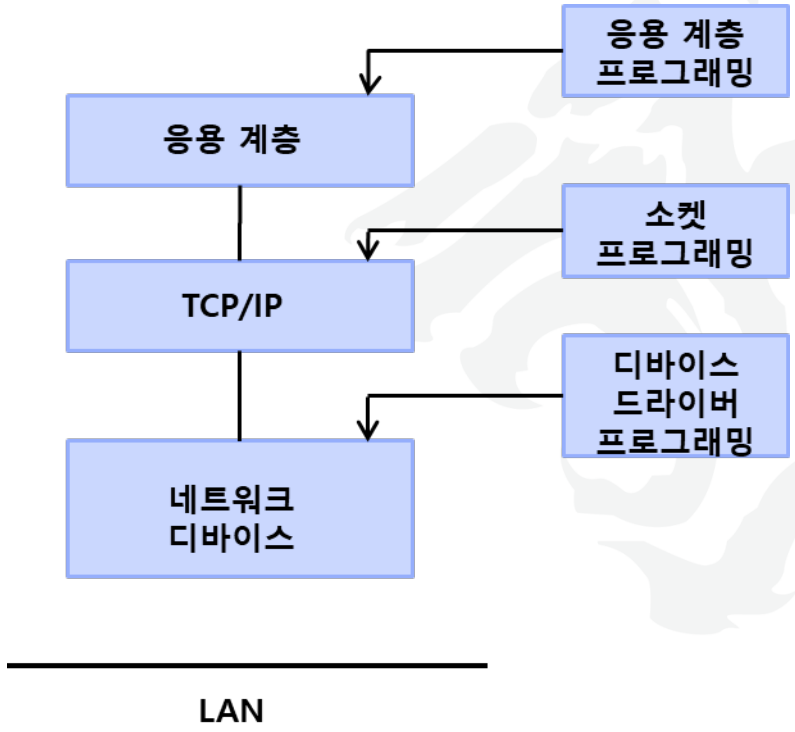
오픈소스소프트웨어

Open-Source Software

ICT융합학부 조용우

소켓

네트워크 프로그래밍 계층별 분류



응용 계층 프로그래밍

- 네트워크 응용 패키지, TCP/IP 응용 프로토콜을 이용
- 데이터의 송수신을 구체적으로 다루지 않음
  - 유닉스의 rsh, rcp
  - RPC
  - http
- 장점
  - 복잡한 네트워크 서비스를 편리하게 구현할 수 있음
- 단점
  - 하위 계층의 구체적인 동작을 직접 제어할 수 없음

소켓(트랜스포트 계층) 프로그래밍

- TCP나 UDP와 같은 트랜스포트 계층의 기능을 직접 이용
- 데이터 그램 단위의 데이터 송수신 처리
- TCP를 이용할 경우 연결설정, 흐름제어, 에러제어가 가능
- 대표적인 API (Application Program Interface)
  - UNIX BSD socket, winsock
- 인터넷 응용 프로그래밍의 기초

디바이스 드라이버 계층 프로그래밍

- OSI 계층 2 이하의 인터페이스를 직접 다루는 프로그래밍
  - 예) Microsoft 사의 NDIS (Network Driver Interface Specification) API
    - Windows에서 이더넷 프레임의 송수신을 직접 처리
- 프레임 단위의 송수신을 구체적으로 제어, 네트워크 상태 모니터링 등
  - LAN 카드 개발 시 테스트 프로그램 작성에 사용
- 프레임을 송수신하는 기능만을 이용
  - 흐름제어, 에러제어, 인터넷주소 관리 등의 기능은 별도로 구현해야 함

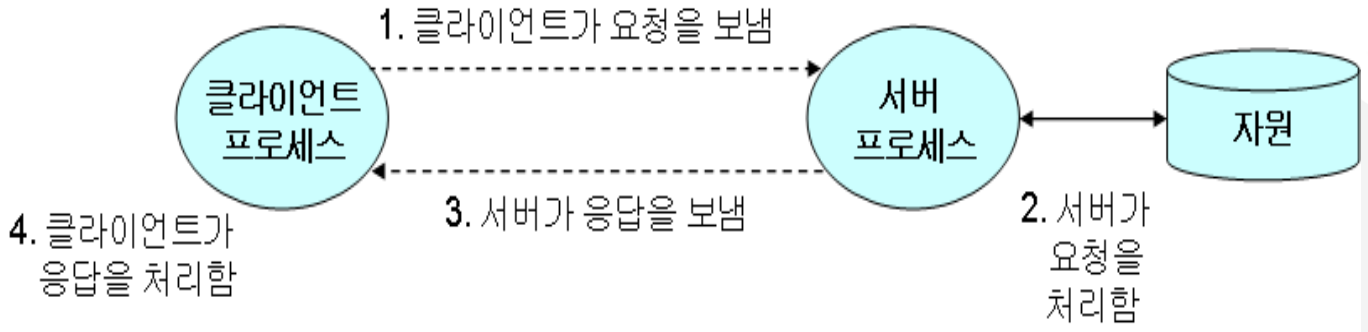
클라이언트-서버 모델

- 대부분의 네트워크 프로그램의 구현 모델
- 서버와 클라이언트
  - 서버 : 서비스를 제공하는 장비
  - 클라이언트 : 서비스를 이용하는 장비
- 서버
  - 일반적으로 클라이언트보다 구현이 복잡
    - 클라이언트의 인증
    - 정보보호
    - 동시 서비스
    - 안정성

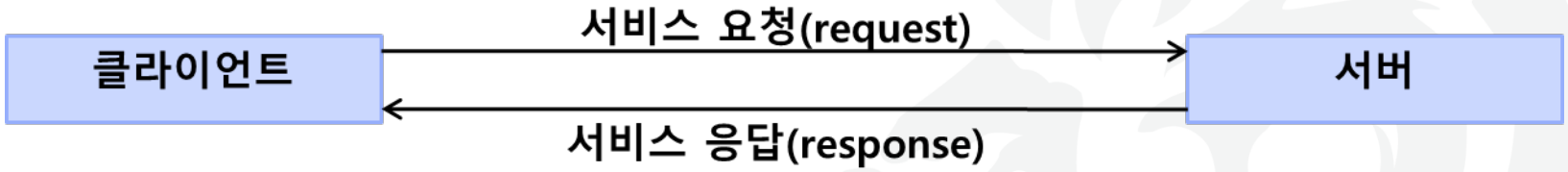
클라이언트-서버 모델

- 네트워크 응용 프로그램
  - 클리이언트-서버 모델을 기반으로 동작한다.

- 클라이언트-서버 모델
  - 하나의 서버 프로세스와 여러 개의 클라이언트로 구성된다.
  - 서버는 자원을 관리하고 클라이언트를 위해 자원 관련 서비스를 제공한다.

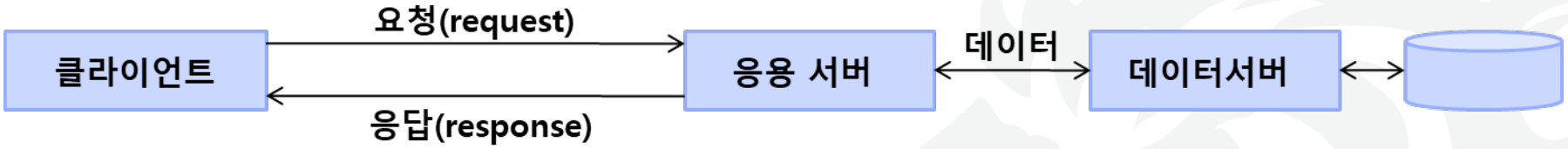


2-tier 클라이언트-서버 모델



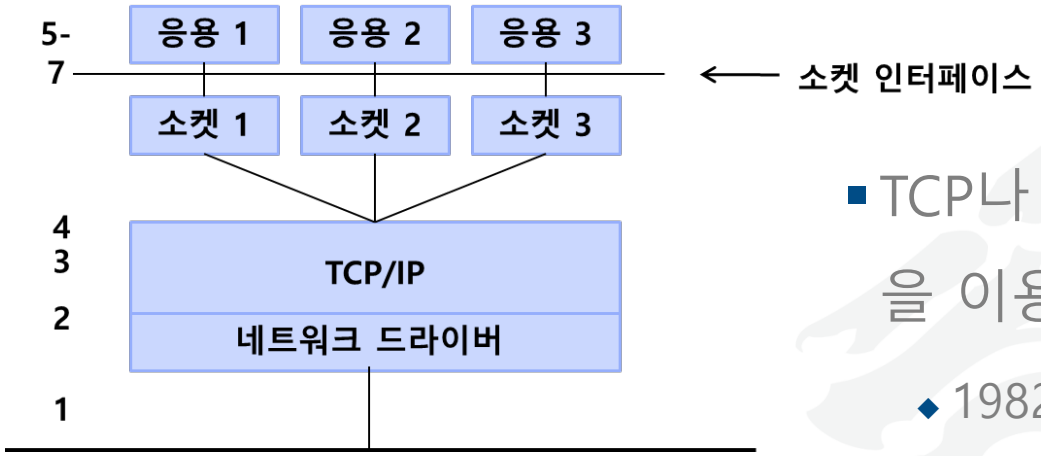
- 대부분의 통신 프로그램
  - 웹(http), ftp, telnet, mail
- 단점
  - 서버에서의 병목 현상
  - 클라이언트의 증가는 서버에 트래픽 집중과 처리 용량 부족 현상 발생

3-tier 클라이언트 서버 모델



- 2-tier 클라이언트-서버 모델의 문제점을 개선한 구조
- '응용서버'와 '데이터서버'로 구분
  - 클라이언트는 응용 서버에 서비스를 요청
  - 응용서버는 데이터 서버로부터 데이터를 얻어 클라이언트에 응답
- 장점
  - 클라이언트는 데이터서버의 정보가 필요없음
  - 클라이언트가 같은 요청을 동시에 하면 응용 서버는 이를 한번만 처리
    - 데이터 서버의 통신 부담 저하

소켓의 정의



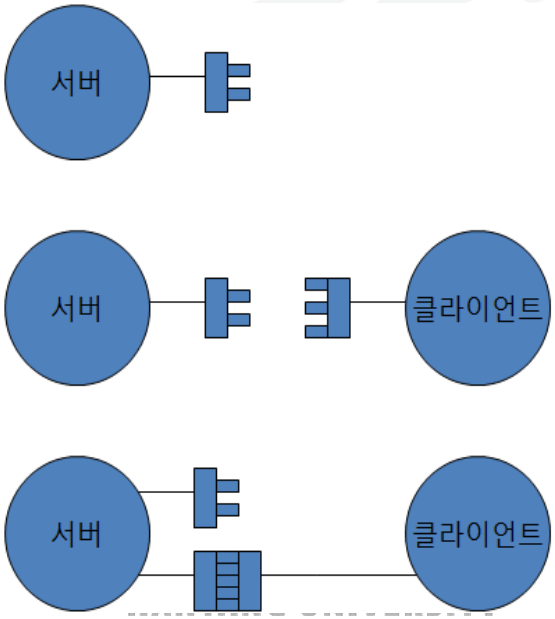
- TCP나 UDP와 같은 트랜스포트 계층을 이용하는 API
  - 1982년 BSD 유닉스 4.1에서 소개
  - 모든 유닉스 운영체제에서 제공
  - Windows는 Winsock으로 제공

소켓의 종류

- 소켓
  - 네트워크에 대한 사용자 수준의 인터페이스를 제공
  - 소켓은 양방향 통신 방법으로 클라이언트-서버 모델을 기반 하고 있어 프로세스 사이의 통신에 매우 적합하다.
- 유닉스 소켓(AF_UNIX)
  - 같은 호스트 내의 프로세스 사이의 통신 방법
- 인터넷 소켓(AF_INET)
  - 인터넷에 연결된 서로 다른 호스트에 있는 프로세스 사이의 통신 방법

소켓 연결

1. 서버가 소켓을 만든다.
2. 클라이언트가 소켓을 만든 후 서버에 연결 요청을 한다.
3. 서버가 클라이언트의 연결 요청을 수락하여 소켓 연결이 이루어진다.



소켓 연결 과정

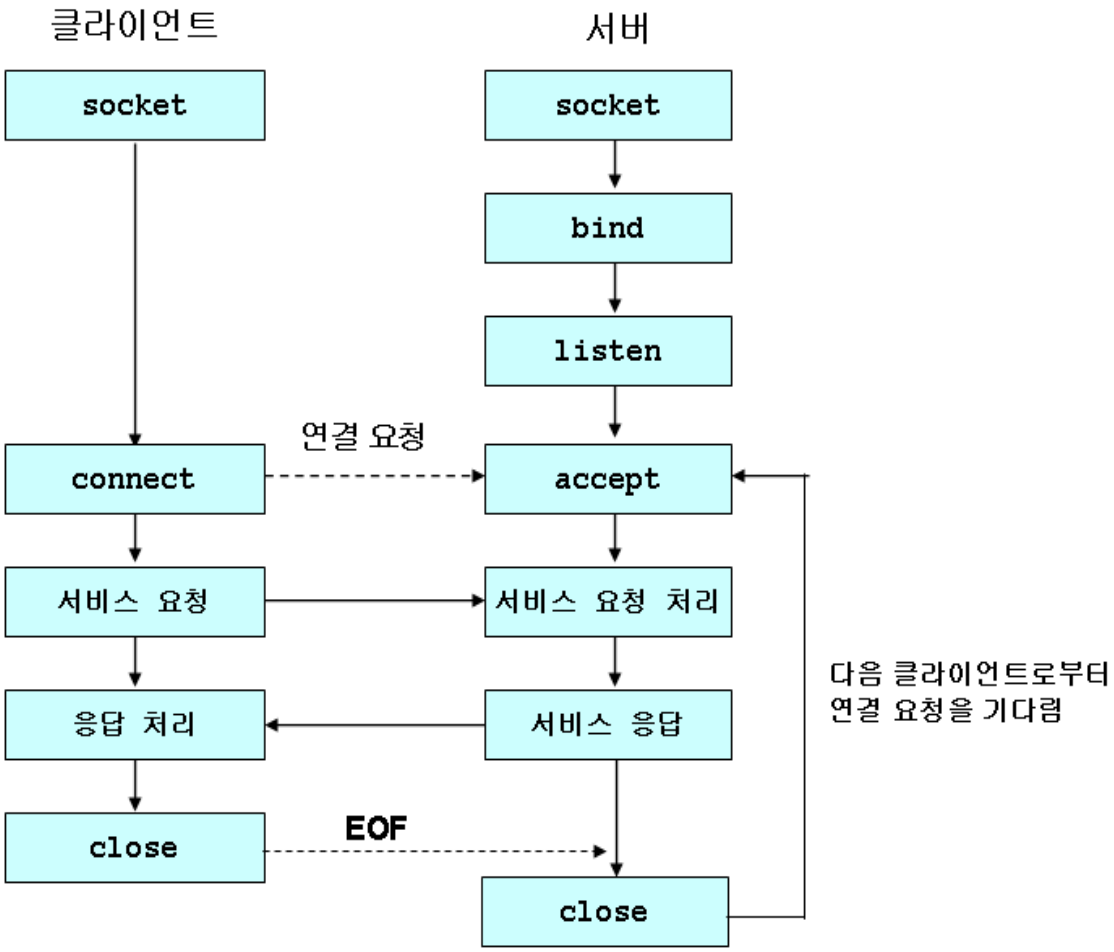
- 서버

1. socket() 호출을 이용하여 소켓을 만들고 이름을 붙인다.
2. listen() 호출을 이용하여 대기 큐를 만든다.
3. 클라이언트로부터 연결 요청을 accept() 호출을 이용하여 수락한다.
4. 소켓 연결이 이루어지면 다음의 행동을 취한다.
   - 자식 프로세스를 생성
   - 클라이언트로부터 요청 처리
   - 클라이언트에게 응답

소켓 연결 과정

- 클라이언트
  1. socket() 호출을 이용하여 소켓을 만든다.
  2. connection() 호출을 이용하여 서버에 연결 요청을 한다.
  3. 서버가 연결 요청을 수락하면 소켓 연결이 만들어진다.
  4. 서버에 서비스를 요청하고 서버로부터 응답을 받아 처리한다.

소켓 연결 과정


클라이언트
서버
socket
socket
bind
listen
연결 요청
connect
accept
서비스 요청
서비스 요청 처리
응답 처리
서비스 응답
close
EOF
close
다음 클라이언트로부터
연결 요청을 기다림

소켓 만들기

int socket(int domain, int type, int protocol)

소켓을 생성하고 소켓을 위한 파일 디스크립터를 리턴

실패하면 -1을 리턴

- 인터넷 소켓

```
fd = socket(AF_INET, SOCK_STREAM, DEFAULT _PROTOCOL);
```

- 유닉스 소켓

```
fd = socket(AF_UNIX, SOCK_STREAM, DEFAULT _PROTOCOL);
```

소켓에 이름(주소) 주기

int bind(int fd, struct sockaddr* address, int addressLen)

소켓에 대한 이름 바인딩이 성공하면 0, 실패하면 -1을 리턴

- 유닉스 소켓 이름

```
struct sockaddr_un {
    unsigned short sun_family;    // AF_UNIX
    char sun_path[108];           // 소켓 이름
}
```

소켓에 이름(주소) 주기

int bind(int fd, struct sockaddr* address, int addressLen)

소켓에 대한 이름 바인딩이 성공하면 0, 실패하면 -1을 리턴

- 인터넷 소켓 이름

```
struct sockaddr_in {
    unsigned short sin_family;      // AF_INET
    unsigned short sin_port;        // 인터넷 소켓의 포트 번호
    struct in_addr sin_addr;        // 32-bit IP 주소
    char sin_zero[8];               // 사용 안 함
}
```

소켓 큐 생성

int listen(int fd, int queueLength)

소켓 fd에 대한 연결 요청을 기다린다. 성공하면 0, 실패하면 -1을 리턴

- listen() 시스템 호출
  - 클라이언트로부터의 연결 요청을 기다린다.
  - 연결 요청 대기 큐의 길이를 정한다.
- listen(serverFd, 5);

소켓이 연결 요청

int connet(int fd, struct sockaddr* address, int addressLen)

성공하면 0, 실패하면 -1을 리턴

- connect() 시스템 호출
  - fd가 나타내는 클라이언트 소켓과 address가 나타내는 서버 소켓과의 연결을 요청한다.
  - 성공하면 fd를 서버 소켓과의 통신에 사용한다.

연결 요청 수락

int accpet(int fd, struct sockaddr* address, int* addressLen)

성공하면 새로 만들어진 복사본 소켓의 파일 디스크립터, 실패하면 -1을 리턴

- 서버가 클라이언트로부터의 연결요청을 수락하는 내부 과정
  1. 서버는 fd가 나타내는 서버 소켓을 경청하고 클라이언트의 연결 요청이 올 때까지 기다린다.
  2. 클라이언트로부터 연결 요청이 오면 원래 서버 소켓과 같은 복사본 소켓을 만들어 복사본 소켓과 클라이언트 소켓을 연결한다.
  3. 연결이 이루어지면 address는 클라이언트 소켓의 주소로 세팅되고 addressLen는 그 크기로 세팅한다.
  4. 새로 만들어진 복사본 소켓의 파일 디스크립터를 리턴한다.

인터넷 소켓

인터넷 상의 호스트

- 인터넷 상의 호스트는 32 비트 IP 주소를 갖는다
  - 예: 211.218.150.150
- IP 주소는 대응하는 도메인 이름을 갖는다.
  - 예: 211.218.150.150 --> ict.hanyang.ac.kr
- 32 비트 IP 주소는 저장

```
/* 인터넷 주소 구조체 */
struct in_addr {
    unsigned int s_addr; // 네트워크 바이트 순서(big-endian)
};
```

DNS(Domain Name System)

- 인터넷은 IP 주소와 도메인 이름 사이의 맵핑을 DNS라 부르는 전세계적인 분산 데이터베이스에 유지한다.

```
/* DNS 호스트 엔트리 구조체 */
struct hostent {
    char *h_name;        // 호스트의 공식 도메인 이름
    char **h_aliases;    // null로 끝나는 도메인 이름의 배열
    int h_addrtype;      // 호스트 주소 타입(AF_INET)
    int h_length;        // 주소의 길이
    char **h_addr_list;  // null로 끝나는 in_addr 구조체의 배열
};
```

DNS 관련 함수

struct hostent *gethostbyaddr(const char* addr, int len, int type);

길이가 len이고 주소 타입 type인 호스트 주소 addr에 해당하는 hostent 구조체를 리턴

struct hostent *gethostbyname(char* name);

도메인 이름에 대응하는 hostent 구조체에 대한 포인터를 리턴

- 호스트의 IP 주소 혹은 도메인 이름을 이용하여 DNS로부터 호스트 엔트리를 검색할 수 있다.

DNS 관련 함수

char* inet_ntoa(struct in_addr address);

IP 주소 address에 대응하는 A. B. C. D 포맷의 스트링을 리턴

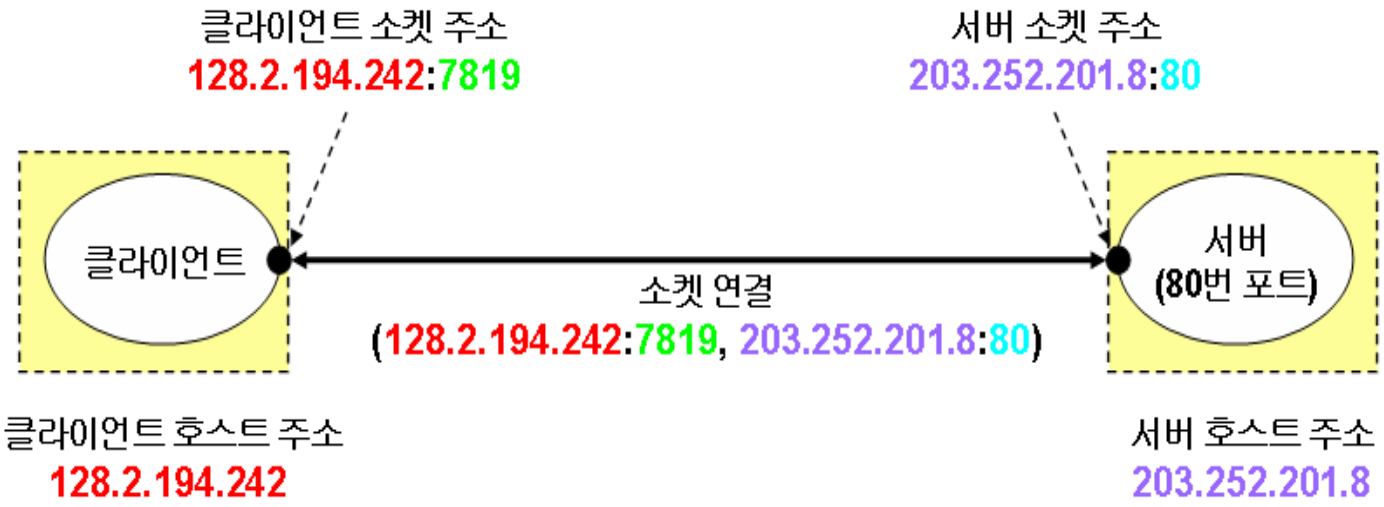
unsigned long inet addr(char* string);

A. B. C. D 포맷의 IP 주소를 네트워크 바이트 순서로 된 이진 데이터로 변환하여 리턴

- in_addr 구조체 형식으로 된 IP 주소를 프린트 할 수 있는 스트링으로 변환한다.

인터넷 소켓

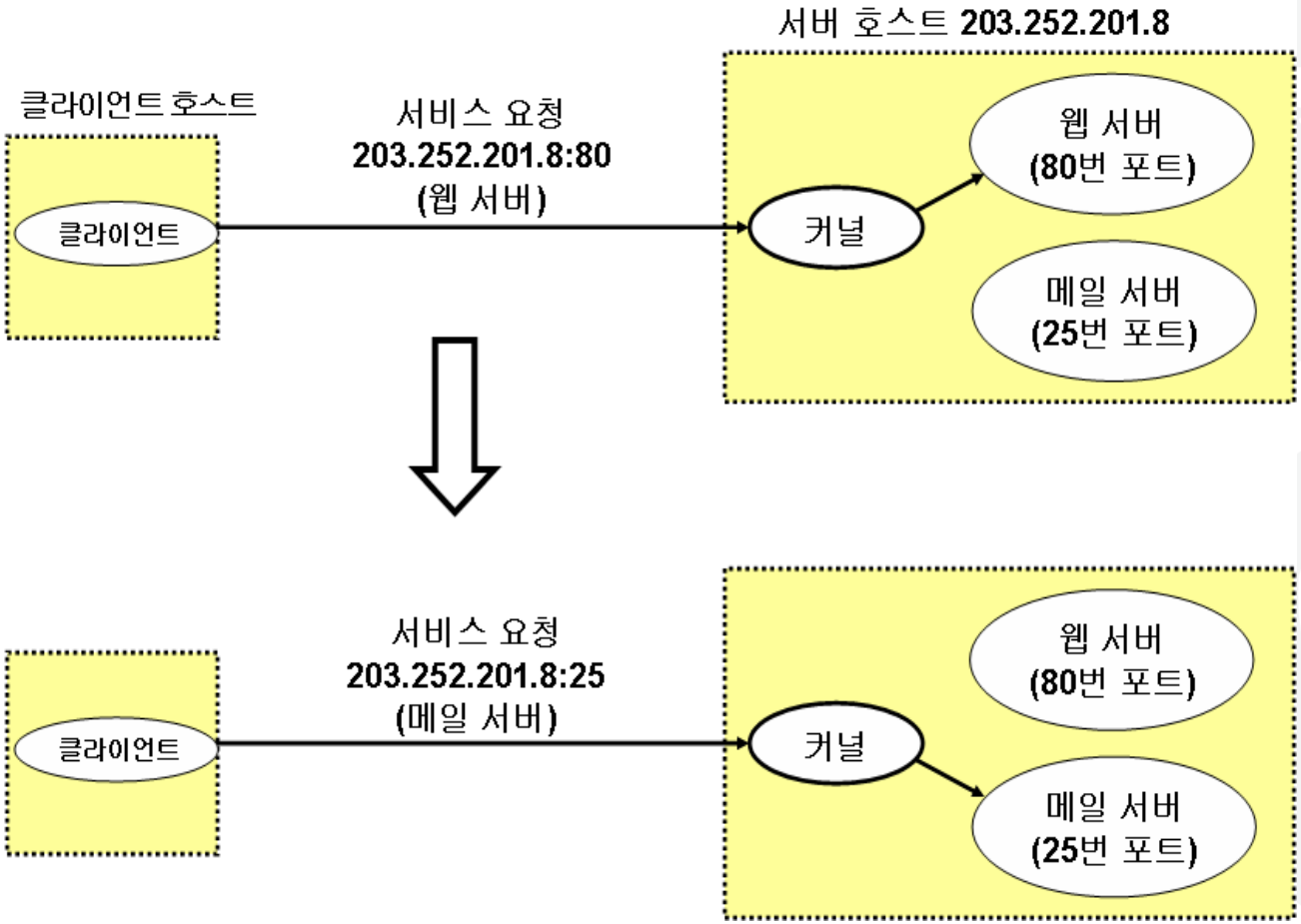
- 인터넷 소켓
  - 서로 다른 호스트에서 실행되는 클라이언트-서버 사이의 통신
  - 양방향(2-way) 통신
  - 소켓을 식별하기 위해 호스트의 IP 주소와 포트 번호를 사용
- 인터넷 소켓 연결 예



인터넷 소켓

- 인터넷 소켓 통신을 사용하는 SW
  - 웹 브라우저
  - ftp
  - telnet
  - ssh
- 잘 알려진 서비스의 포트 번호
  - 시간 서버: 13번 포트
  - ftp 서버: 20,21번 포트
  - 텔넷 서버: 23번 포트
  - 메일 서버: 25번 포트
  - 웹 서버: 80번 포트

클라이언트-서버 인터넷 소켓 연결 과정



인터넷 소켓 이름(주소)

- 인터넷 소켓 이름(주소)

```
struct sockaddr_in {
    unsigned short sin_family;      // AF_INET
    unsigned short sin_port;        // 인터넷 소켓의 포트 번호
    struct in_addr sin_addr;        // 32-bit IP 주소
    char sin_zero[8];               // 사용 안 함
}
```

- 소켓 이름을 위한 포괄적 구조체

```
struct sockaddr {
    unsigned short sa_family;       // 프로토콜 패밀리
    char sa_data[14];               // 주소 데이터
};
```

파일 서버 클라이언트

- 서버
  - 파일 이름을 받아 해당 파일을 찾아 그 내용을 보내주는 서비스
  - 명령줄 인수로 포트 번호를 받아 해당 소켓을 만든다.
  - 이 소켓을 통해 클라이언트로부터 파일 이름을 받아 해당 파일을 열고 그 내용을 이 소켓을 통해 클라이언트에게 보낸다.
- 클라이언트
  - 명령줄 인수로 연결할 서버의 이름과 포트 번호를 받아 해당 서버에 소켓 연결을 한다.
  - 이 연결을 통해 서버에 원하는 파일 이름을 보낸다.
  - 서버로부터 해당 파일 내용을 받아 사용자에게 출력한다.