Network

인터넷이란, 전세계에 거미줄처럼 연결되어 있는 컴퓨터들의 연결망(International Network)

1. 네트워크

1-1. 네트워크의 종류

◘ 근거리 통신망 (LAN, Local Area Network)

- 선의 총 연장이 약 1Km 이상을 벗어나지 않는 네트워크
- 표준 통신 방식은 이더넷(Ethernet), 또는 FDDI(광케이블) 통신
- 100Mbps까지의 통신속도

◘ WAN (Wide Area Network)

- 1Km이상의 거리를 연결해야만 하는 네트워크
- LAN을 서로 연결하여 구성되는 네트워크
  (LAN이 컴퓨터들을 연결하는 네트워크라면 WAN은 LAN과 LAN을 연결하는 네트워크)

◘ MAN (Metropolitan Area Network)

- 대도시를 포괄하는 네트워크
- 통신망의 총 거리는 수백 Km
- Gbps(Giga bps)까지의 통신속도를 지원
- 지하에 광케이블을 매설하여 구성

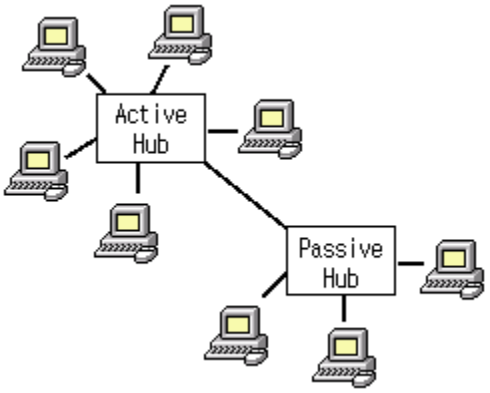
◘ VAN (Value Added Network)

- 부가가치 통신망
- 공중 통신망을 통해 일반에게 정보를 서비스하는 네트워크
- 부가가치를 창출하는 네트워크로서 LAN, WAN, MAN과 같이 물리적인 구분이 아닌 서비스 차원의 네트워크
- 유니텔, 하이텔과 같은 PC통신, 또는 한국통신의 HinetP와 같은 통신망에서의 정보 서비스

1-2. 근거리 통신망(LAN) 의 구성

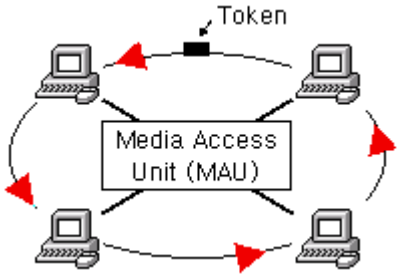
○ 스타(Star)방식의 구성

- 제어 노드가 중앙에 있으며 사용자의 노드는 모두 중앙 제어 노드에 연결
- 중앙 제어 노드에 문제가 발생하면 네트워크 전체가 통신 불능 상태
- 전송 지연 시간이 길다.



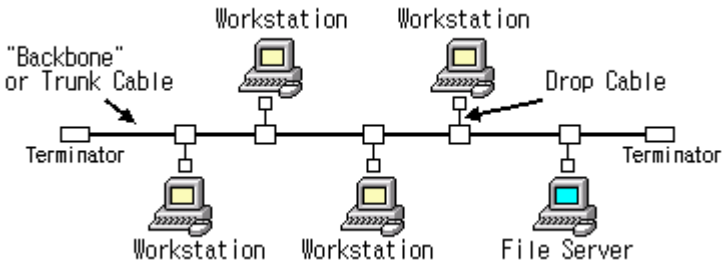
○ 링(Ring)방식의 구성

- 링 상에 존재하는 노드들간의 연결을 최소화
- 링 상에 존재하는 노드간의 통신은 인접 노드로 수신지를 찾을 때까지 데이터를 전송
- 수신이 이루어지면 링 상에서 데이터를 제거
- 전송 지연 시간이 길다.
- 네트워크상의 어떤 노드라도 문제가 발생하면 네트워크 전체가 통신 불능상태



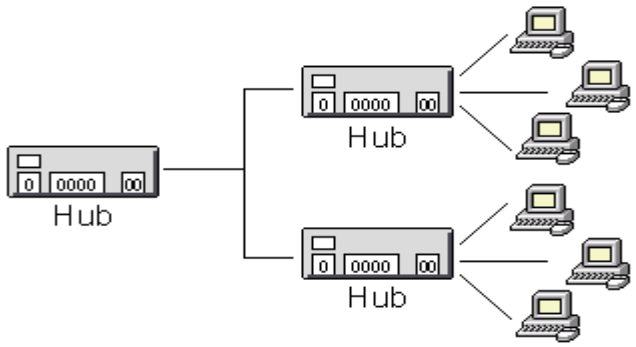
○ 버스(Bus)방식의 구성

- 하나의 통신 회선을 모든 노드가 공유하는 방식
- 종단에서 종단 사이에 노드들이 배치
- 브로드캐스팅 방식으로 메시지를 모든 노드로 전달
- 각 메시지에는 수신지의 주소가 포함되어 있으므로 수신 노드만 데이터를 전달 받는다
- 브로드캐스팅 방식이므로 특정 노드의 상태에 따라서 네트워크의 상태가 변하지 않는다
- 속도가 비교적 빠르다.



○ 허브/트리(Hub/Tree)방식의 구성

● 버스 방식이 변화한 것으로 허브(Hub)는 다수의 버스 방식을 트리처럼 연결하는 방식



1-3. 무선 네트워크

● LAN , WAN 은 전적으로 선(케이블)이 필요한 유선 통신이다. 이러한 유선 통신망은 구축하는 비용은 적게 들지만, 일단 구축한 후에 추가 확장이나 위치 이동이 어려운 단점이 있다.

● 무선을 사용한 네트워크는 네트워크의 추가 확장이나 위치 이동을 아주 간단하게 할 수 있다.

● 여러 개의 층으로 이루어진 건물 전체를 네트워크망으로 연결할 때, 유선 통신을 이용한다면 각 층의 연결이 상당이 어렵겠지만, 무선 네트워크를 이용한다면 문제점 해결 가능.

● 무선 네트워크의 단점은 설치비용이 비싸며, 네트워크 사이의 거리가 멀거나 전파 음영 지역에서는 통신 이용이 불가능

● 무선 네트워크 방식

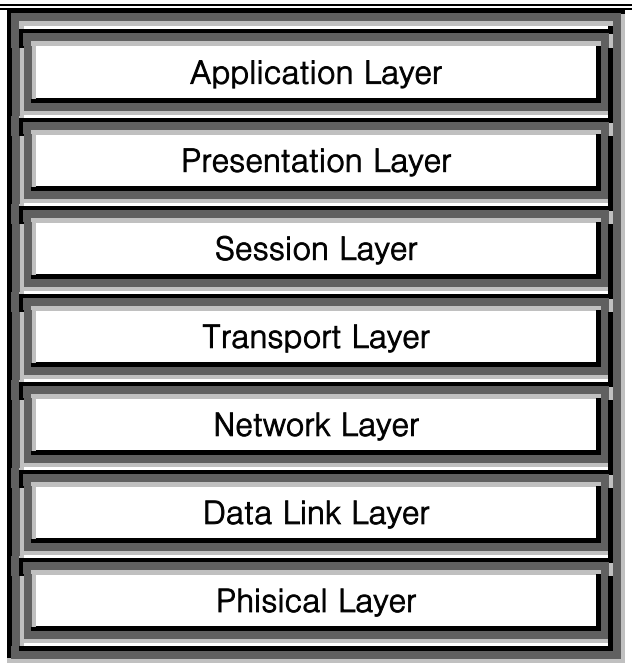
❶ 적외선(IrDA) : 간단하게는 TV 리모콘에서부터 복잡하게는 노트북의 기본적인 통신에게까지 채용되고 있는 방식이다. 여러대의 복잡한 네트워크는 구축할 수 없으며, 주로 1:1 의 간단한 데이터 왕복에 사용

❷ 레이져(Laser) : 레이져를 통신 수단에 사용
거리는 멀지만, 직선상에 장애물이 없을 경우 사용 가능

❸ 전파: 가장 널리 쓰이는 방식
간단하게 노트북에 핸드폰을 연결하여 무선 모뎀으로 사용하는 것도 무선 네트워크의 한 종류라고 볼 수 있다. 그러나 좀더 적극적인 의미로 생각해 보면 무선 랜카드를 사용한 네트워크의 구축을 말한다.

● 무선 랜카드는 CDMA(Code Division Multiple Access) 방식을 이용하는데, 기존의 유선 네트워크망과의 혼합이나 무선망의 네트워크 구성에 사용된다

2. OSI 7 Layer (Open Systems Interconnection Layer)



2-1. OSI(open systems interconnection)이란?

● 컴퓨터가 통신하기 위한 7가지 단계를 표현
● 폐쇄형 시스템이 아닌 개방형 시스템 간의 연결을 의미.
● 개방형이란 같은 종류의 시스템끼리만 통신이 가능하던 종래와는 달리 시스템의 종류나 구현의 방법, 시스템의 규모 등의 조건에 제약받지 않고, 서로 다른 시스템끼리도 연결하여 통신이 가능하도록 한 것
● 모델의 정립에 있어서 어떤 특정 시스템이나 제품을 모델로 삼은 것이 아니고, 시스템 간의 통신에 있어서 요구 또는 고려되어지는 사항들을 정리, 추상화시켜 모델로 정립
● osi는 시스템 간의 정보 교환만을 다루는 것은 아니며, 시스템끼리 어떤 공동의 일을 하기 위한 상호협력관계 등에도 관계되어 있다. (예를 들어 프로세스 간의 통신, 데이타의 표현, 데이타의 저장, 프로세스 및 자원 관리 등).
● open systems interconnection, 개방형 시스템간 상호 접속 이라고 불리는 7계층의 데이타 통신 모델을 제정하였으며, 각 계층은 하나 이상의 고유한 통신 기능을 수행하며 고유한 규정을 갖고 있다.
● 모델의 각 계층은 바로 아래와 위의 계층에 대해 각각 클라이언트와 서버의 역할을 한다.
● 특정 제조업체에 의존하지 않는 중립적인 기관이 결정한 표준적인 통신의 규칙.

2-2. 계층별 요약

7계층 - Application(응용) Layer

-. 웹브라우져나 FTP 클라이언트 처럼 파일 전송, DB, 원격접속, 전자메일등을 관리.
-. 통신상대, 서비스 품질, 사용자 인증과 비밀을 고려하고, 데이터 구문의 제약을 정한다 (응용 프로그램이 응용 계층의 기능을 수행하지만 응용 프로그램 자체는 아니다).
-. 물리 매체를 통해 실제적인 데이타를 비트 스트림으로 전송한다

6계층 - Presentation(표현) Layer

-. 눈에 보이는 아스키 코드같은것들은 기본코드로 변환해주는 부분.
-. 운영체계의 한 부분으로 입력 또는 출력되는 데이터를 하나의 표현 형태에서 다른 표현 형태로 변환하는 것이다 (예를 들면 텍스트로 도착한 데이터를 팝업 윈도우 형태로 변환).
-. 표현 계층을 문법 계층이라고 하기도 한다.

5계층 - Session (세션)Layer

-. 연결설정, 유지, 종료등을 관리 하고, 컴퓨터의 resource를 점유할 수있는 실제적인 layer 이며, 데이터 동기화, 네트워크 오류, 이벤트 검사 등을 한다
-. 종단 호스트 프로그램 사이에서 메시지를 주고받기 위한 설정을 하고, 데이터를 받는 동기를 제어하는 역할을 한다.
-. 통신 세션을 구성하는 역할을 한다.

4계층 - Transport(전송) Layer

-. TCP, UDP, SPX 등 프로토콜과 관련된 층으로 데이터 전송, 에러복구 및 흐름제어, 네트워크 어드레싱 등 작업을 처리.
-. 종단간 제어와 에러를 관리한다. 즉, 신뢰성 있는 데이터 전송을 보장한다.

3계층 - Network(네트워크) Layer

-. 데이터의 전송 경로를 설정 시스템을 연결하는데 필요한 데이터 전송과 기능을 제공한다.
-. IP, IPX등 각종 전송 프로토콜이 활동하는 지역. 시스템 접속 장비관리나 패킷 관리 한다.
-. 데이터 경로를 제어한다 (패킷이 정확한 수신자에게 보내지도록 올바른 경로는 제어하여 수신 쪽에서 받을 수 있게 한다).
-. 경로를 설정하고 다른 쪽으로 전송한다.

2계층 - Data Link (데이터 링크) Layer

-. physical layer에 전송할 신호를 생성해 준다.
-. 계층물리적 계층에서 넘어오는 데이터의 에러를 검사하고 복구 기능도 담당한다.
-. 물리적인 연결을 통하여 두 장치간의 신뢰성 있는 정보 전송 시스템간의 전송 속도 차에 의한 오류나 흐름제어를 담당한다
-. 물리적 레벨의 에러 제어와 동기를 제공하고, 5를 초과하는 1의 스트링으로 비트화한다.
-. 전송 확인과 관리를 담당한다.

1계층 – Phisical(물리) 계층

-. 기계적인 측면(케이블 및 접속장치)과 전기적인 측면(전압, 신호를 변조하는 기술)을 포함한 컴퓨터와 네트워크 사이의 물리적인 연결을 정의 한다.

-. 하드웨어 장비 및 케이블이 여기에 속하며 단지 0과 1의 비트를 전송한다. 비트를 전송하는 책임, 멀티플렉싱, 물리적 토폴로지 관여, 비트 동기화

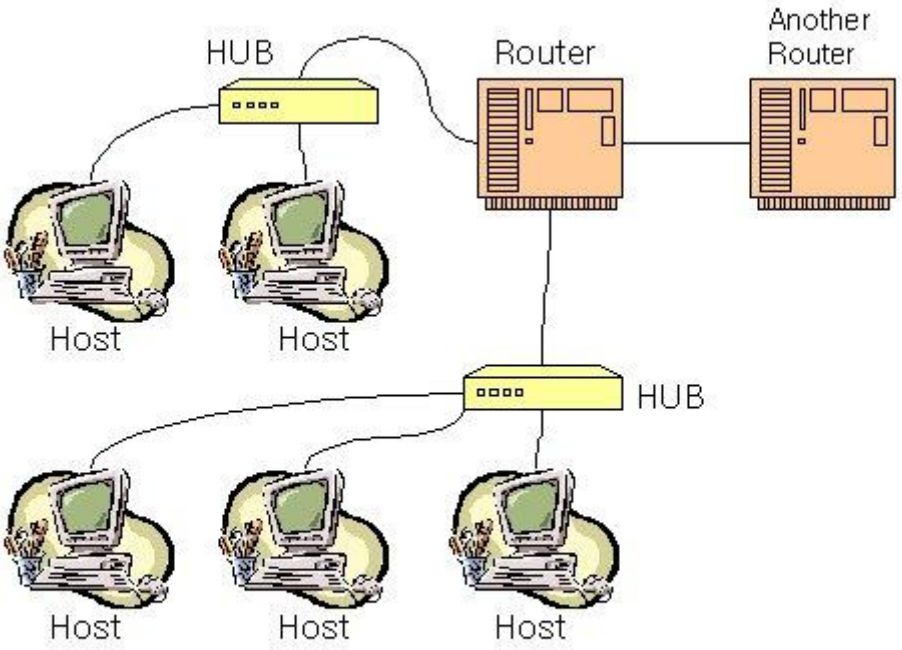
-. 전기 기계적으로 체계를 갖춘 네트웍을 통하여 비트열을 나른다.

-. 전송 매체를 통해 데이터를 주고받는 하드웨어 수단을 제공한다

참고

2대 이상이 데이터를 주고 받게 된다면 아래와 같은 그림을 그릴 수 있다.

네트워크



네트웍의 물리적 연결

Host 는 컴퓨터, Hub 는 공유기 또는 분배기의 역할. 그리고 Router 는 인터넷을 구성하는 Local(근거리통신망-LAN) 망을 연결하는 장치이다.

참고 - 흐름 체크

❶ 내 컴퓨터에서 웹브라우져를 띄웠다.

❷ OSI 7 Layer 에서 7 Layer 인 응용계층에 해당

❸ Yahoo 에 접속을 하려고 www.yahoo.co.kr 를 입력 창에 다가 입력하고 Enter를 쳤다

❹ 6 Layer 인 표현 계층에서 적당한 코드로 변경

❺ 5 Layer 인 세션계층에서 연결 설정을 한다음

❻ 4 Layer 인 전달계층에서 데이터를 전송할 것이고(실제는 데이터를 특정단위로 만들어 3 Layer로 전달)

❼ 3 Layer 에서 2 Layer 인 네트웍카드(NIC: Network Interface Card) 로 전달하고, Network Card는 케이블을 통해 Hub 에게 전달.

❽ 다시 Router로 전달되면, Router는 www.yahoo.co.kr 이 속한 Network으로 데이터를 전송.

❾ 상대측 OSI 7 Layer 에서 반대작용을 해서 www.yahoo.co.kr 서버로 전달되고, 처리 된값이 제가 보고 있는 웹브라우져 까지 동일한 동작으로 보내지게 된다

2-3. OSI 7 Layer 와 TCP/IP Layer 와 그 계층에 해당하는 주요 프로토콜

● TCP/IP Layer 라는 것은 실제 OSI 7 Layer 보다 먼저(OSI 7 Layer: 1984, TCP/IP Layer: 1969) 미국무성의 지원을 받아 시작한 연구프로젝트에서 나와 발전된 것으로 인터넷의 근간이 되는 ARPANET(Advanced Reserch Projects Agency)이라 불리는 초기의 컴퓨터 네트워크에서 발전된 것이다.

● 웹브라우져의 입력창에 HTTP 라는 글자도 프로토콜이다.
HTTP(Hyper Text Transfer Protocol) 는 웹서버라고 불리는 서버어플리케이션이 수행되는 컴퓨터와 웹브라우져가 데이터를 주고 받는 프로토콜이다.
그 내부적으로 TCP(Transmission Control Protocol) 을 사용하는 응용계층 프로토콜이다.

● "Ping www.yahoo.co.kr"이라고 실행창에 치면, 무언가 주고 받았으며, 걸린 시간은 얼마다 라고 화면에 보인다. 보통 네트웍의 상태를 체크하기 위해 쓰는데, 내부적으로 ICMP(Internet Control Protocol)이라고 하는 인터넷 계층 프로토콜을 사용한다.

● FTP 는 File 을 주고 받을 때 쓰는 프로토콜이고 내부적으로 보통 TCP 를 사용한다.

● 계층마다 서로다른 프로토콜이 있고, 그 상위 계층은 그 프로토콜을 이용하기도 한다
HTTP 는 TCP를, FTP 도 마찬가지, Ping 은 ICMP 를 사용한다.
응용계층에는 이것말고도 엄청나게 많은 프로토콜이 있으나 보통은 TCP 나 UDP를 기본으로 하여 많이 사용한다

3. 프로토콜 (Protocol)

3-1. 프로토콜(Protocol)이란?

● 컴퓨터 통신에서 데이터의 전송에 필요한 통신규약.
● 네트워크 장비를 통해 각 PC들이 데이터를 주고받을 수 있도록 하는 일종의 소프트웨어
● 전송규약, 규칙 등으로 해석되는 것은 프로토콜이 PC를 포함, 네트워크를 구성하는 장비들이 데이터를 송수신하는 데 꼭 지켜야 할 원칙(표준화)
● LAN에서 사용되는 프로토콜은 비독점적인 프로토콜과 독점적인 프로토콜로 나뉜다.
비독점적이란 수식어가 붙은 것은 「업계 공통으로 사용한다」는 의미이며, 독점적이란 것은「특정업체의 제품으로 구성된 네트워크를 지원한다」는 뜻을 담고 있다.
비독점적인 프로토콜의 대표적인 예는 TCP/IP(Transmission Control Protocol/Internet Protocol)가 있다

3-2. TCP/IP (Transmission Control Protocol/Internet Protocol)

● 컴퓨터 통신에 사용되는 프로토콜에는 여러 가지가 있지만 인터넷에서 사용하는 프로토콜은 TCP/IP이다.
● 현재 서로 다른 기종과 운영체제를 사용하는 네트워크 또는 컴퓨터들이 서로 데이터를 송수신하는 데에 사용할 수 있는 유일한 표준 프로토콜이다.

TCP (Transmission Control Protocol)	전송할 데이터를 패킷으로 나누고 수신시 결합할 정보를 패킷에 저장한다.
IP (Internet Protocol)	패킷으로 나뉘어진 데이터를 실제로 목적지로 전송한다

3-3. 대표적인 TCP/IP 프로토콜의 응용 프로토콜

FTP(File Tranfer Protocol)	파일 전송을 위한 프로토콜
HTTP(HyperText Transmission Protocol)	하이퍼텍스트 전송 프로토콜, 월드와이드웹에서 사용된다.
SMTP(Simple Mail Transfer Protocol)	인터넷 전자메일 전송을 위한 프로토콜, 메일 서버와 메일 서버의 메일 전송을 위한 프로토콜이다.
POP(Post Office Protocol)	서버와 클라이언트간의 메일 전송을 위한 프로토콜
NNTP(Netwoek News Transfer Protocol)	뉴스그룹의 뉴스 전송을 위한 프로토콜
SNMP (System Network Management Protocol)	네트워크 관리를 위한 프로토콜

3-4. 패킷 통신

- 패킷 통신은 데이터를 패킷으로 나누어 전송하는 컴퓨터 통신
- 하나의(한 조각의) 패킷은 1024byte를 전후한 크기(Size)를 가지며 전송
- 전송 할 데이터를 패킷 사이즈로 쪼개어 전송하는 것은 데이터를 한 번에 전송하는 것보다 안정적으로 전송할 수 있기 때문.
- 하나의 패킷에는 도착지의 주소와 일련번호가 포함되어 있어 하나의 회선에 여러 개의 데이터들에서 쪼개진 패킷들이 뒤섞인다 해도 정확하게 도착지까지 전송 가능
- 패킷의 헤더에 포함되는 주소 정보인데 TCP/IP 프로토콜은 이 주소의 형식을 정의하고 있어 반드시 이 형식을 따라야 한다.

3-5. IP Address

- 인터넷에서 사용되는 주소를 **IP 어드레스**.
- 32비트로 구성되어 있으며 이론적으로 2^{32}개의 주소를 사용할 수 있다
- 구조는 3개의 점(Dot)을 이용하여 (210.127.184.26)과 같이 **4부분**으로 기술하고, 각 부분은 8비트(32/4), 즉 1바이트로서 0부터 255까지 사용할 수 있다.
- 인터넷에 접속한 컴퓨터에 부여되는 **고유한 주소**로서 임의로 사용할 수 없으며 중복해서 사용할 수 없다

3-6. IP 어드레스의 클래스(Class)

IP 어드레스는 네트워크 사이즈에 따라 다음과 같은 단계로 구분되어 사용된다.

A class (1.0.0.0 - 127.0.0)	상위 1바이트(1자리)까지 고정된 어드레스를 할당받아 나머지 3바이트를 활용할 수 있다. 즉, 2^{24}개의 어드레스를 사용할 수 있다.
B class (128.0.0.0 - 191.255.0.0)	상위 2바이트(2자리)까지 고정된 어드레스를 할당받아 나머지 2바이트를 활용할 수 있다. 즉, 2^{16}개의 어드레스를 사용할 수 있다.
C class (192.0.0.0 - 223.255.255.0)	상위 3바이트(3자리)까지 고정된 어드레스를 할당받아 나머지 1바이트를 활용할 수 있다. 즉, 2^{8}개의 어드레스를 사용할 수 있다.
D,E,F class (224.0.0.0 - 254.0.0.0)	D,E,F 클래스 어드레스는 현재 사용하지 않고 예비용으로 두고 있다.

3-7. Domain Name

● IP Address는 숫자로 구성되어 있어 기억하기 힘들다. 그러한 숫자를 문자로 표현한 것이 '**도메인네임**'입니다.
● "."으로 구분된 4자리의 영문으로 구성되어 있고 규칙이 있습니다.

일반적인형식> **컴퓨터이름.단체이름.기관성격.국가코드**

예) www.choongang.co.kr -> 중앙그룹의 홈페이지 도메인입니다.
-. 도메인의 제일 앞에 나오는 '**컴퓨터이름**'은 그 컴퓨터를 관리하는 사람이 붙인다.
(주로 WWW을 사용하는데 모두가 그렇지는 않다.)
-. '**단체이름**' 역시 그 단체에서 정한다. 가능하면 모든 사람들이 추측하기 쉬운 걸로 해야만 주소를 모르더라도 접속할 수 있다

[기관성격 도메인]

기관	미국 제외한 국가	미국
학회, 교육기관	ac	edu
기업체, 상업기관	co	com
정부기관	go	gov
네트워크 관련기관	nm, net	net
그외(비영리 단체등)	or	org

[국가코드]

국가	도메인
한국	kr
캐나다	ca
독일	du
일본	jp
영국	uk

3-8. DNS Server

● 도메인 이름을 IP 어드레스로 바꾸어주는 서비스를 DNS서비스라고 하며 이런 서비스를 담당하는 서버를 DNS Server라고 한다.
● DNS 서버는 도메인 이름을 IP어드레스로 변환할 수 있도록 데이터베이스를 저장하고있다.
● 네트워크별로 DNS 서버가 있어 해당 네트워크에 대한 DNS서비스를 제공하는 것이 일반적이다.

3-9. URL(Uniform Resource Locator)

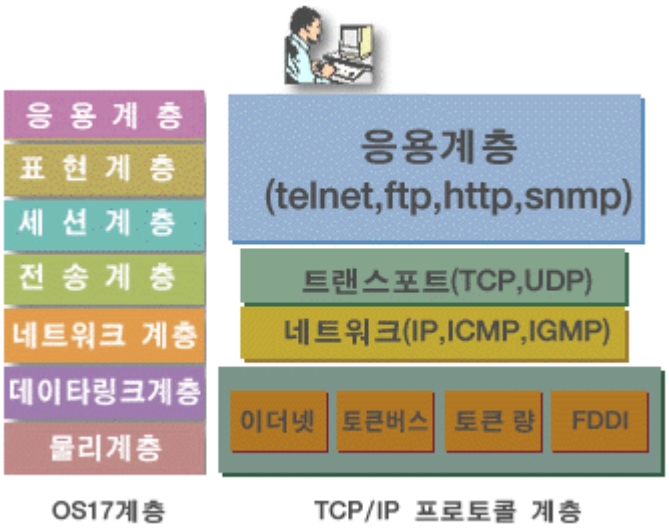
● 월드와이드웹(WWW)에서 사용하는 주소 형식
● 정보의 형태와 위치를 담고 있어 정보 검색에 용이하도록 되어 있다.

```
<표기형식> 프로토콜://호스트이름[:포트번호][/디렉토리경로]
```

[URL에서 사용되는 프로토콜들]

HTTP (hypertext transmission protocol)	월드와이드웹에서 사용하는 프로토콜이며 이미지 등 멀티미디어 정보 표시가능 (예 : http://www.imagenet.co.kr/index.html)
FTP (file transfer protocol)	파일 전송 프로토콜. FTP서버에서 제공하는 파일의 위치를 표시 (예:ftp://ftp.imagenet.co.kr/pub/test.txt)
Gopher	고퍼(Gopher)서버의 정보를 표시 (예 : gopher://cair-archive.kaist.ac.kr)
NEWS	뉴스그룹의 정보를 표시 (예 : news://han.www-forum)
Telnet	가상 단말 서비스를 제공하는 프로토콜 (예 : telnet://gohitel.kol.co.kr)
FILE	FILE 프로토콜은 로칼 컴퓨터의 파일 정보를 표시 (예 : FILE:///c|windows/win.ini) ** 프로토콜 다음의 슬래쉬를 3개 사용하며 드라이브명 다음의 기호는 ":"이 아닌 "|"기호를 사용한다.
Mailto	전자메일 주소를 표시 (예 : mailto://hyoo@mail.imagenet.co.kr)

4. TCP/IP(Transmission Control Protocol/Internet Protocol) & UDP(User Datagram Protocol)



4-1. TCP/IP

● TCP/IP는 인터넷의 기본적인 통신 프로토콜

● 인트라넷이나 엑스트라넷과 같은 사설 망에서 사용된다. 사용자가 인터넷에 접속하기 위해 자신의 컴퓨터를 설정할때 TCP/IP 프로그램이 설치되며, 이를 통하여 역시 같은 TCP/IP 프로토콜을 쓰고 있는 다른 컴퓨터 사용자와 메시지를 주고받거나, 또는 정보를 얻을 수 있게 된다

● OSI의 트랜스포트 이상 상위계층인 세션계층과 표현계층을 응용계층에 포함하였고, 전송 매체에 독립적이어야 하는 요구사항은 물리계층과 데이타링크 계층을 결합하여 네트워크 계층에서 해결한다. 보편적 접속성에 대한 요구 사항은 ip에 의해 해결된다.

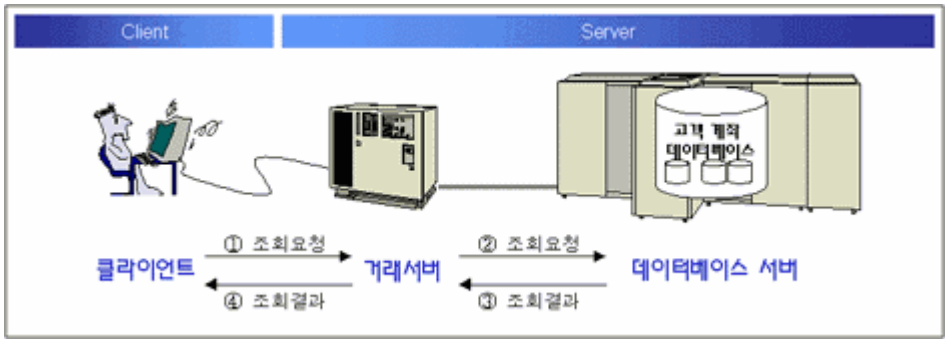
● icmp는 ip를 위한 에러 및 진단 정보를 전송하는 기능을 갖고 있으며, ip를 구현하고 있는 모든 시스템이 공통적으로 가지는 기능이다

● 네개의 계층 중 가장 중요한 2개의 계층으로 이루어져 있다

상위계층인 TCP (Transmission Control Protocol)는 메시지나 파일들을 좀더 작은 패킷으로 나누어 인터넷을 통해 전송하는 일과, 수신된 패킷들을 원래의 메시지로 재조립하는 일을 담당한다.

하위계층인 IP (Internet Protocol)는 각 패킷의 주소부분을 처리함으로써, 패킷들이 목적지에 정확하게 도달할 수 있게 한다. 네트웍 상의 각 게이트웨이는 메시지를 어느 곳으로 전달해야할지를 알기 위해, 메시지의 주소를 확인한다. 한 메시지가 여러 개의 패킷으로 나뉘어진 경우 각 패킷들은 서로 다른 경로를 통해 전달될 수 있으며, 그것들은 최종 목적지에서 재조립된다.

- TCP/IP는 통신하는데 있어 클라이언트/서버 모델을 사용
  (컴퓨터 사용자(클라이언트)의 요구에 대응하여, 네트웍 상의 다른 컴퓨터(서버)가 웹 페이지를 보내는 식의 서비스를 제공)
- TCP/IP는 본래 점대점(点對点) 통신을 하는데, 이는 각 통신이 네트웍 상의 한 점 (또는 호스트 컴퓨터)으로부터 시작되어, 다른 점 또는 호스트 컴퓨터로 전달된다는 것을 의미한다.
- TCP/IP와 TCP/IP를 이용하는 상위계층의 응용프로그램들은 모두 "커넥션리스 (connectionless)"라고 불리는데, 이는 각 클라이언트의 요구가 이전에 했던 어떠한 요구와도 무관한 새로운 요구로 간주된다는 것을 의미한다 (일상적인 전화통화가 통화시간 내내 지속적으로 연결되어 있어야 하는 것과는 다르다).
- 커넥션리스는 네트웍을 독점하지 않으므로, 모든 사람들이 그 경로를 끊임없이 공동으로 사용할 수 있게 한다 (사실 TCP 계층 그 자체는 어떤 한 메시지가 관계되어 있는 한 커넥션리스가 아니라는데 유의해야 한다. TCP 접속은 어떤 한 메시지에 속하는 모든 패킷들이 수신될 때까지 계속 유지된다).

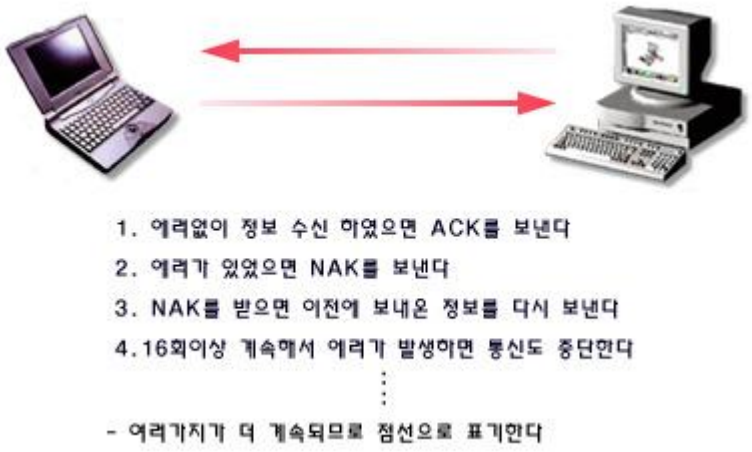


클라이언트/서버 모델을 이용하여 거래내용을 처리하는 것은 매우 보편적인 일이다. 예를 들어, 누군가 자신의 은행계좌 내역을 조회하려고 하는 경우, 먼저 자신의 PC에 있는 클라이언트 프로그램이 은행에 위치한 거래 서버에 그 요구사항을 전송하게 되고, 거래 서버는 다시 계좌내역을 검색해주는 일을 하는 데이터베이스 서버에 그 요구사항을 보내게 된다. 데이터베이스 서버가 계좌내역을 검색하여 그 내용을 거래 서버에 보내면, 거래 서버는 다시 그 내용을 계좌 내역을 요구한 PC의 클라이언트 프로그램으로 보냄으로써, 최종적으로 화면에 나타나게 된다.

4-2. TCP (Transmission Control Protocol)

- 연결형 프로토콜
  (데이터를 주고 받기 위해서 보내는 측이 받는측에게 연결을 요구하고 받는 측이 연결요구를 받아 들이면, 그제서야 데이터를 주고 받기 때문)
- 신뢰성 있는 프로토콜
  보내는 측이 데이터를 주면, 받는 측은 잘 받았다라고 하는 ACK 메시지라는 것을 보내게

되면 중간에 까먹는다거나 바뀐다든지 하더라도 데이터는 정확하게 주고 받게 되는 것이다.



- TCP는 인터넷상의 컴퓨터들 사이에서 데이터를 메시지의 형태로 보내기 위해 IP와 함께 사용되는 프로토콜
- IP가 실제로 데이터의 배달처리를 관장하는 동안, TCP는 데이터 패킷을 추적 관리한다. (메시지는 인터넷 내에서 효율적인 라우팅을 하기 위해 여러 개의 작은 조각으로 나뉘어지는데, 이것을 패킷이라고 부른다).

예를 들면, HTML 파일이 웹 서버로부터 사용자에게 보내질 때, 서버 내에 있는 TCP 프로그램 계층은 파일을 여러 개의 패킷들로 나누고, 패킷 번호를 붙인 다음, IP 프로그램 계층으로 보낸다. 각 패킷이 동일한 수신지 주소(IP 주소)를 가지고 있더라도, 패킷들은 네트웍의 서로 다른 경로를 통해 전송될 수 있다. 다른 한쪽 편(사용자 컴퓨터 내의 클라이언트 프로그램)에 있는 TCP 는, 각 패킷들을 재조립하고, 사용자에게 하나의 완전한 파일로 보낼 수 있을 때까지 기다린다.

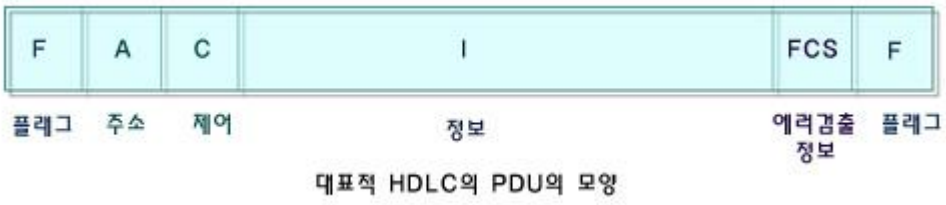
참고 : 프로토콜 데이타 유니트

프로토콜 이용자 정보를 실어 나르기 위해서는 프로토콜 데이터 유니트(PDU: Protocol Data Unit)를 사용한다. 즉 PDU 는 물건을 운반할 때 상자 단위로 포장하여 운반하는 것과 같이 프로토콜이 정보의 운반을 위해서는 PDU 라는 상자를 이용한다.

우리가 상자 단위로 물건을 포장하여 운반할 때 그 상자마다 물품의 내용이나 발송처 수신처 등을 표기하는 것과 마찬가지로 PDU 에도 이용자 정보뿐만 아니라 정보의 발신처, 수신처등의 주소와 전송 등에 에러의 발생이 있었는지를 점검하기 위한 정보, 그밖에 흐름제어 등을 위한 정보 등이 같이 들어가게 된다. 계층화된 프로토콜에서는 계층마다 PDU 이름을 독특하게 붙여 사용하는 경우가 있다.

계층 2 PDU 는 프레임(Frame), 계층 3 PDU 는 패킷(Packet), 계층 4 PDU 는 세그먼트(Segment)등으로 부르는 것이 일반적이다. 이러한 특별한 이름이 없는 경우에는 그냥

몇 계층의 PDU 라고 부르게 된다. 그리고 세그먼트라는 PDU 이름은 TCP 에서 사용하는 경우가 많다.



대표적 HDLC의 PDU의 모양

4-3. UDP (User Datagram Protocol)

● 비연결형 프로토콜

● 신뢰성이 없는 프로토콜

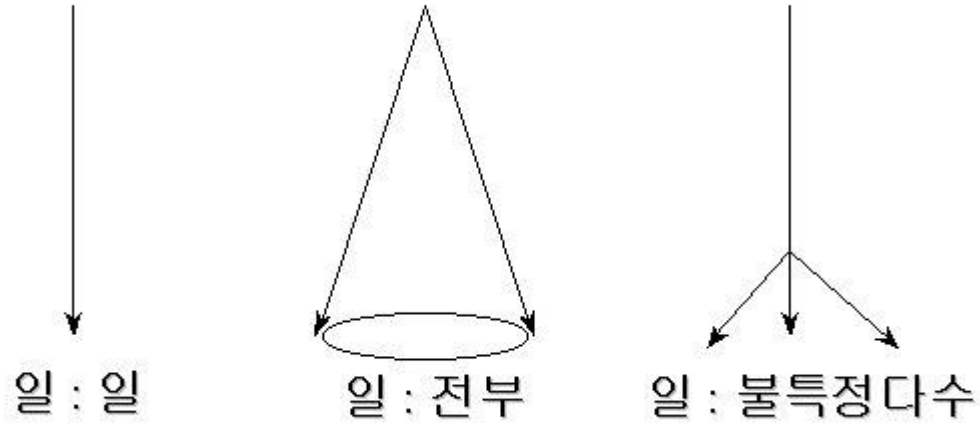
TCP와는 달리, 메시지를 패킷(데이터그램)으로 나누고, 반대편에서 재조립하는 등의 서비스는 제공하지 않으며, 특히 도착하는 데이터 패킷들의 순서를 제공하지 않는다. 이 말은 UDP를 사용하는 응용프로그램은, 전체 메시지가 올바른 순서로 도착했는지에 대해 확인할 수 있어야 한다는 것을 의미한다. 교환해야할 데이터가 매우 적은(그러므로 재조립해야할 메시지도 매우 적은) 네트웍 응용 프로그램들은 처리시간 단축을 위해 TCP 보다 UDP를 더 좋아할 수 있다.

● IP를 사용하는 네트웍 내에서 컴퓨터들 간에 메시지들이 교환될 때 제한된 서비스만을 제공하는 통신 프로토콜이다.

● TCP의 대안(代案)이며, IP와 함께 쓰일 때에는 UDP/IP라고 표현하기도 한다. TCP와 마찬가지로 UDP도 한 컴퓨터에서 다른 컴퓨터로 데이터그램이라고 불리는 실제 데이터 단위를 받기 위해 IP를 사용한다.

● UDP는 IP 계층에서 제공되지 않는 두 개의 서비스를 제공하는데, 하나는 다른 사용자 요청을 구분하기 위한 포트 번호와, 도착한 데이터의 손상여부를 확인하기 위한 체크섬 기능이다.

5. Unicast/ Broadcast/ Multicast



(Unicast) (Broadcast) (Multicast)

● 유니캐스트(Unicast)

-. Point-to-Point

-.무전기나 전화와 같이 일대일 로 연결

● 브로드 캐스트(Broadcast)

-.Point-to-MultiPoint

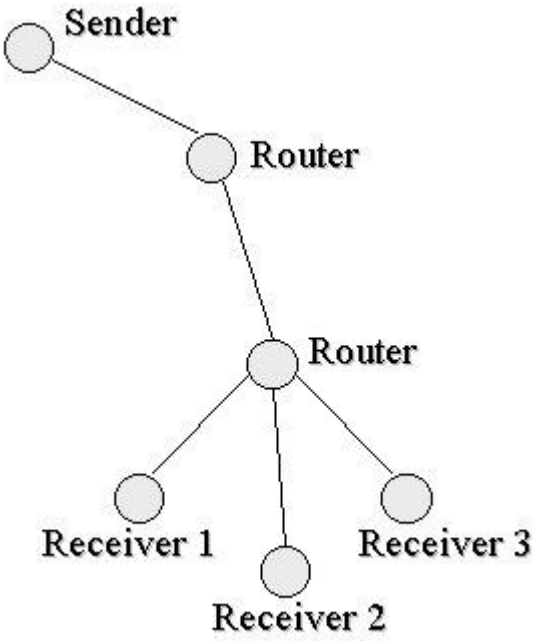
-.공중파 방송

● 멀티캐스트(Multicast)

-.Point-to-MultiPoint

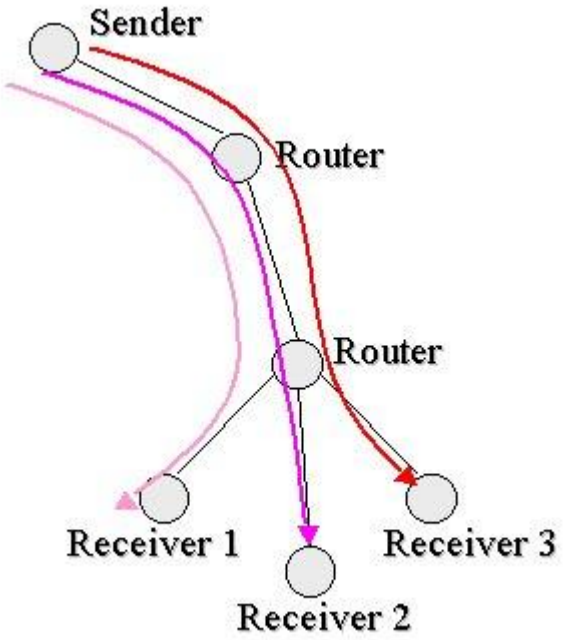
[가상의 네트웍 - 호스트(host) 들과 라우터(router) 들로 이루어짐]

1] 데이터를 보내는 Sender 가 있을 것이고, 그것을 받는 Receiver 가 있다고 가정



2] 유니캐스트는 일대 일로 데이터를 주고 받는다

Sender 에서 Receiver 1, 2, 3 에게 동시에 데이터를 주려고 할때에 다음과 같이 될 것이다. 일대 일로 통신을 하기 때문에 Sender 는 Receiver1 에게 한번주고, Receiver 2 에게도 한번주고, Receiver 3 에게도 한번을 줘야 한다.



3] 멀티캐스트

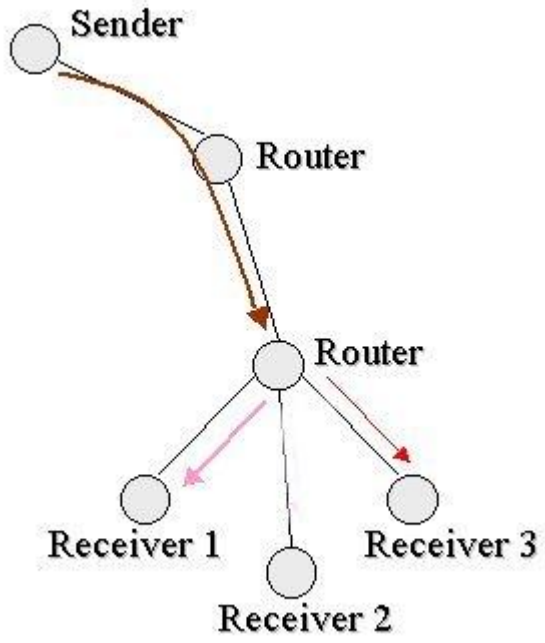
-. 일대 불특정 다수

-. 데이터를 받고자 하는 특정 호스트만 데이터를 받게 된다.

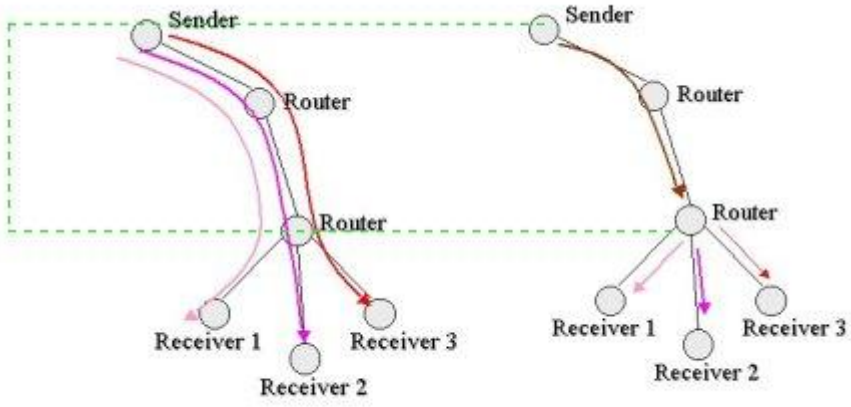
-. 유니캐스트와는 달리 Sender 와 Sender 측 라우터 에서 Receiver 측 라우터 까지 데이터의 흐름이 한번밖에 없다

-. 유니캐스트의 단점을 멀티캐스트로 해결

-. 유니캐스트의 단점은 이전 유니캐스트의 그림처럼 받고자 하는 호스트에게 전부가 일대 일로 데이터를 주어야 하기 때문에 네트웍의 대역폭을 많이 차지 하게 된다. 이 단점을 해결하기 위해서 멀티캐스트는 Sender 측에서 Receiver 측의 라우터에게 한번만 데이터를 주면 라우터가 그 데이터를 그하위 네트웍에 속한 호스트중 받고자 하는 호스트에게만 데이터를 복사해 전달한다



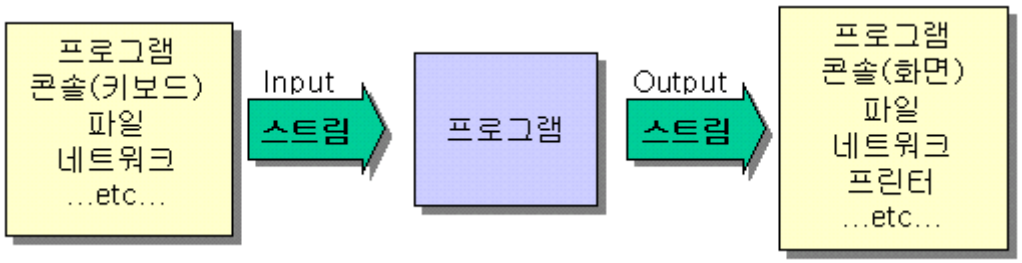
4] 멀티캐스트



1. 스트림의 이해(입력 스트림 / 출력 스트림)

스트림에는 사실 여러 종류가 있다. byte 스트림, 문자열스트림, 데이터그램스트림 등이 있다.
그 중 문자열(String)스트림만을 다루어 보겠다. byte 스트림과 문자열 스트림은 12 라는 정수가 "12"로 전송되느냐(문자열스트림) 아니면 0C(12)인 숫자(byte 스트림)으로 가느냐의 차이가 있을 뿐 거의 비슷하다. 데이터그램 스트림은 약간 다르지만 문자열 스트림을 알고 나면 금방 이해할 수 있을 것이다.

다 알고 있는 내용으로 스트림은 입력과 출력이 있다. 즉 스트림을 보내는 것과 받는 것, 두 가지 동작이 있을 수 있다 것이다. 프린터에 출력하는 경우, 프로그램에서는 프린터스트림에 보내는(출력) 역할을 할 것이고, 프린터는 그것을 스트림에서 받아서(입력) 인쇄하는 구조를 가지고 있을 것이다.



[스트림 입력과 출력]

2. Stream 의 사용

C 와 같은 다른 언어와 마찬가지로 자바에서도 InputStream 과 OutputStream 을 지원하고 있다. 즉 이 클래스들을 입력 또는 출력 하고자 하는 적당한 객체에 연결시켜준 다음 read 나 write method 등을 사용하여 입력과 출력을 하게 되는 것이다. 만일 키보드로부터 입력을 받아서 화면에 출력하고자한다면,

```
InputStream in=System.in;
OutputStream out=System.out;
```

와 같이 선언하여, in.read()문으로 내용을 입력 받아서 out.write()메소드를 사용하여 출력할 수 있다. 만약 in 과 out 이 각각 file 이나, 프린터 등에 연결되어 있다면 물론 file 에서 내용을 읽거나 쓸 수 있고, 프린터에 출력할 수도 있다.

자바에서는 화면에 출력하기 위해서 콘솔(화면)을 사용하는 경우는 거의 없습니다.
하지만 콘솔(키보드)에서 입력 받기 위한 경우나, 프린터에 출력 하고자 하는 경우, file 의 내용을 읽거나 쓰는 경우에는 반드시 스트림을 사용합니다. 또한 TCP/IP Protocol 을 이용한 인터넷에서의 통신에서도 물론 스트림을 사용한다.

package java.net

NetworkEx1.java (InetAddress Class Example)

```
import java.io.*;
import java.net.*;

public class NetworkEx1{
	public static void main(String args[])throws IOException{
		BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
		System.out.print("₩n₩nInput Domain name? : ");
		String name = input.readLine();
		InetAddress inet = InetAddress.getByName(name);
		String st = inet.getHostAddress();
		System.out.println("₩nIP Address is " + st);
	}
}
////////////////////// Console ////////////////////
Input Domain name? : www.daum.net
IP Address is 211.233.28.116
//////////////////////////////////////////////////////
```

NetworkEx2.java (URL Class Example)

```
import java.io.*;
import java.net.*;

public class NetworkEx2{
	public static void main(String args[])throws IOException{
		BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
		System.out.print("₩n₩nInput Domain? : ");
		String name = in.readLine();
		try{
			URL url = new URL(name);
			InputStream is = url.openStream();
			BufferedReader b = new BufferedReader(new InputStreamReader(is));
			//한줄씩 읽어준다.
			String str = "";
			while((str = b.readLine()) != null)
				System.out.println(str);
			}catch(Exception ex){}
	}
}
////////////////////////// Console ////////////////////////
Input Domain? : http://www.daum.net
<html>
<head>
<title>Daum - 우리 인터넷, Daum</title>
.
</html>
//////////////////////////////////////////////////////////
```

♪ Socket in TCP/IP protocol ♪

1. Socket

사실 awt 나 swing 과 같은 Abstract Window Toolkit 을 사용하면 키보드로부터 입력 받는 것을 직접 구현할 필요도 없고, 프린터 출력 부분도 사실 남이 작성해 놓은 코드를 복사해서 약간 수정하면 쉽게 할 수 있다. file 입출력도 스트림을 제대로 설정만 해놓으면 화면에 출력하듯이 간단하게 입력과 출력을 사용할 수 있다. 하지만 socket 을 이용한 프로그램만은 그렇지가 않다. 물론 개념은 스트림을 사용하는 다른 작업들과 크게 다르지 않지만 기술적인 면에 있어서 원하는 기능을 구현하는 것이 그리 쉽지만은 않기 때문이다.
먼저 소켓을 정의해서 스트림을 사용하여 입력과 출력을 받는 것을 구현하는 간단한 Program 을 작성해보고, 하나의 서버에 여러 개의 클라이언트가 접속하는 멀티 클라이언트 Chatting Program 을 진행하겠다.

1-1. 서버와 클라이언트

Socket Program 의 작동 순서.

1. 서버가 소켓을 생성하고 연결을 기다린다.
2. 클라이언트에서 서버의 소켓에 연결한다.
3. 클라이언트에서 데이터를 전송한다.
4. 서버에서 데이터를 받아서 화면에 출력한다.
5. 서버와 클라이언트의 소켓 연결을 끊는다.

예제 프로그램>
Server Side Program : SocketServer.java
Client Side Program : SocketClient.java

```
C:₩Example>javac *.java
C:₩Example>start java SocketServer 6000
C:₩Example>java SocketClient 6000 211.35.136.174
```

SocketServer 에서 6000 은 서버 실행 시 포트값을 나타내며, 주지 않으면 5777 번이 포트번호가 된다. 포트번호는 3000 번 이상의 값을 주는 게 좋다.
SocketClient 실행 시 파라미터는 포트번호와 IP 를 가지고 실행한다. 여기서 포트번호는 반드시 서버실행시의 포트번호와 일치하여야 한다. 또한 IP 는 서버가 실행되고 있는 곳(접속하고자 하는 곳)의 IP 를 넣어준다. 이때 자신의 컴퓨터가 서버일 경우 localhost or 127.0.0.1 번을 주면 된다.
위와 같이 실행하게 되면 클라이언트는 서버에 연결하여 열글자의 데이터를 보낸 후 종료하게 되고 서버도 소켓을 끊고 종료하게 된다.
소켓을 사용하여 어떤 프로그램과 연결을 하기 위해서는 그 Server 프로그램이 실행 되고있는 컴퓨터의 IP 와 ServerSocket 을 만든 port 번호를 알고 있어야 한다.

SocketServer.java

```
// 소켓을 하나 열어서 클라이언트 소켓에서 10 글자를 얻어오고 소켓을 닫는다.

import java.io.*;
import java.net.*;
import java.util.*;

public class SocketServer {

	// 소켓을 만들기 위한 인스턴스들
	private ServerSocket serverSocket = null;
	private Socket socket = null;

	// 소켓에서 스트림을 얻어오기 위한 인스턴스들
	private InputStream is = null;
	private InputStreamReader reader = null;

	public SocketServer(int port) throws IOException {
		try {
		// 서버 소켓을 생성하고, 클라이언트에서 스트림을 받아드릴 소켓을 하나 더 생성합니다.
			serverSocket = new ServerSocket(port);
			System.out.println("소켓을 생성하여 클라이언트의 연결을 기다립니다.
					포트번호 = " + port);
			Socket = serverSocket.accept();
			System.out.println("클라이언트와 연결되었습니다.");
		} catch (IOException e) { throw e; }
	}

	public void startSocket() {

		String str = "";

		try {
			// 소켓에 스트림을 연결하고, 스트림을 읽어들일 reader 를 만듭니다.
			is = socket.getInputStream();
			reader = new InputStreamReader(is);
		} catch (IOException e) { System.out.println("소켓 연결에 실패했습니다."); }

		for (int i=0; i<10; i++) {
			try {
				str = str + (char)reader.read();
			} catch (IOException e) {
				System.out.println("데이터를 받는 데에 실패했습니다.");
			}
		}
		System.out.println("받은 데이터 : " + str);
		try {
			serverSocket.close();
			socket.close();
			reader.close();
			is.close();
		} catch (IOException e) { System.out.println("소켓을 닫는데에 실패했습니다."); }
	}

	public static void main(String args[]) {
		SocketServer s;
		int port = 5777;

		if (args.length>0) {
			port = Integer.parseInt(args[0]);
		}

		try {
			s = new SocketServer(port);
			s.startSocket();
		} catch (IOException e) { System.out.println("소켓 생성에 실패했습니다."); }
	}
}
```

SocketClient.java

```
// 서버에 연결하여 10 글자를 보낸다.

import java.io.*;
import java.net.*;
import java.util.*;

public class SocketClient {
    // 소켓을 만들기 위한 인스턴스들
    private Socket socket=null;

    // 소켓에서 스트림을 얻어오기 위한 인스턴스들
    private OutputStream os = null;
    private OutputStreamWriter writer = null;

    // 생성자. 주어진 IP 와 포트번호로 소켓을 생성한다.
    public SocketClient(String ip, int port) throws IOException {
        try {
            // 서버 소켓을 생성하고, 클라이언트에서 스트림을 받아드릴 소켓을 하나 더 생성합니다.
            socket=new Socket(ip, port);
            System.out.println("소켓을 생성하여 서버와 연결하였습니다.");
        } catch (IOException e) {
            throw e;
        }
    }

    public void startSocket() {
        //String str="abcdefghij";
        String str="나랏말싸미뒹귁에달아";
        try {
            // 소켓에 스트림을 연결하고, 스트림을 보낼 Writer 를 만듭니다.
            os = socket.getOutputStream();
            writer = new OutputStreamWriter(os);

        } catch (IOException e) {
            System.out.println("소켓 연결에 실패했습니다.");
        }
        try {
            writer.write(str,0,10);
            writer.flush();
        } catch (IOException e) {
            System.out.println("데이터 전송에 실패했습니다.");
        }
        System.out.println("보낸 데이터:" + str);
        try {
            socket.close();
            writer.close();
            os.close();
        } catch (IOException e) {
            System.out.println("소켓 닫기에 실패했습니다");
        }
    }

    public static void main(String args[]) {
        SocketClient c;
        String ip="127.0.0.1";
        int port=5777;

        if (args.length>1) {
            ip=args[1];
            port=Integer.parseInt(args[0]);
        } else if (args.length==1) {
            port=Integer.parseInt(args[0]);
        }
        try {
            c=new SocketClient(ip, port);
            c.startSocket();
        } catch (IOException e) {System.out.println("소켓 생성에 실패했습니다.");}
    }
}
```

1-2. Server Side Program(SocketServer.java)

1) ServerScoket & Socket Class

통신을 하기 위해 서버에서는 소켓을 준비해야 하는데 자바에서는 java.net packet 에서 ServerSocket 이라는 클래스가 제공된다.

```
ServerSocket ss = new ServerSocket(int port);
```

< ServerSocket 생성예 >

```
ServerSocket ss = new ServerSocket(1004);
```

서버측에 생성된 서버소켓을 사용하여 클라이언트와 통신하게 될 소켓을 생성해야 한다. 서버 쪽에서는 반드시 두 개의 소켓이 필요하다. 하나는 서버용 소켓인데 이것으로는 통신을 할 수가 없고 단지 클라이언트의 연결을 기다린다. 따라서 클라이언트와 연결이 되면 실제로 통신을 하기 위해서는 소켓을 하나 더 만들어서 연결해주어야 한다. 이 Socket 객체는 ServerSocket Class 가 가지고 있는 accept() method 를 이용하여 만들어 줄 수 있다.

```
Socket soc = ss.accept();
```

여기서 ss 는 위에서 생성한 서버소켓이다. 실제 클라이언트와 통신을 담당하게 되는 클래스가 Socket Class 이다. 클라이언트에서도 이 소켓을 사용하여 연결하게 된다.

2) 입/출력 Stream

입/출력을 위한 스트림을 선언. 입출력의 경우 모든 작업이 소켓을 통해서 이루어 지기 때문에 I/O Stream 의 객체는 soc(Socket 객체)로부터 얻어 와야 한다.

```
InputStream is = soc.getInputStream();
InputStreamReader isr = new InputStreamReader(is);
BufferedReader br = new BufferedReader(isr);

OutputStream out = soc.getOutputStream();
OutputStreamWriter osw = new OutputStreamWriter(out);
```

데이터를 읽기 위해서는 InputStreamReader 에 있는 read() method 를 사용한다. int a = isr.read();와 같이 하면 a 에 한 글자가 입력된다. int 값으로 넘어오는데, 문자로 바꾸기 위해서는 형변환(type casting)을 해주면 된다. char a = (char) isr.read(); 와 같이 하면 된다. 들어오는 문자들을 하나의 문자열에 저장하기 위해서는 str = str + (char)isr.read(); 와 같이 문자열에 더해주거나 또는 아래서 생성한것과 같이 BufferedReader 가 가지고 있는 readLine() method 를 이용하면 된다
예를 들어 String str = in.readLine();을 쓰게 되면 한 줄 단위로 문자열을 얻을 수 있다. 즉 '\n'을 기준으로 해서 얻는다.

1-3. Client Side Program(SocketClient.java)

1) Client Socket

클라이언트 측에서 사용하는 Socket 은 ServerSocket 이 필요 없고, InputStream 을 OutputStream 으로, InputStreamReader 를 OutputStreamWriter 로 바꾸기만 하면 됩니다.

Client 에서 사용하는 Socket 의 객체를 생성하기 위해서는 앞에서 생성한 서버의 IP 와 port 번호를 알고 있어야 한다.

```
Socket soc = new Socket("127.0.0.1", 1004);
```

여기서 "127.0.0.1"은 Server 측의 IP 주소이고, 1004 는 포트번호입니다.

2) 입/출력 Stream

서버로 데이터를 보내는 경우는 OutputStream 을 서버에서 전달된 데이터를 받을 경우는 InputStream 을 사용한다..

```
InputStream is = soc.getInputStream();
InputStreamReader isr = new InputStreamReader(is);
BufferedReader br = new BufferedReader(isr);

OutputStream out = soc.getOutputStream();
OutputStreamWriter osw = new OutputStreamWriter(out);
```

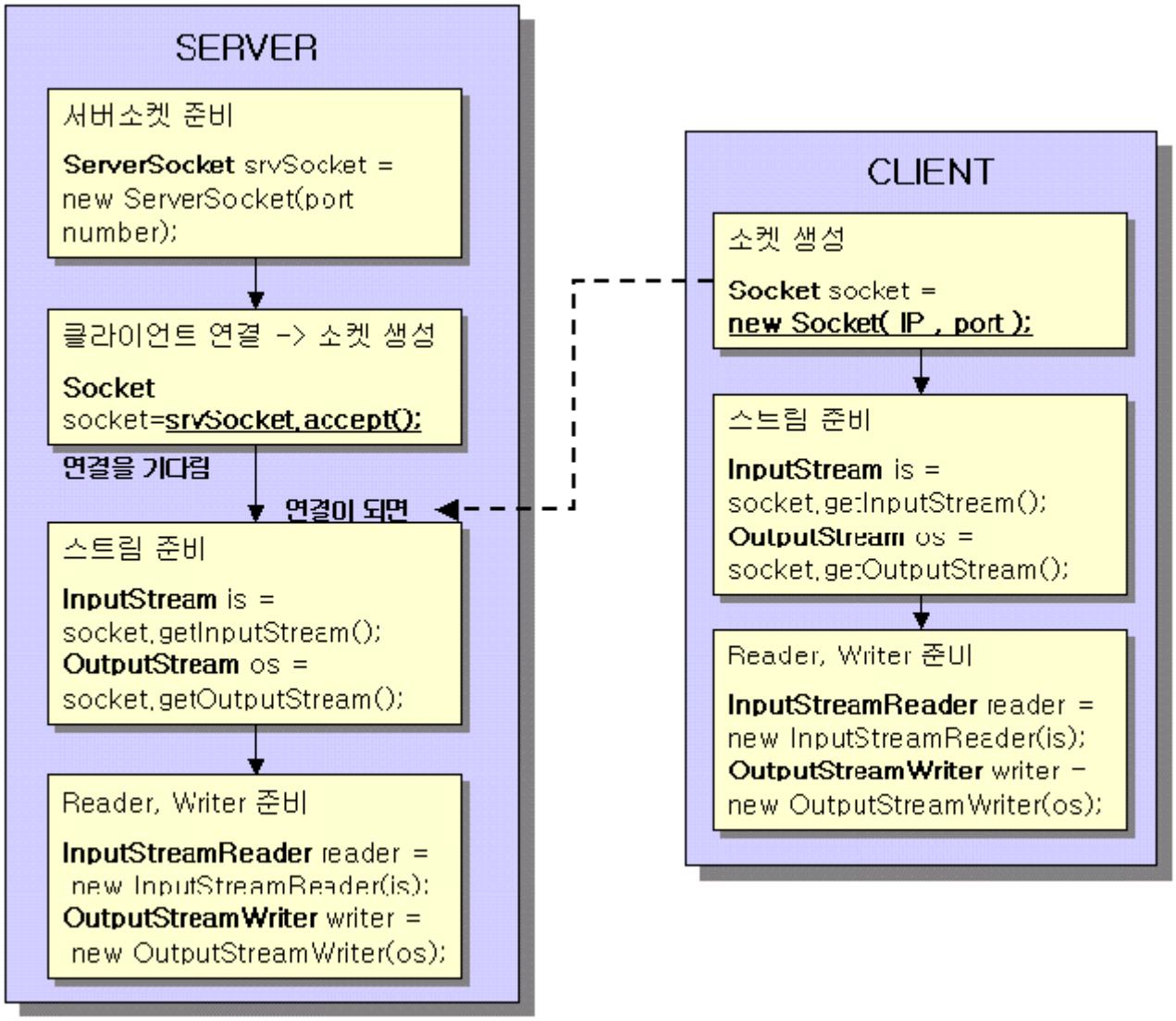
데이터를 전송하기 위해서는 OutputStream 이 가지고 있는 write()메소드를 이용하여 데이터를 전송하게 된다.

보내는 쪽 :

```
String str = “안녕하세요”;
out.write((str+”\n”).getBytes());
```

받는 쪽 :

```
String msg = br.readLine();
```


SERVER
서버소켓 준비
ServerSocket srvSocket = new ServerSocket(port number);
클라이언트 연결 -> 소켓 생성
Socket socket=srvSocket.accept();
연결을 기다림
연결이 되면
스트림 준비
InputStream is = socket.getInputStream();
OutputStream os = socket.getOutputStream();
Reader, Writer 준비
InputStreamReader reader = new InputStreamReader(is);
OutputStreamWriter writer = new OutputStreamWriter(os);
CLIENT
소켓 생성
Socket socket = new Socket(IP , port);
스트림 준비
InputStream is = socket.ge:InputStream();
OutputStream os = socket.ge:OutputStream();
Reader, Writer 준비
InputStreamReader reader = new InputStreamReader(is);
OutputStreamWriter writer = new OutputStreamWriter(os);

Socket을 이용한 간단한 Echo Chat

EchoServer.java

```
import java.awt.*;
import java.io.*;
import java.net.*;

class EchoServer{
  public static void main(String arg[]){
    ServerSocket ss = null;
    int port = 1000;
    try{
      ss = new ServerSocket(port);
    }catch(Exception e) {System.out.println("error ");}

    System.out.println("server wating ......");
    Socket s = null;
    try{
      s = ss.accept(); //accept에 의해 연결됨..
    }catch(Exception e) {System.out.println("error ");}
    try{
      PrintWriter out = new PrintWriter(s.getOutputStream(), true);
      BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
      String inputLine;
      System.out.println("Socket연결 성공");
      while((inputLine = br.readLine()) != null){
        out.println(inputLine);
        System.out.println("[" + s.getInetAddress().getHostName()+"]"+inputLine);
      }
      out.close();
      br.close();
      s.close();
      ss.close();
    }catch(Exception e){}
  }
}
```

EchoClient.java

```
import java.io.*;
import java.net.*;

public class EchoClient{
   public static void main(String arg[]){
      Socket s = null;
      PrintWriter out = null;
      BufferedReader br = null;
      try{
         s = new Socket(arg[0],   Integer.parseInt(arg[1]));
         out = new PrintWriter(s.getOutputStream(), true);
         br = new BufferedReader(new InputStreamReader (s.getInputStream()));
      }catch(UnknownHostException uh){System.out.println("error");}
      catch(IOException ie){}
      try {
         BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
         String userInput;
         while((userInput = stdIn.readLine()) != null){
            System.out.println(userInput.length());
            out.println(userInput);
            System.out.println("Echo " + br.readLine());
         }
         out.close();
         br.close();
         stdIn.close();
      }catch(Exception e) {System.out.println("error ");}
   }
}
```

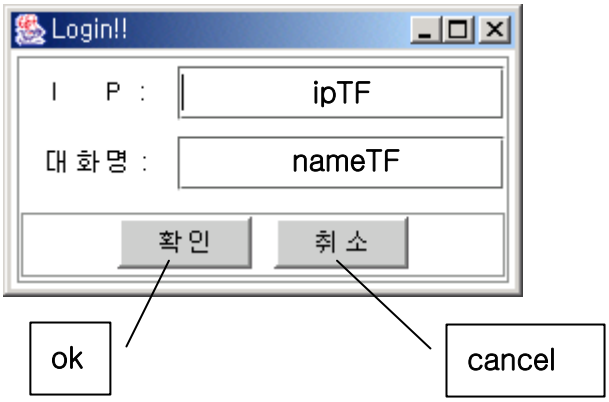
실행>

```
javac *.java
java EchoServer
java EchoClient localhost 1000
```

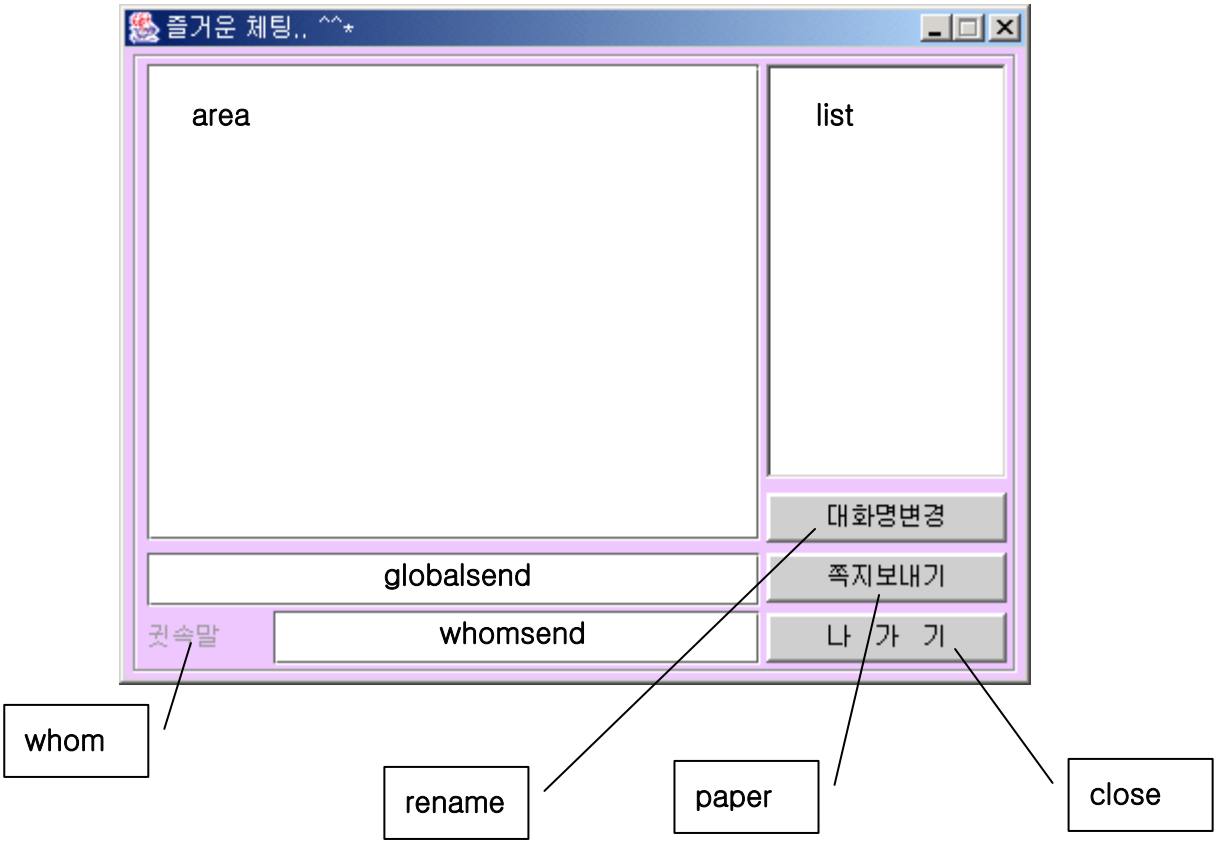
위 chatting program의 문제점을 생각해 보고 해결책은??

Socket을 이용한 MultiChat Programming

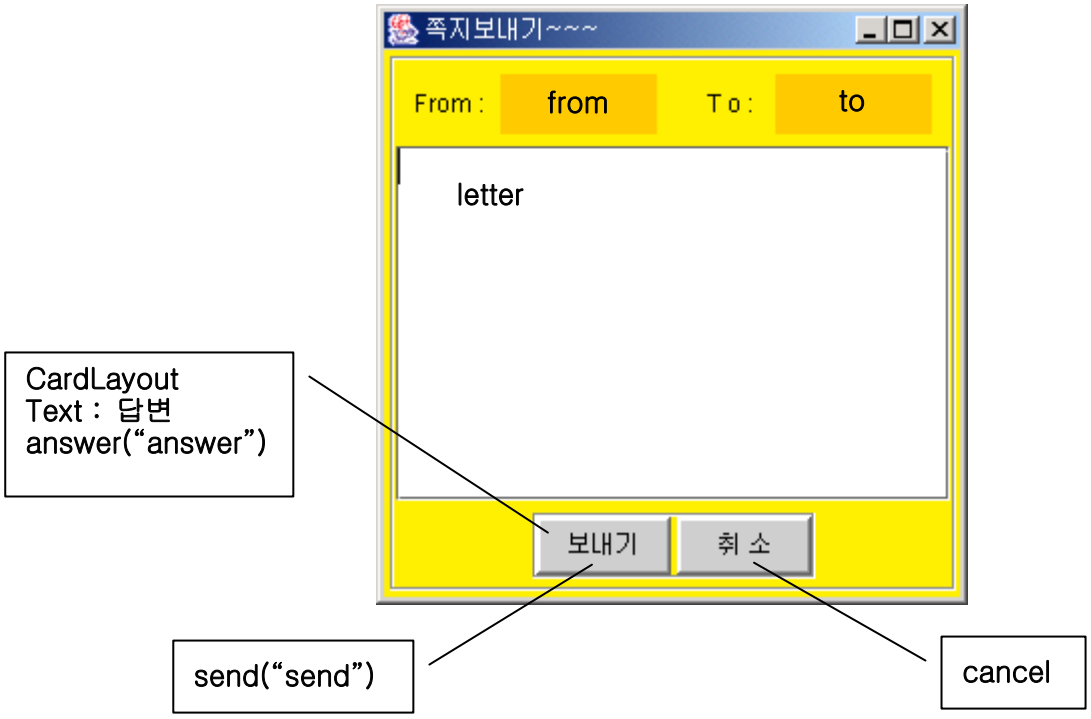
< Login.java >



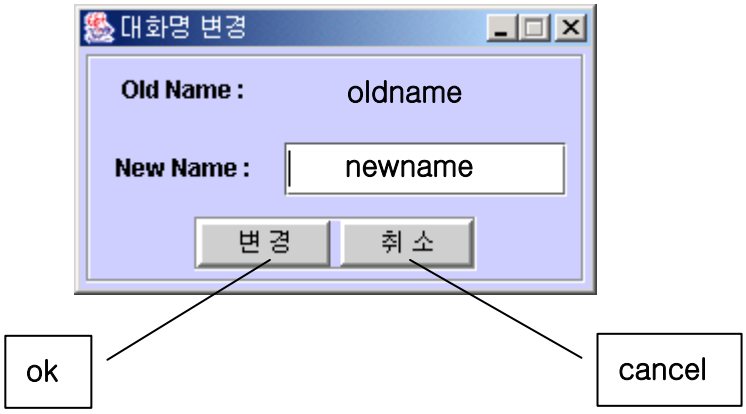
< Chat.java >



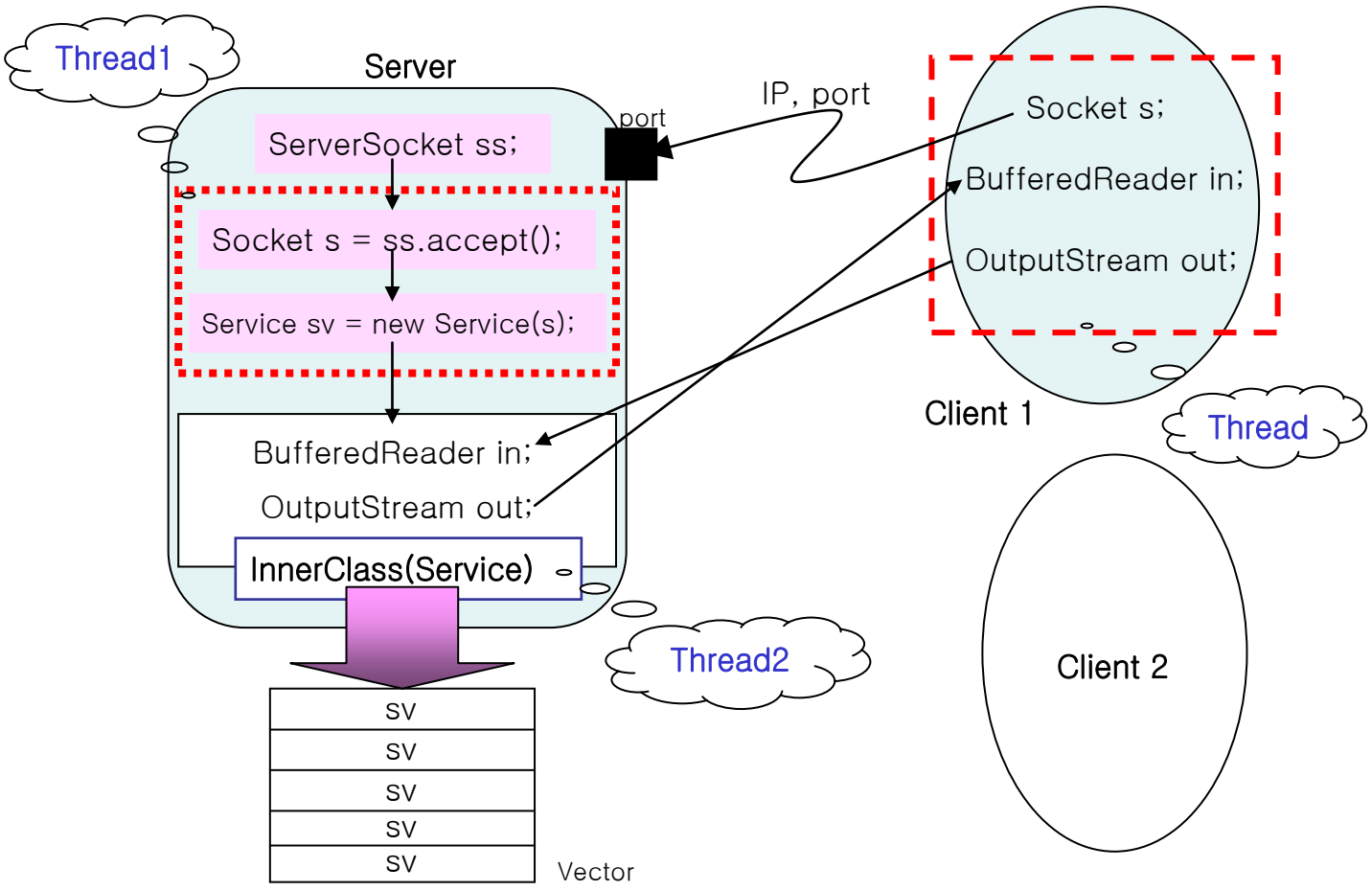
< Paper.java >



< ReName.java >



Chat Program의 기본 원리



[그림 Chat Program의 원리]

Chat Programming에서 사용되는 주요 class	
Server Side	Client Side
ServerSocket ss Socket s InnerClass(Service) BufferedReader in OutputStream out Vector Thread 2개	Socket BufferedReader in OutputStream out Thread

Server.java 기본 작업

```
import java.io.*;
import java.util.*;
import java.net.*;

public class Server extends JFrame implements Runnable{
   Vector globalvc = new Vector(); //전체 구성원 관리
   Vector roomvc = new Vector(); //방 정보 관리

 public void run(){
     ServerSocket ss = null;
     try
     {
        ss = new ServerSocket(3456);
     }catch(Exception e){}

     while(true){
        try{
           Socket s = ss.accept();
           System.out.println("접속한 클라이언트 : " + s);
           Service sv = new Service(s);
           sv.start();
        }catch(Exception ex){}
     }
  }

  public static void main(String[] args) {
     Server server = new Server();
     new Thread(server).start();
  }

  class Service extends Thread{
     String myname, myroom;
     BufferedReader in;
     OutputStream out;
     Socket s;

     Service(Socket s){
        try{
           this.s = s;
           in = new BufferedReader(new InputStreamReader(s.getInputStream()));
           out = s.getOutputStream();
        }catch(Exception ex){}
     }
     public void run(){
        while(true){
           try{
              String msg = in.readLine();
              if(msg == null)
                  return;
              System.out.println("클라이언트로부터 받은 메시지: " + msg);
              StringTokenizer st = new StringTokenizer(msg, "|");
              int protocol = Integer.parseInt(st.nextToken());
              switch(protocol){

               // 내용 추가//////////////////////////////

              }
           }catch(Exception e) {}
        }//while end
     } //run end
```

```
    //모든 클라이언트에게 전송할 때 사용하는 메소드
    protected void putMessageAll(String msg){
      synchronized(this){
        for(int i=0;i<globalvc.size();i++){
          Service sv = (Service)globalvc.elementAt(i);
          try{
            sv.putMessageTo(msg);
          }catch(Exception ex){
            globalvc.removeElementAt(i--);
          }
        }
      }
    } //putMessageAll end

    //특정 방에 존재하는 클라이언트에게 메시지 전송하는 메소드
    protected void putMessageRoom(String msg){
      synchronized(this){
        for(int i=0;i<roomvc.size();i++){
          Room room = (Room)roomvc.elementAt(i);
          if(myroom.equals(room.title)){
            for(int j=0;j<room.uservc.size();j++){
              Service sv = (Service)room.uservc.elementAt(j);
              try{
                sv.putMessageTo(msg); //개인에게…
              }catch(Exception ex){
                room.uservc.removeElementAt(j--);
              }
            }
            break;
          }
        }
      }
    } //putMessageRoom end

    //특정 클라이언트에 메시지 전송 메소드
    protected void putMessageTo(String msg) throws Exception{
      synchronized(this){
        try{
          out.write((msg+"₩r₩n").getBytes());
        }catch(Exception ex){}
      }
    }//putMessageTo end

  } //Service end
}//전체 클래스 끝.
```