5. TCP와 UDP

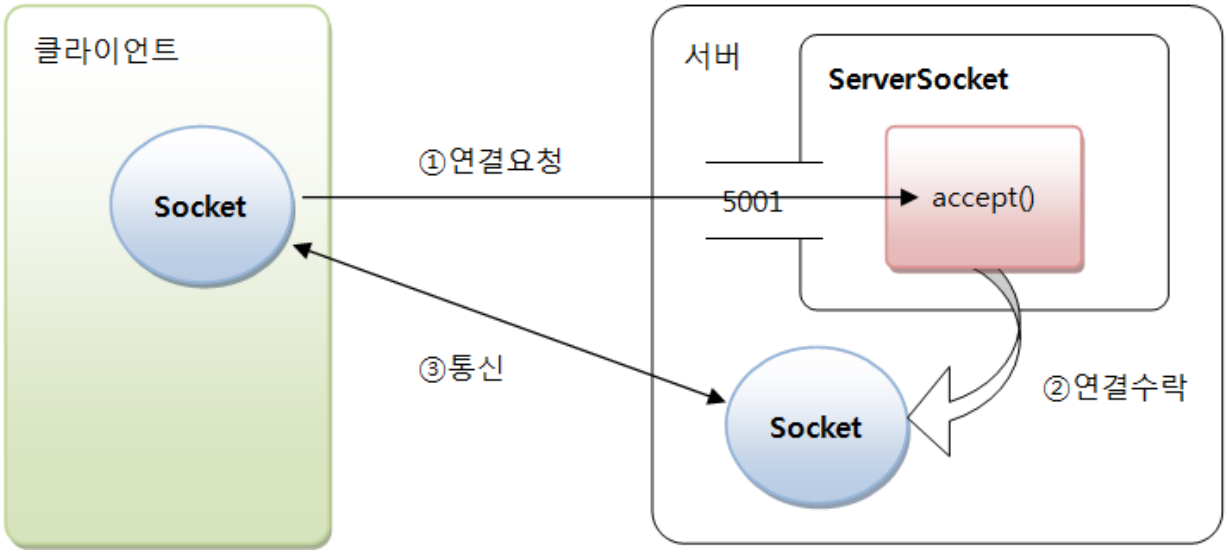
1. 자바 소켓 클래스
2. TCP 네트워킹
3. UDP 네트워킹

1. 자바 소켓 클래스

❖ TCP(Transmission Control Protocol)

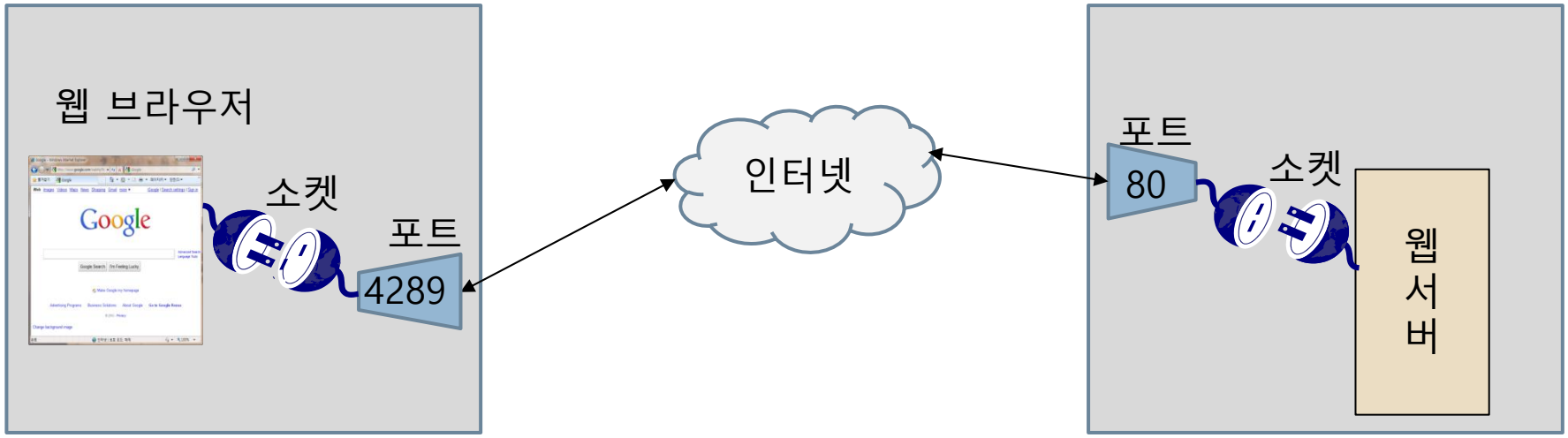
- 연결 지향적 프로토콜 -> 시간 소요
- 통신 선로 고정 -> 전송 속도 느려질 수 있음
- 데이터를 정확하고 안정적으로 전달

- java.net API
  - ServerSocket, Socket
- ServerSocket과 Socket 용도

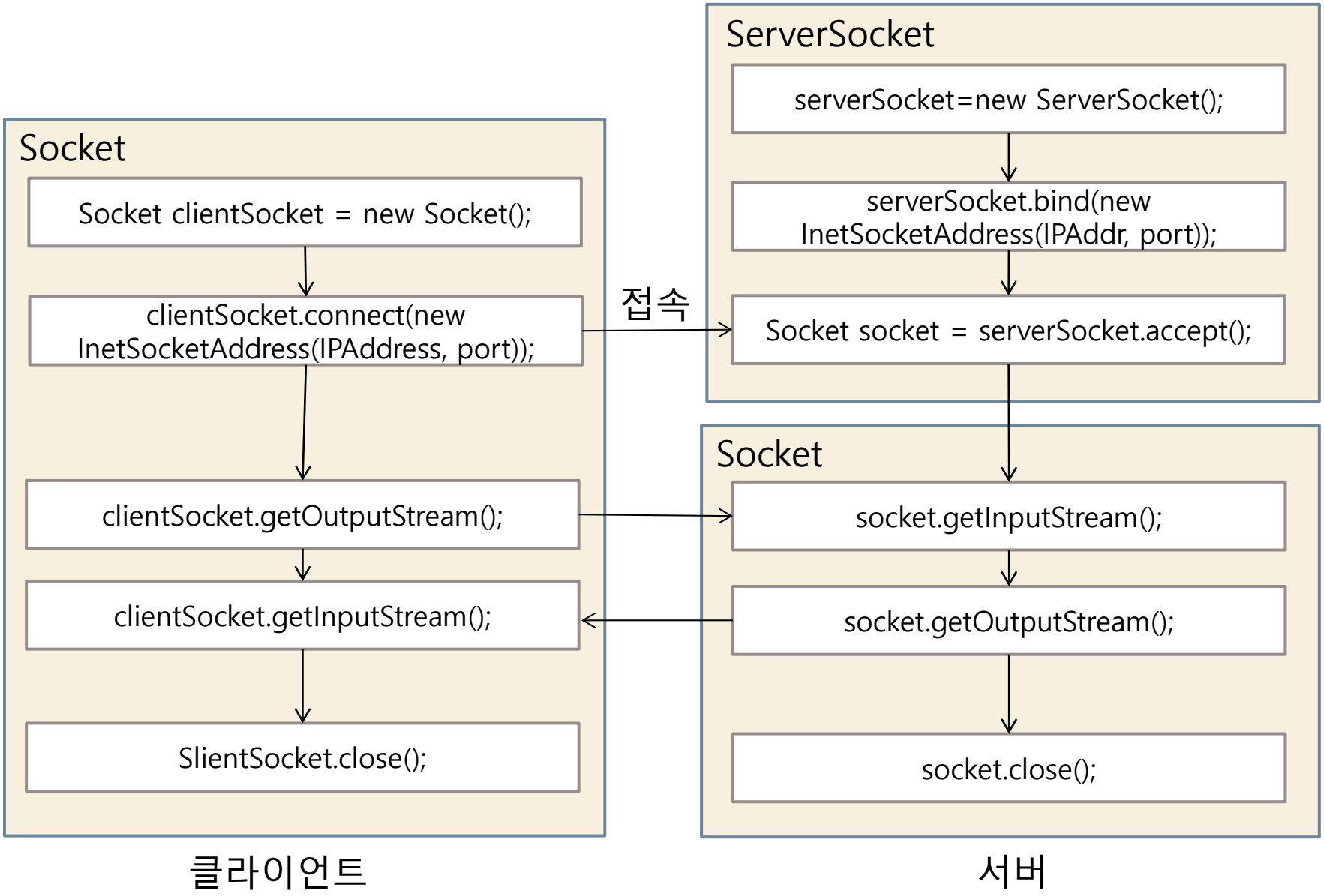


소켓 프로그래밍

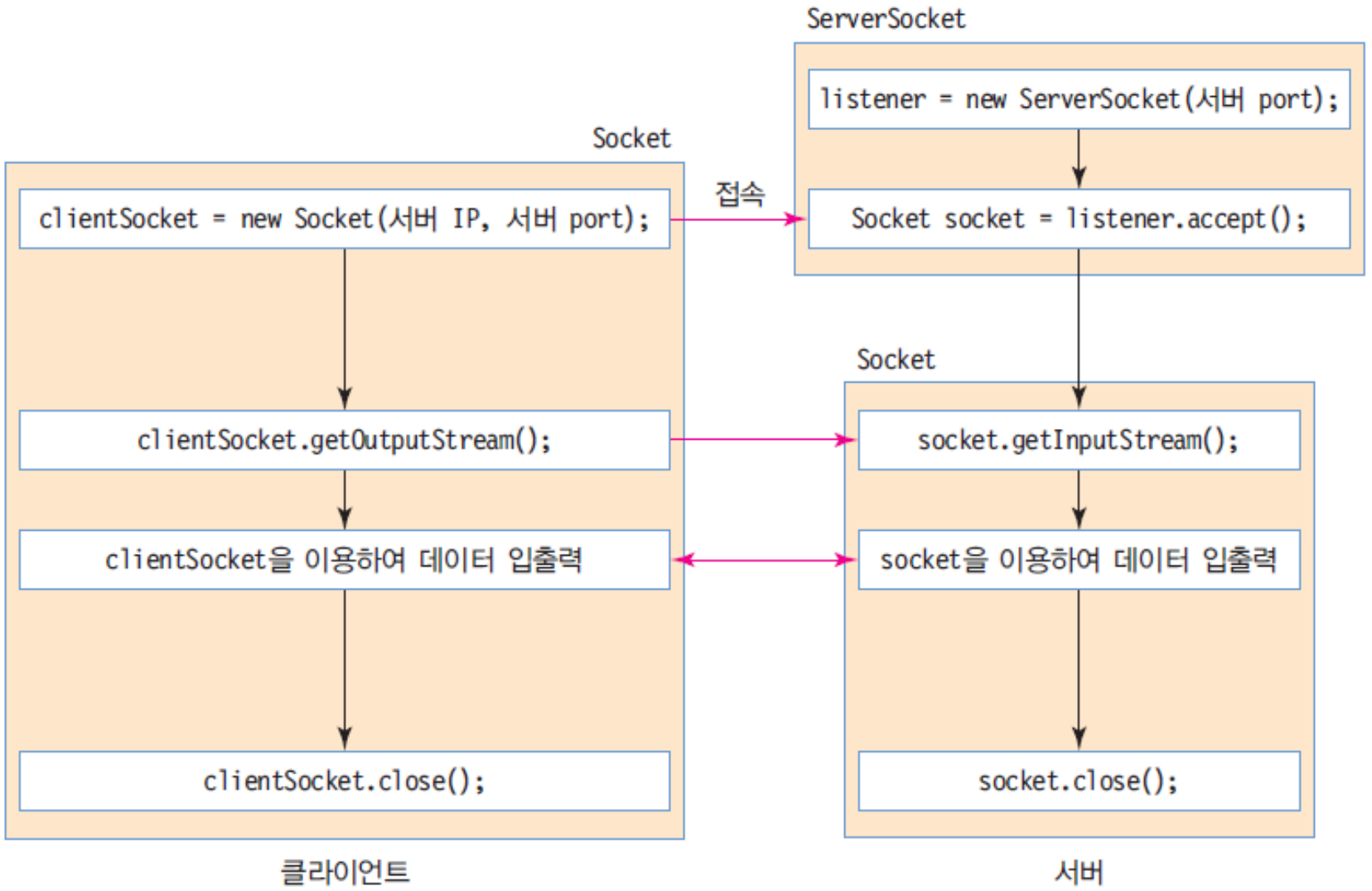
- **소켓 (socket)**
  - 소켓은 네트워크 상에서 수행되는 두 프로그램 간의 양방향 통신 링크의 한쪽 끝 단을 의미
  - 소켓은 특정 포트 번호와 연결되어 있음
    - TCP에서 데이터를 보낼 응용 프로그램을 식별할 수 있음.
  - 자바에서의 데이타 통신 시 소켓 사용
  - 소켓 종류
    - 서버 소켓과 클라이언트 소켓



소켓을 이용한 서버 클라이언트 통신 프로그램의 구조-1



소켓을 이용한 서버 클라이언트 통신 프로그램의 구조-2



1. 자바 소켓 클래스

❖ ServerSocket 생성과 연결 수락

- ServerSocket 생성과 포트 바인딩
  - 생성자에 바인딩 포트 대입하고 객체 생성

- 연결 수락
  - accept() 메소드는 클라이언트가 연결 요청 전까지 블로킹 → 대기
  - 연결된 클라이언트 IP 주소 얻기

```
InetSocketAddress socketAddress = (InetSocketAddress) socket.getRemoteSocketAddress();
```

리터타입	메소드명(매개변수)	설명
String	getHostName()	클라이언트 IP 리턴
int	getPort()	클라이언트 포트 번호 리턴
String	toString()	"IP:포트번호" 형태의 문자열 리턴

- ServerSocket 포트 언바인딩
  - 더 이상 클라이언트 연결 수락 필요 없는 경우

ServerSocket 클래스, 서버 소켓

- **ServerSocket 클래스**
  - 서버 소켓에 사용되는 클래스
  - java.net 패키지에 포함
  - 주요 생성자

생성자	설명
ServerSocket()	소켓을 생성, bind 함수에 의해 로컬주소와 포트번호에 연결
ServerSocket(int port)	소켓을 생성하여 지정된 포트 번호에 연결.

  - 주요 메소드

메소드	설명
Socket accept()	Client 연결 요청을 기다리다 요청이 들어오면 수락하고 새 Socket 객체를 반환
void close()	서버 소켓을 닫는다.
InetAddress getInetAddress()	서버 소켓에 연결된 로컬 주소 반환
int getLocalPort()	서버 소켓이 연결 요청을 모니터링 하는 포트 번호 반환
boolean isBound()	서버 소켓이 로컬 주소에 연결되어있으면 true 반환
boolean isClosed()	서버 소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	accept()에 대한 타임 아웃 시간 지정. 0이면 타임아웃이 해제.

Socket 클래스, 클라이언트 소켓

Socket 클래스

- 클라이언트 소켓에 사용되는 클래스
- java.net 패키지에 포함
- 주요 생성자

생성자	설명
Socket()	소켓을 생성, connect() 메소드에 의해 서버와 연결
Socket(InetAddress address, int port)	소켓을 생성하여 지정된 IP 주소와 포트 번호에 연결.
Socket(String host, int port)	소켓을 생성하여 지정된 호스트와 포트 번호에 연결, 호스트 이름이 null인 경우는 루프백(loopback) 주소로 가정.

Socket 클래스, 클라이언트 소켓

- 주요 메소드

메소드	설명
void close()	소켓을 닫는다.
void connect(SocketAddress endpoint)	소켓을 서버에 연결
InetAddress getInetAddress()	소켓이 연결한 서버의 주소 반환
InputStream getInputStream()	소켓에 대한 입력 스트림 반환
InetAddress getLocalAddress()	소켓이 연결된 로컬 주소 반환
int getLocalPort()	소켓이 연결된 로컬 포트 번호 반환
int getPort()	소켓이 연결한 서버의 포트 번호 반환
OutputStream getOutputStream()	소켓에 대한 출력 스트림 반환
boolean isBound()	소켓이 로컬 주소에 연결되어있으면 true 반환
boolean isConnected()	소켓이 서버에 연결되어 있으면 true 반환
boolean isClosed()	소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	데이터 읽기 타임아웃 시간 지정. 0이면 타임아웃 해제.

소켓 생성, 서버 접속, 입출력 스트림 생성

- 클라이언트 소켓 생성 및 서버에 접속

```
Socket clientSocket = new Socket("128.12.1.1", 5550);
```

  - Socket 객체의 생성되면 곧 바로 128.12.1.1의 주소로 자동 접속
- 네트워크 입출력 스트림 생성

```
BufferedReader in = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));
BufferedWriter out = new BufferedWriter(
                        new OutputStreamWriter(clientSocket.getOutputStream()));
```

  - 일반 스트림을 입출력 하는 방식과 동일
- 서버로 데이터 전송
  - flush()를 호출하면 스트림 속에 데이터를 남기지 않고 모두 전송

```
out.write("hello"+"₩n");
out.flush();
```

- 서버로부터 데이터 수신

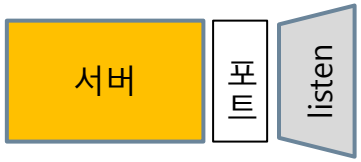
```
int x = in.read();          // 서버로부터 한 개의 문자 수신
String line = in.readline(); //서버로부터 한 행의 문자열 수신
```

- 네트워크 접속 종료

```
clientSocket.close();
```

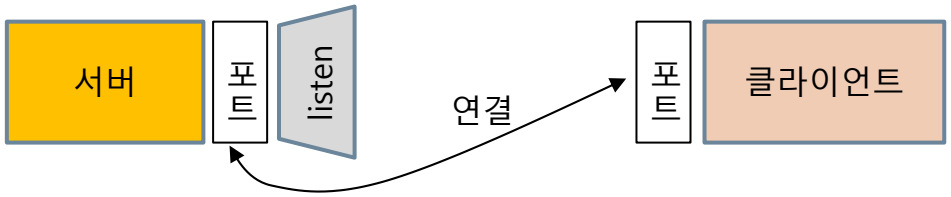
클라이언트와 서버 연결 순서

- **클라이언트와 서버 연결**
  - 서버는 서버 소켓으로 들어오는 연결 요청을 기다림



  - 클라이언트가 서버에게 연결 요청

  - 서버가 연결 요청 수락하고 새로운 소켓을 만들어 클라이언트와 연결 생성



1. 자바 소켓 클래스

❖ Socket 생성과 연결 요청

- Socket 생성 및 연결 요청
  - java.net.Socket 이용
  - 서버의 IP 주소와 바인딩 포트 번호를 제공하면 생성과 동시에 사용가능

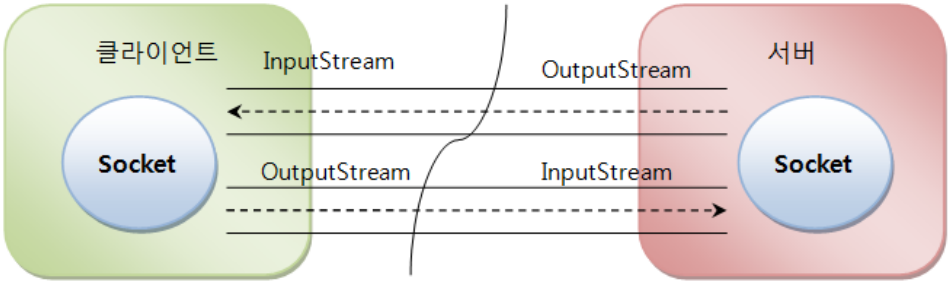
- 연결 끊기
  - Exception 처리 필요

- 1061 페이지 예제를 통해 생성 → 연결 → 끊기 이해

1. 자바 소켓 클래스

❖ Socket 데이터 통신

- Socket 객체로 부터 입출력 스트림 얻기



- 입출력 스트림 구현 예제 (p.1063~1066)
  - 연결 성공 후 클라이언트가 서버에 "Hello Server"
  - 서버가 데이터 받음
  - 서버가 클라이언트에 "Hello Client" 보냄
  - 클라이언트가 데이터 받음
- read()의 블로킹 해제

블로킹이 해제되는 경우	리턴값
상대방이 데이터를 보냄	읽은 바이트 수
상대방이 정상적으로 Socket 의 close()를 호출	-1
상대방이 비정상적으로 종료	IOException 발생

1. 자바 소켓 클래스

- **클라이언트/서버 프로그램 작성**
  - 바이트 스트림을 문자 스트림으로 바꾸는 방법
    - InputStream 객체를 InputStreamReader 객체로 변경

```
InputStreamReader reader = new InputStreamReader(in);
```

InputStream 객체

    - OutputStream 객체를 PrintWriter 객체로 변경

```
PrintWriter writer = new PrintWriter(out);
```

OutputStream 객체

2. TCP 네트워킹

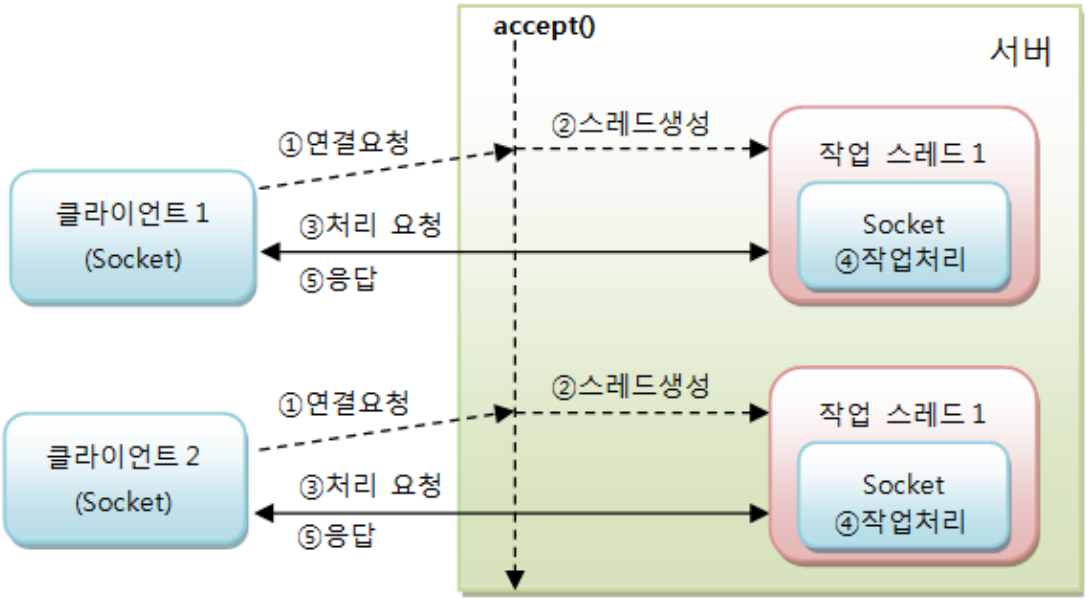
❖ 스레드 병렬 처리

- 블로킹(대기 상태)가 되는 메소드
  - ServerSocket의 accept()
  - Socket 생성자 또는 connect()
  - Socket의 read(), write()

- 병렬 처리의 필요성
  - 스레드가 블로킹되면 다른 작업을 수행하지 못한다.
    - 입출력 할 동안 다른 클라이언트의 연결 요청 수락 불가
    - 입출력 할 동안 다른 클라이언트의 입출력 불가
  - UI 생성/변경 스레드에서 블로킹 메소드를 호출하지 않도록
    - UI 생성 및 변경이 안되고 이벤트 처리 불가

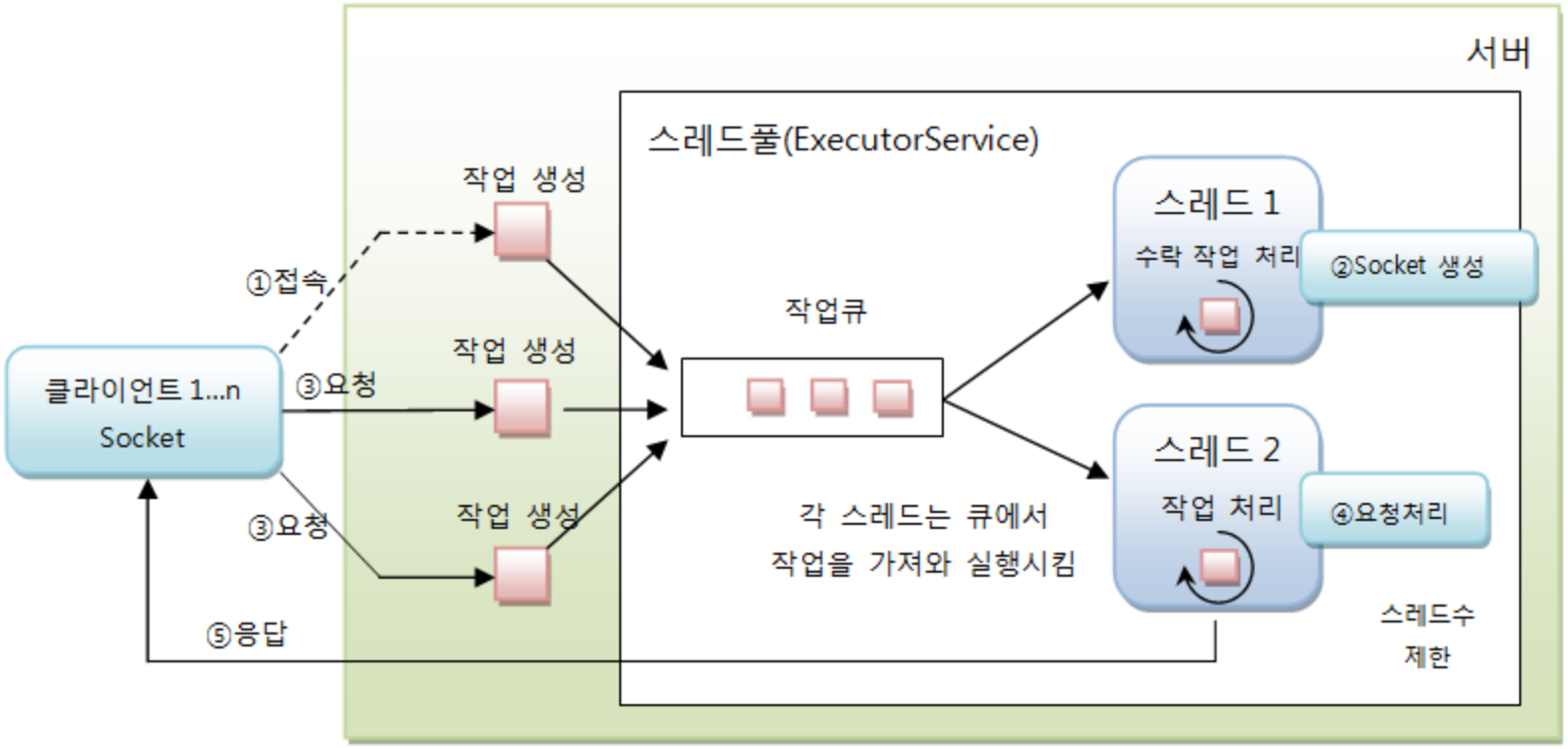
2. TCP 네트워킹

- 스레드 병렬 처리
  - Accept(), connect(), read(), write()는 별도 작업 스레드 생성



2. TCP 네트워킹

- 스레드풀 사용해 스레드 수 관리

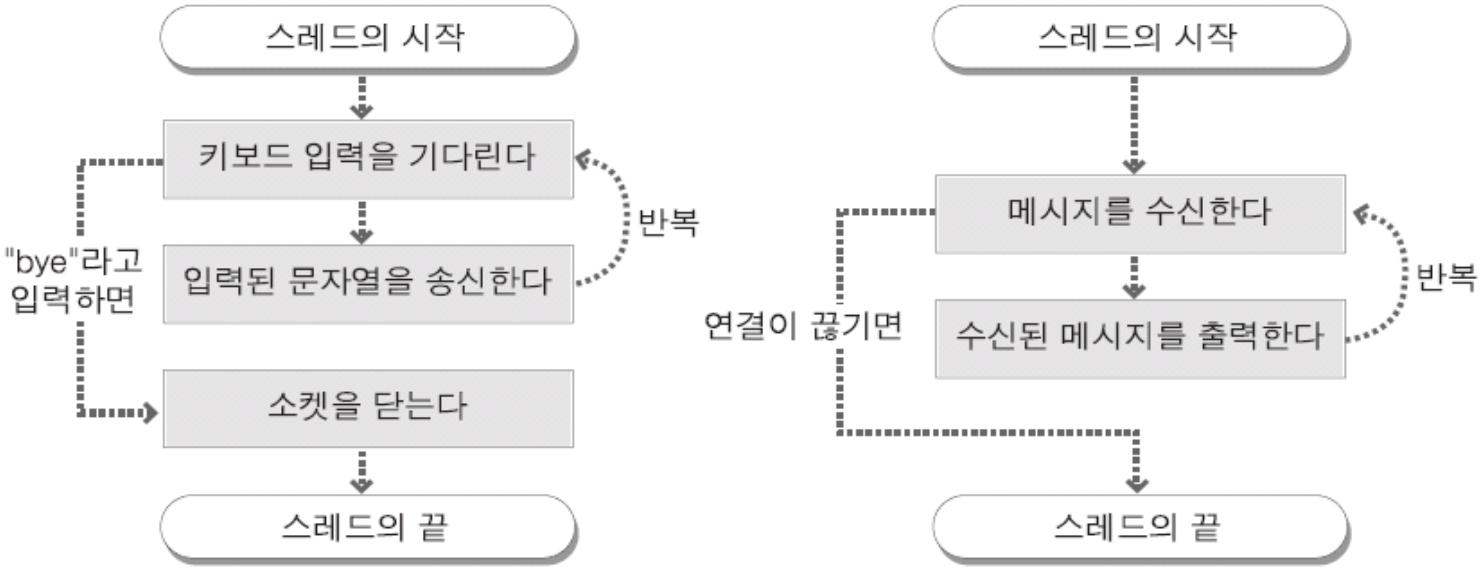


- 스레드풀은 스레드 수 제한해 사용
- 갑작스런 클라이언트의 폭증은 작업 큐의 작업량만 증가
  - 서버 성능은 완만히 저하
  - 대기하는 작업량 많아 개별 클라이언트에서 응답을 늦게 받기도

2. TCP 네트워킹

❖ 스레드 병렬 처리 1- 송신과 수신을 동시에 하는 프로그램

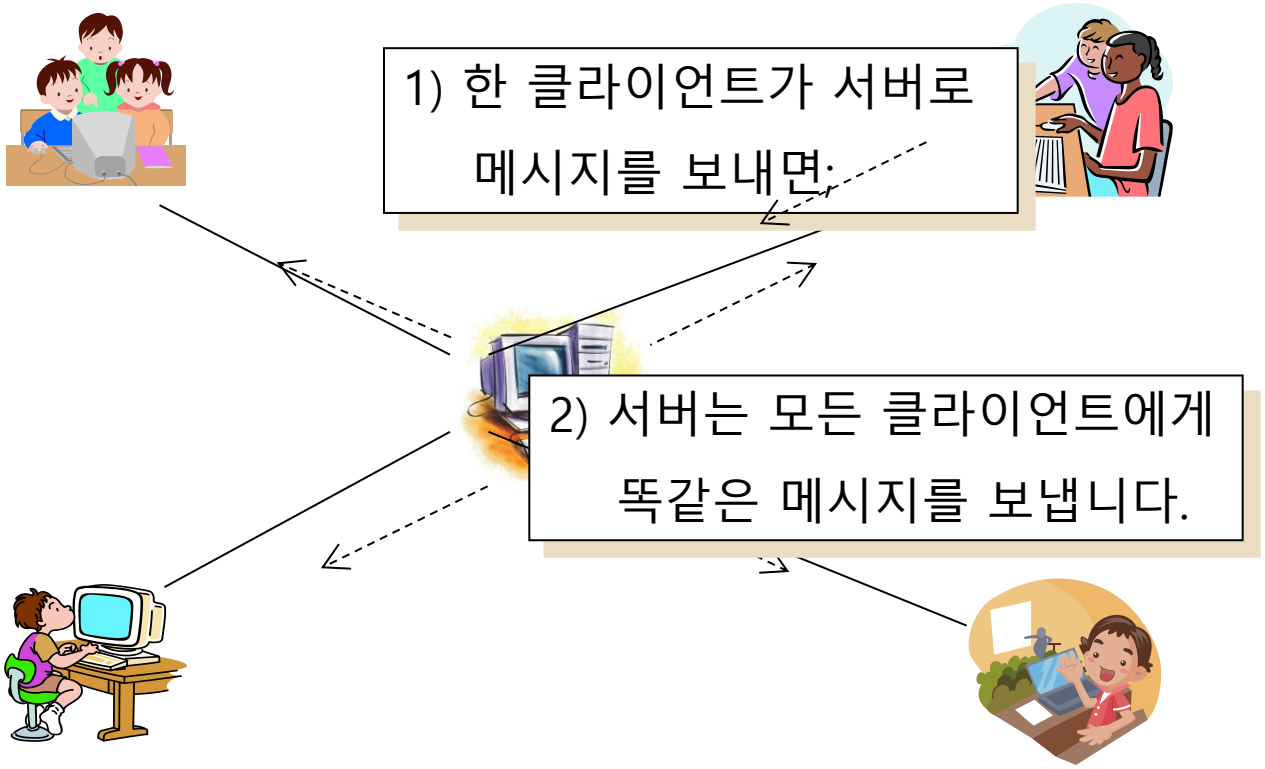
- 클라이언트 프로그램과 서버 프로그램의 실행 흐름
- 1:1 채팅



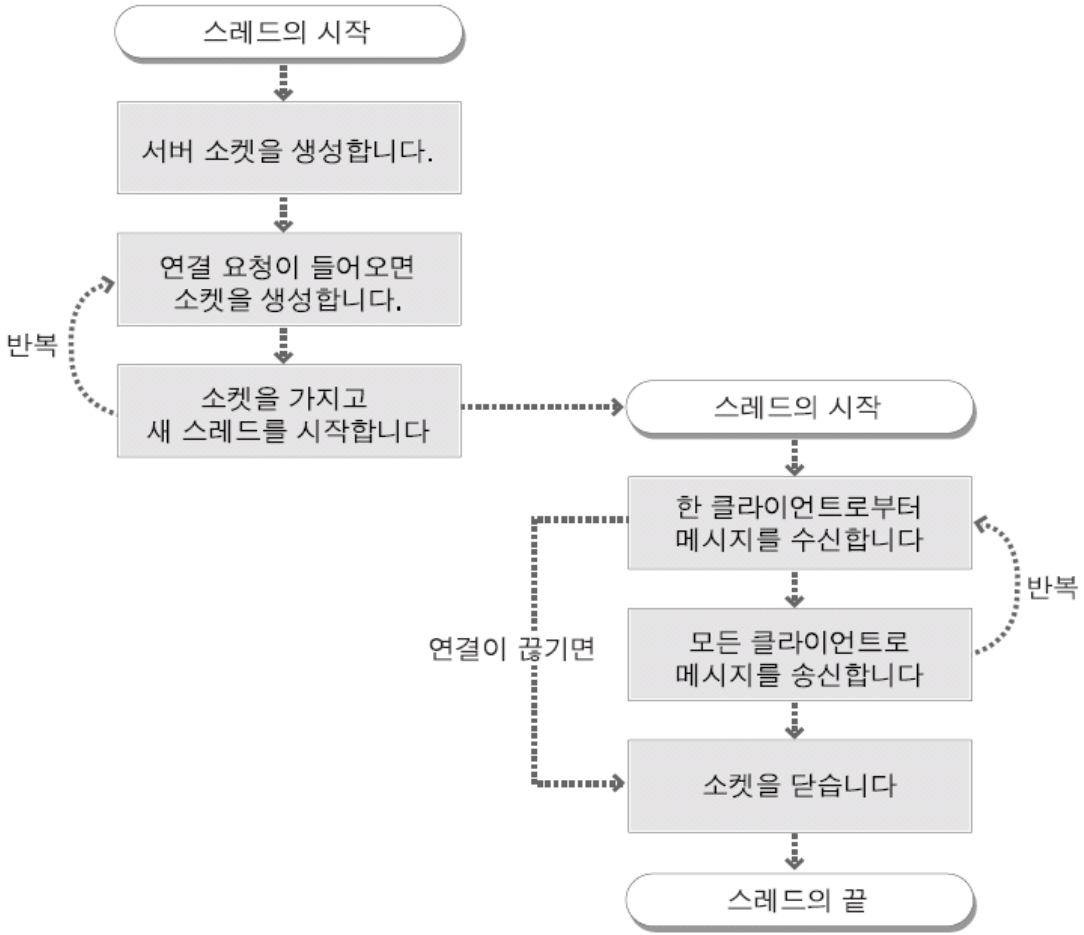
2. TCP 네트워킹

❖ 다중 사용자 채팅 프로그램

- 일반적인 채팅 프로그램의 작동 방식



❖ 여러 명이 참여하는 채팅 프로그램의 실행 흐름



2. TCP 네트워킹

- 스레드 병렬 처리-3
  - 스레크 풀 사용한 채팅 서버 및 클라이언트 구현
    - 서버
      - startServer ()
        - Executor Service , 서버소켓 생성, 포트 바인딩, 연결수락 코드
      - stopServer()
        - 연결된 모든 소켓, 서버소켓 닫기, Executor Service 종료
      - 클라이언트 클래스
        - 다수 클라이언트 관리 → 각자 클라이언트 인스턴스 생성해 관리
      - UI 생성 코드 (java Swing 이용한 UI 생성 코드)
    - 클라이언트
      - Startclient() – 소켓 생성 및 연결요청 코드
      - Stopclient() – 소켓 통신 닫는 기능도 포함
      - Receive () – 서버에서 보낸 데이터 받음
      - Send(String data) - 사용자가 보낸 메시지 서버로 보냄
      - UI 생성 코드

3. UDP 네트워킹

UDP(User Datagram Protocol)

- 특징
  - 비연결 지향적 프로토콜
    - 연결 절차 거치지 않고 발신자가 일방적으로 데이터 발신하는 방식
    - TCP 보다는 빠른 전송
  - 통신 선로가 고정적이지 않음
    - 데이터 패킷들이 서로 다른 통신 선로 통해 전달될 수 있음
    - 먼저 보낸 패킷이 느린 선로 통해 전송될 경우, 나중에 보낸 패킷보다 늦게 도착 가능
  - 데이터 손실 발생 가능성
    - 일부 패킷은 잘못된 선로로 전송되어 유실 가능
    - 데이터 전달 신뢰성 떨어짐

3. UDP 네트워킹

- java.net API
  - DatagramSocket, DatagramPacket

- UDP 네트워킹 구현 예제
  - 발신자 구현 코드 – 소켓 통해 데이터 패킷 전송
  - 수신자 구현 코드 – 바인딩한 특정 포트로 데이터 받아 저장
  - DatagramSocket 닫기