Software Architecture

Distributed Architecture

Types of Distributed Architecture Style

- Client-server
- Multitier
- Proxy
- Dispatcher (Load Balancer)
- P2P
- Broker
- Service-oriented architecture
- Microservice architecture

Client-Server Architecture Style

Eunmi Choi
Kookmin University

Client-Server Architecture Style

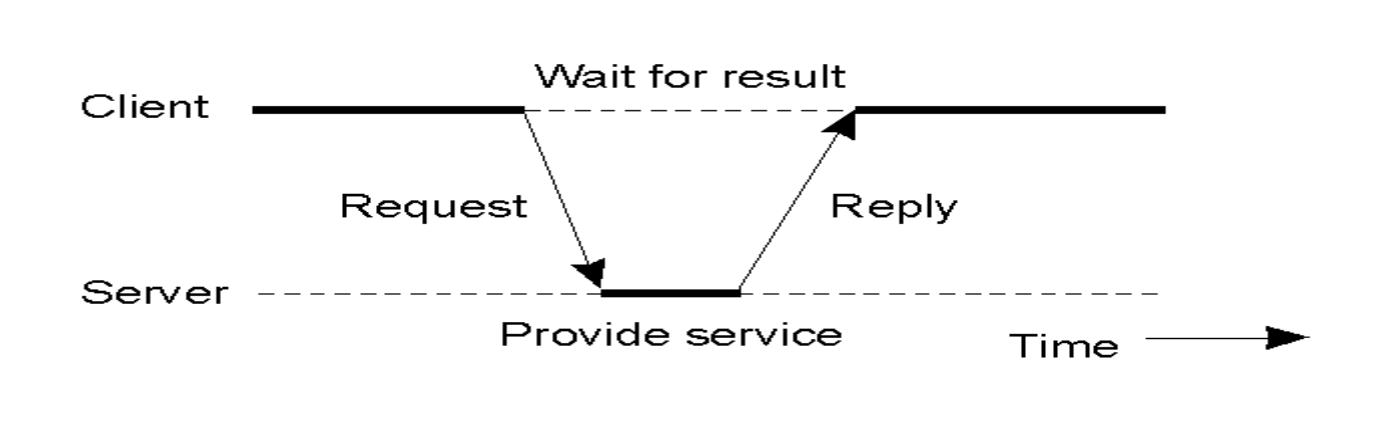
클라이언트 프로세스와 서버 프로세스가 있고 커뮤니케이션 할 수 있도록 함.
서버쪽은 리퀘스트 받을 준비가 되어 있고 그에 따른 프로세싱을 하여 그 결과를 클라이언트에게 보냄 (서비스 한다). 처리자.
클라이언트 서버에게 리퀘스트를 보내는 요청자.

- Synapsis
  - The most commonly distributed system architecture
  - Two communicating processes, usually running on different processors
- Structure
  - Decomposes a system into two major subsystems:
    - client and server.
    - The first process, the client
      - issues a request to the second process, the server.
    - The server
      - process receives the request (serving data from a database, printing a document), carries it out, and sends a reply to the client.

Client-Server Architecture Style

- Processes in DS are divided into two groups
  - Server: a process implementing a specific service
  - Client: a process that requests a service from a server by sending it a request and subsequently waiting for the server's reply

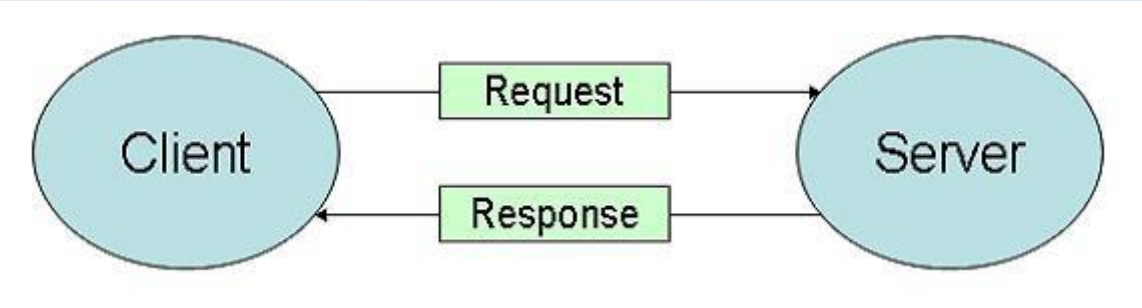
- Request-reply interaction
  - General interaction between a client and a server.



Client-Server Architecture Style

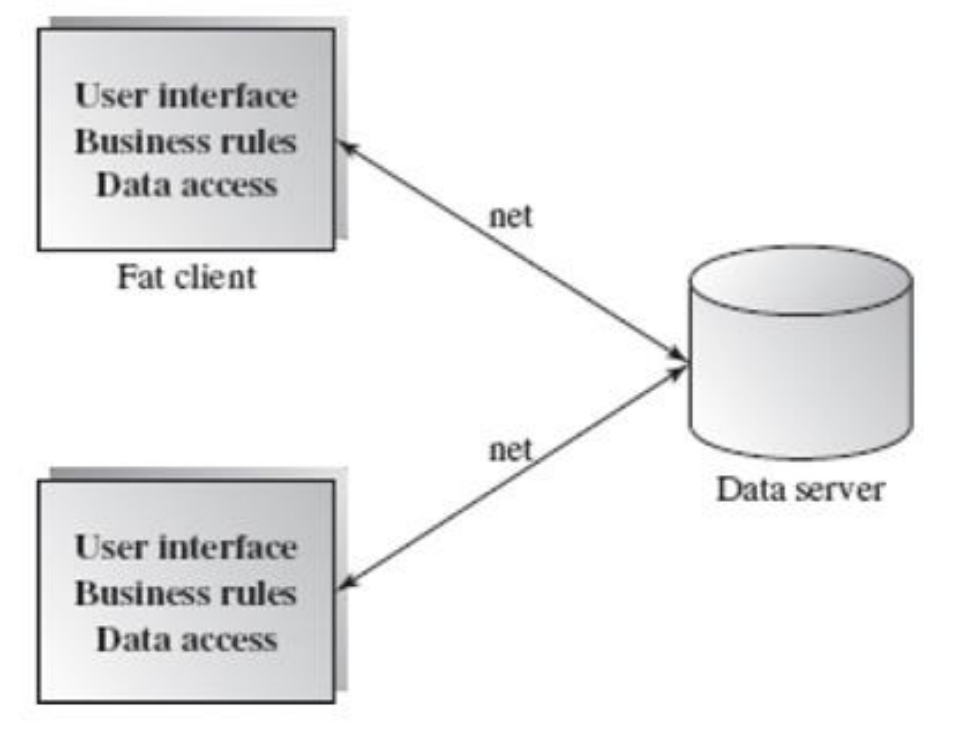
- *Client/server* describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server.

둘의 일은 자명하게 분리된다.



Client-Server Architecture Style

- Two-tier client-server architecture



Client-Server Architecture Style

- Advantages:
  - Separation of responsibilities such as user interface presentation and business logic processing
  - Reusability of server components 어떤 상황에 사용하는지에 따라 서버의 컴포넌트는 재사용 가능
- Disadvantages:
  - Lack of heterogeneous infrastructure to deal with the requirement changes 서로 요청하는 경우가 다른 경우 이것은 고민해야. 이종 시스템에서는 프로토콜을 맞추기 힘듦. 표준화를 잘 써야함. 표준화를 안쓰면 동종이어야 함
  - Security complications 세큐리티가 복잡하다.
  - Server availability and reliability 서버가 항상 가용해야 함. 유저빌리티(쓰기 편한 것)
    신뢰성.
  - Testability and scalability 측정 가능하고
    서버가 여러개 붙어 있으면 스캐일어빌리티
  - Fat clients with presentation and business logic together 클라이언트와 서버가 하는 일을 잘 구분해야 함.