Client Dispatcher Server 패턴

아꿈사 http://cafe.naver.com/architect1
TTF http://www.npteam.net

CDS 패턴의 3가지 컴포넌트

Client Component
  - 클라이언트 컴포넌트

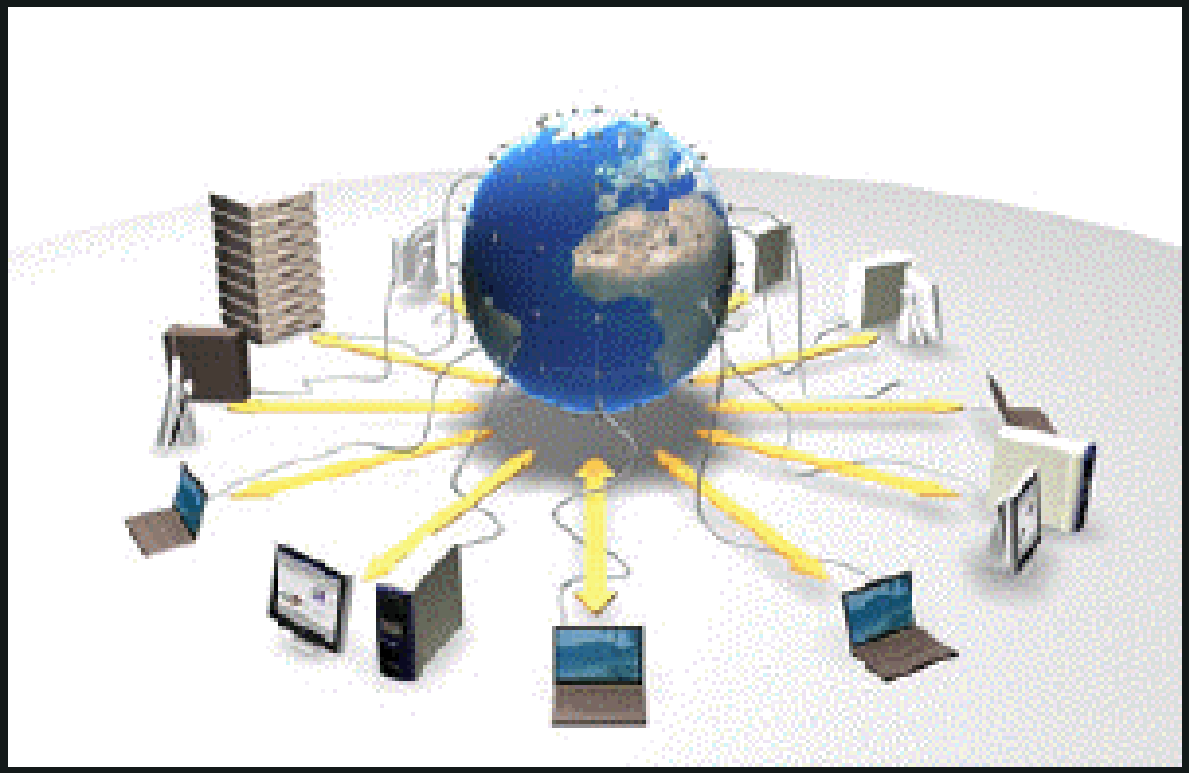
Dispatcher Component
  - 디스패처 컴포넌트

Server Component
  - 서버 컴포넌트

클라이언트와 서버간에 디스패처 컴포넌트를 도입하여
위치투명성을 제공하고,
클라이언트←→서버간 통신을 위한 세부 구현을 숨긴다.

CDS Pattern - Context

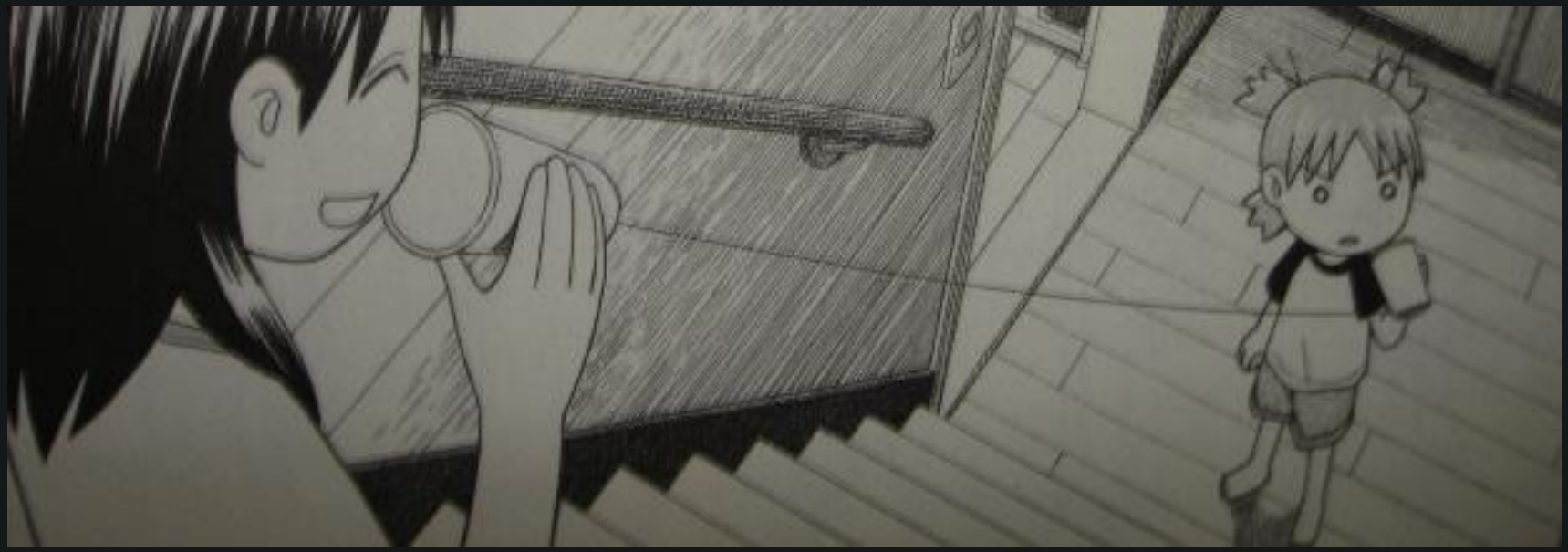
- 정황(Context)
  - 로컬 혹은 네트워크에서 실행되는 분산된 서버들을 통합해야 한다.



이미지 출처 : http://data.ygosu.com/editor/attach/20100812/QSycLPT8dyvQG.gif

CDS Pattern - Problem

- 문제(Problem)
  - 분산된 서버간 통신할 방법을 제공해야 한다.
  - 핵심 기능과 세부 구현은 분리되어야 한다.



이미지 출처 : http://www.comixest.com/bbs/view.php?id=news&type=all&r_page=1&key=&no=1396

CDS Pattern - Problem

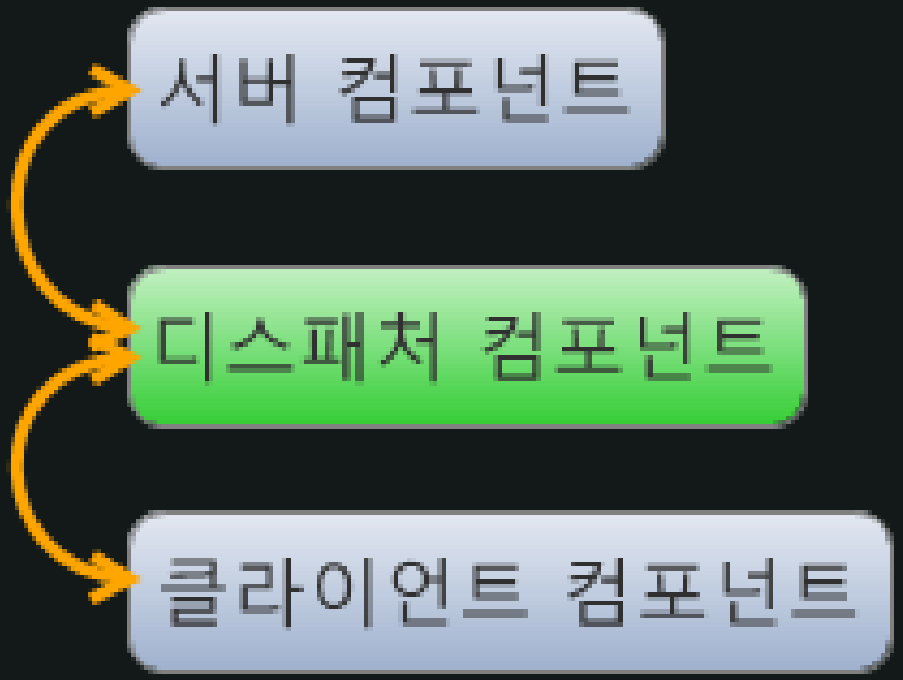
- 문제(Problem)
  - 클라이언트는 서버의 위치를 알 필요가 없다.
  - 서버 위치 변경 및 장애 허용성을 제공한다.

CDS Pattern – Problem(Force)

- 문제에 내포된 영향력(Force)
  - 컴포넌트는 서비스 제공자의 **위치**와 상관없이 서비스를 사용할 수 있어야 한다.
  - 서비스 소비자의 **핵심 기능**을 구현하는 코드는 서비스 제공자와 **연결 설정**을 위한 코드와 **분리** 되어야 한다.

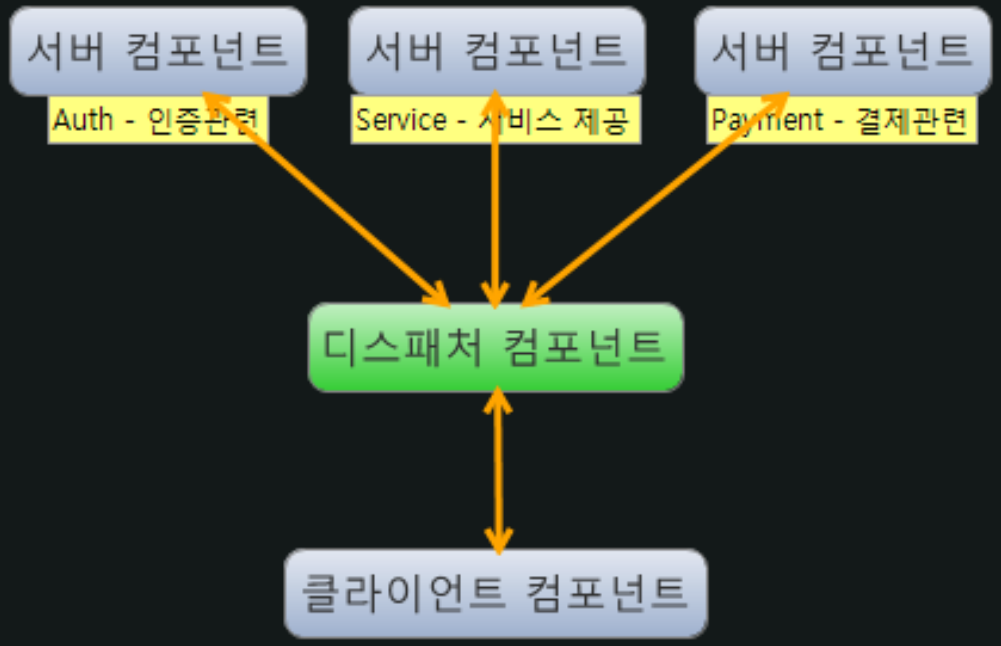
CDS Pattern – Solution

- 해법(Solution)
  - 클라이언트와 서버 간의 중간 레이어 역할을 하는 디스패처 컴포넌트를 제공한다.



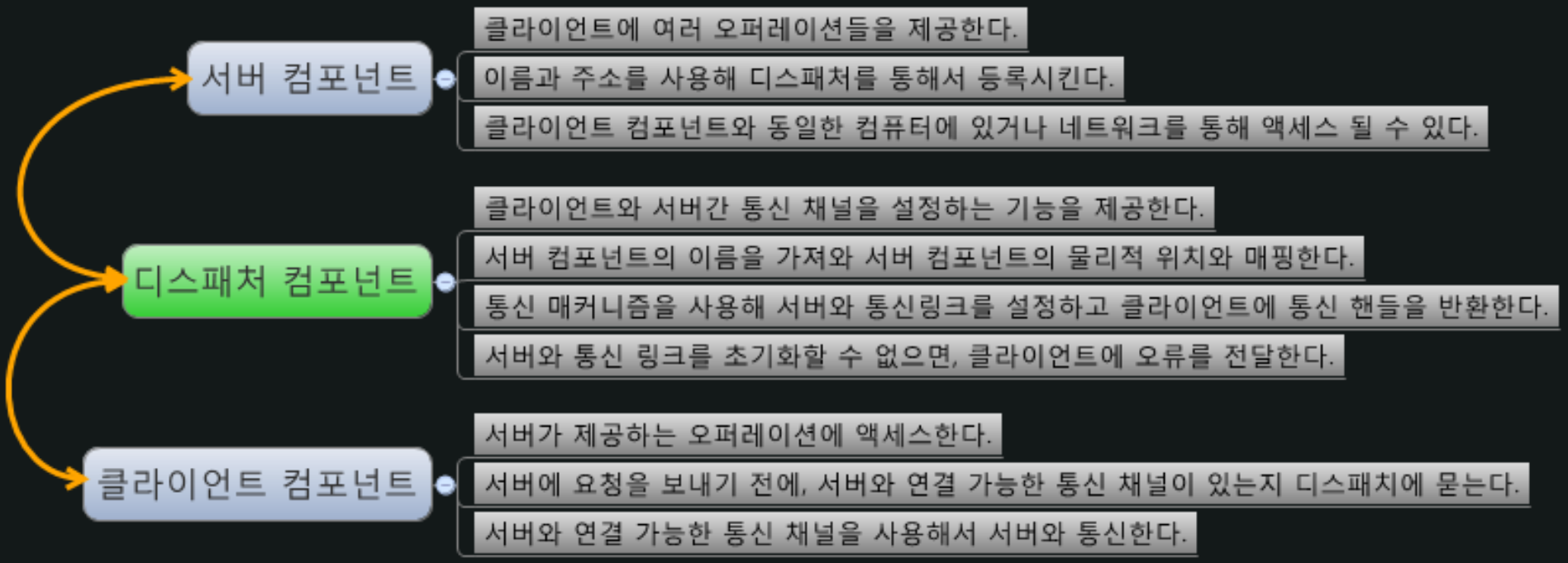
CDS Pattern – Solution

- 해법(Solution)
  - 디스패처 컴포넌트는 네임 서비스를 구현한다.
  - 네임서비스를 제공하여 위치 투명성을 제공한다.
  - 디스패처는 클라이언트와 서버간 통신 채널을 설정하는 책임을 맡는다.



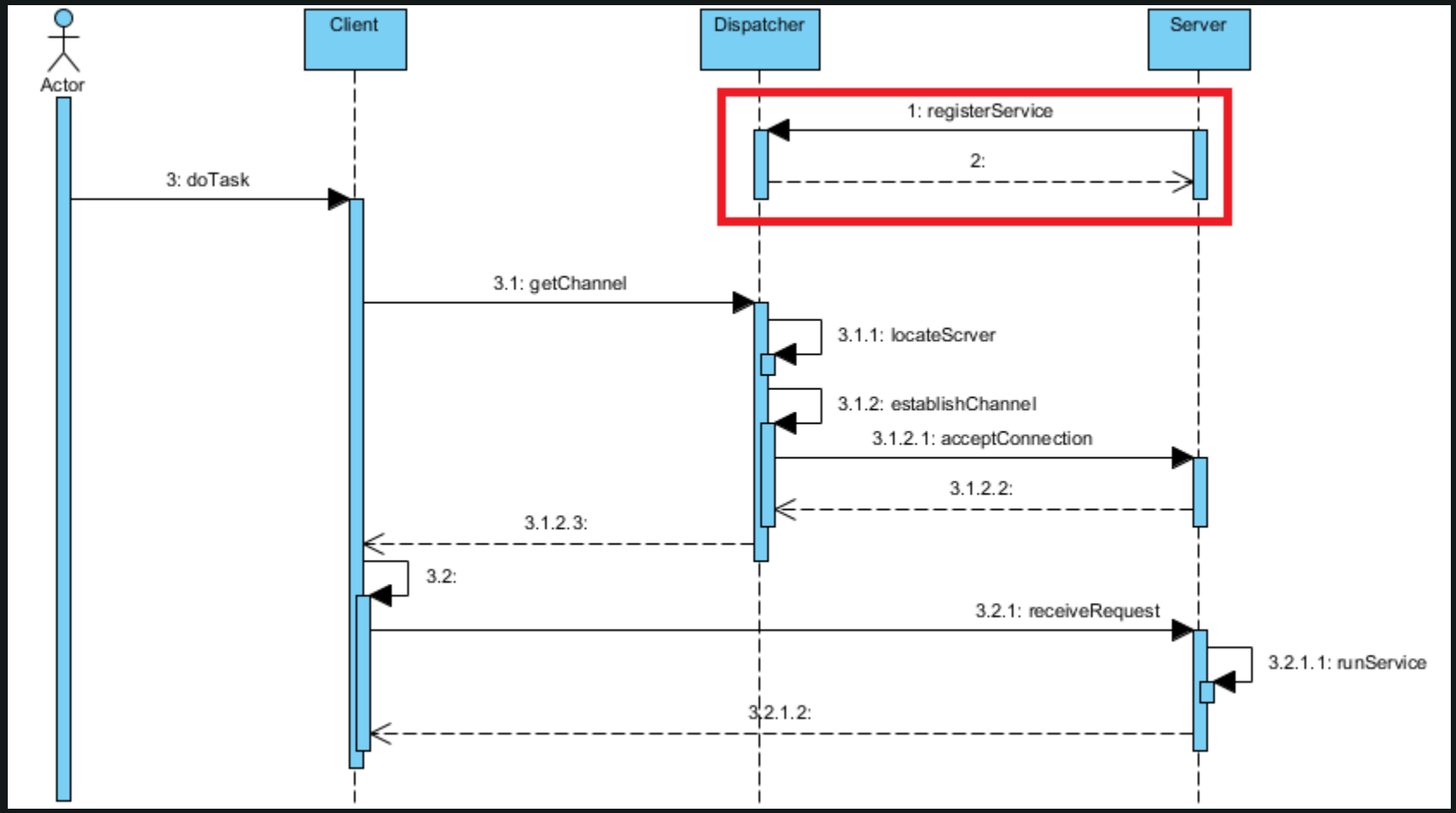
CDS Pattern – Structure

- 구조(Structure)
  - 각 컴포넌트는 다음과 같이 구성된다.



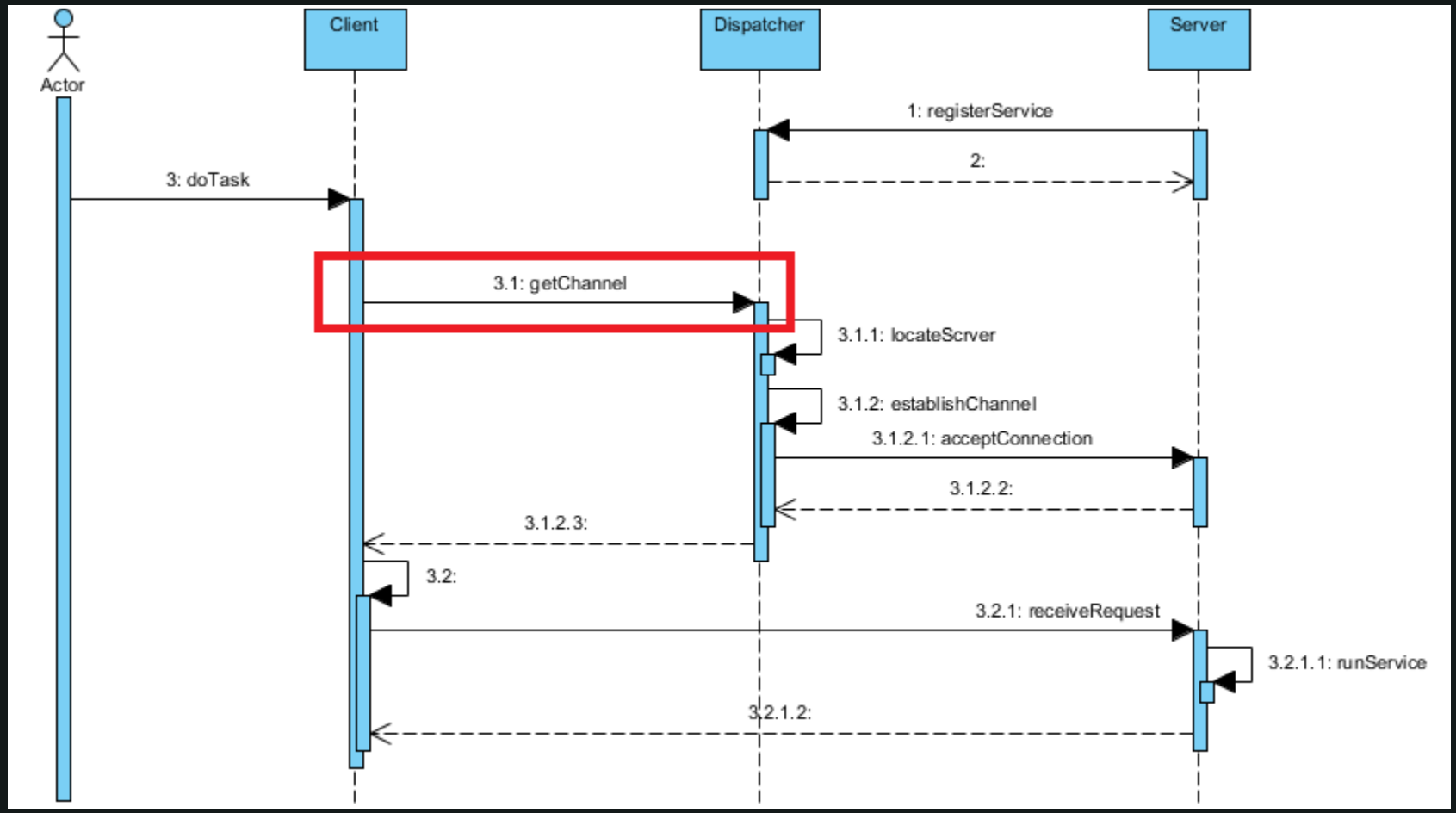
CDS Pattern – Dynamic

- CDS 동작 묘사
  - 서버는 디스패처 컴포넌트에 자신을 등록시킨다.



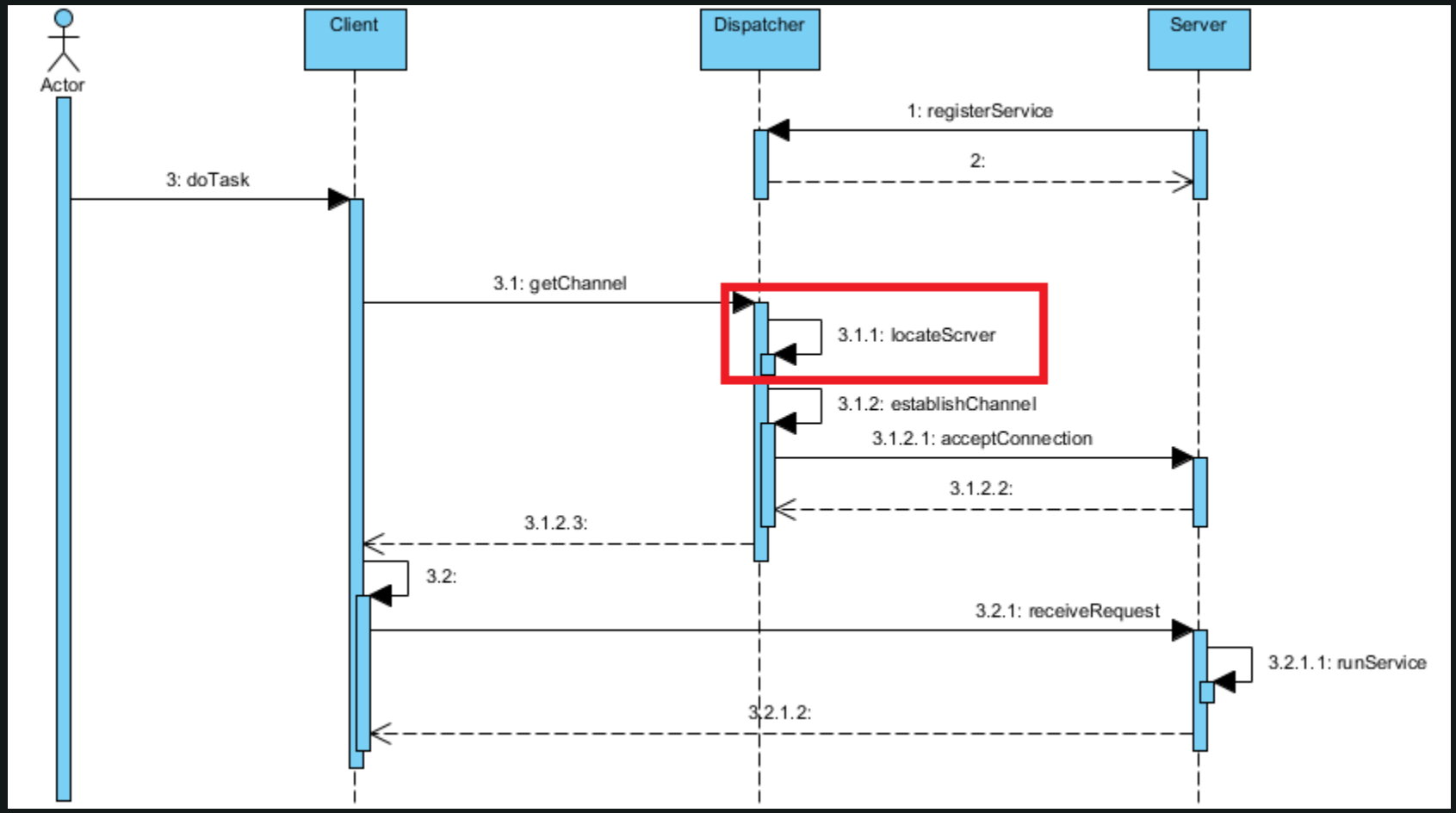
CDS Pattern – Dynamic

- CDS 동작 묘사
  - 클라이언트는 지정된 서버에 통신 채널을 설정할 수 있는지 디스패처에 묻는다.



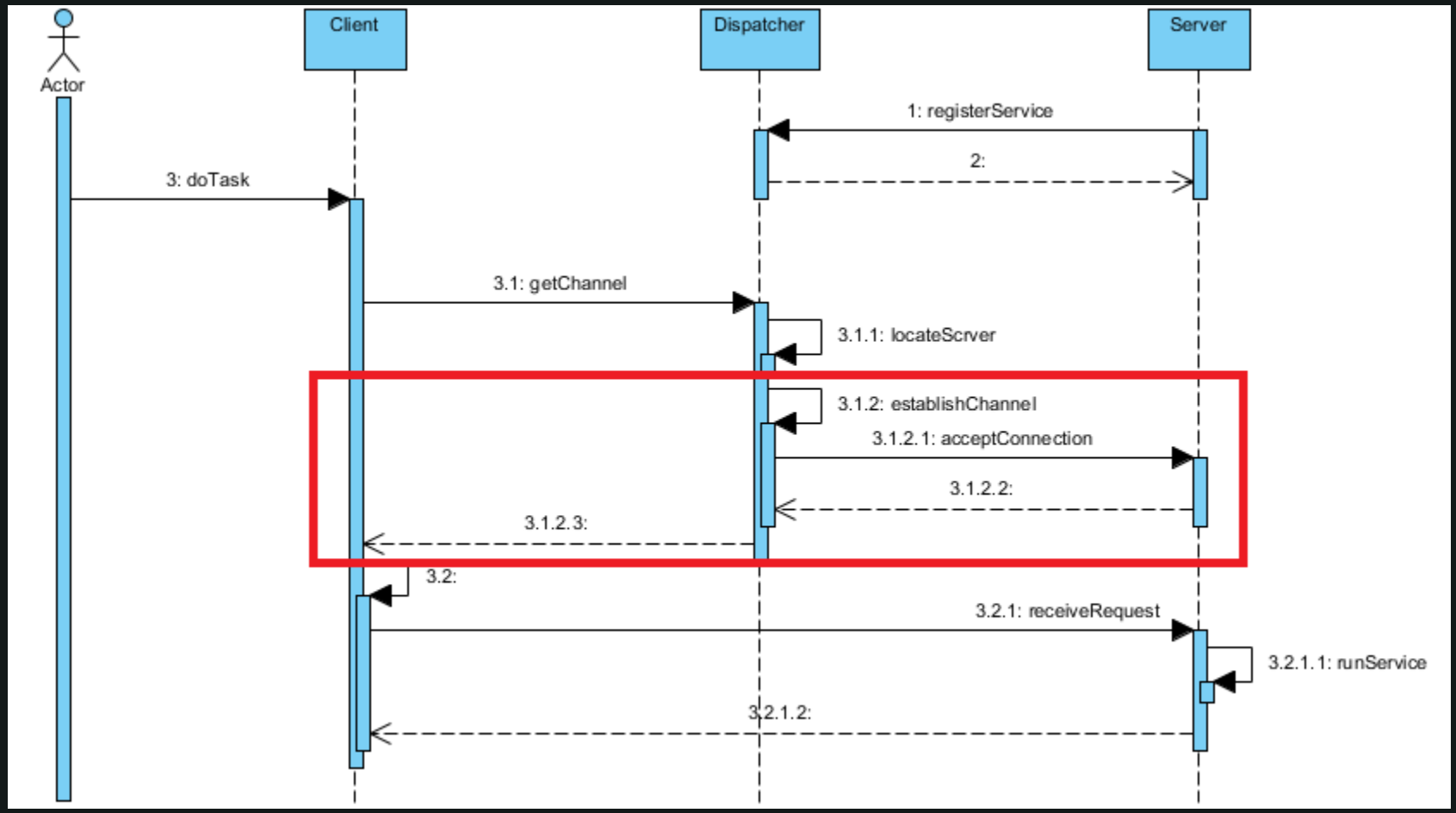
CDS Pattern – Dynamic

- CDS 동작 묘사
  - 디스패처는 자체 레지스트리에서 클라이언트가 지정한 이름과 연결된 서버를 검색한다.



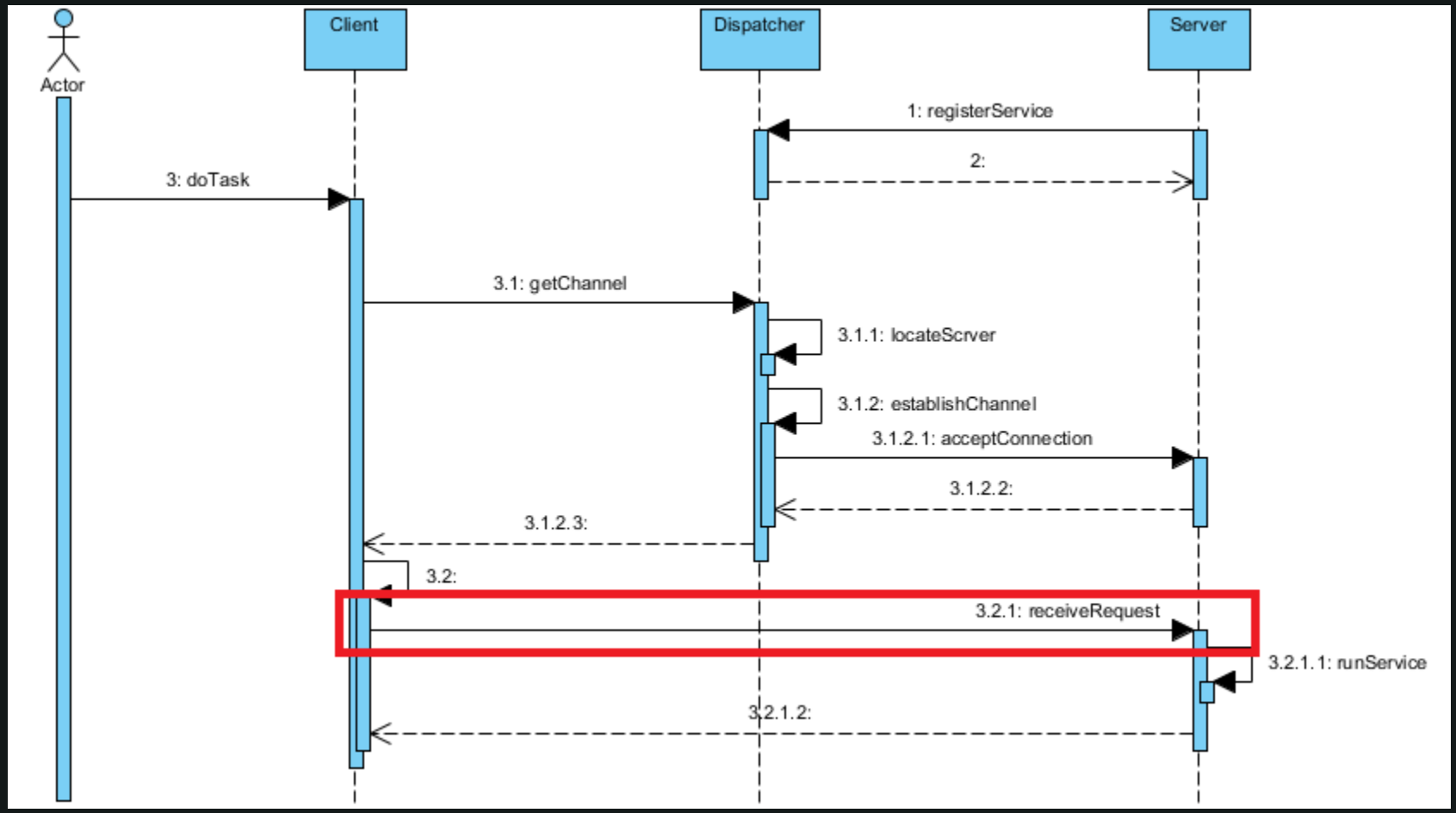
CDS Pattern – Dynamic

- CDS 동작 묘사
  - 디스패처는 서버와 통신 링크를 설정한다.
  - 링크 성공시 채널을 반환, 실패하면 오류를 반환한다.



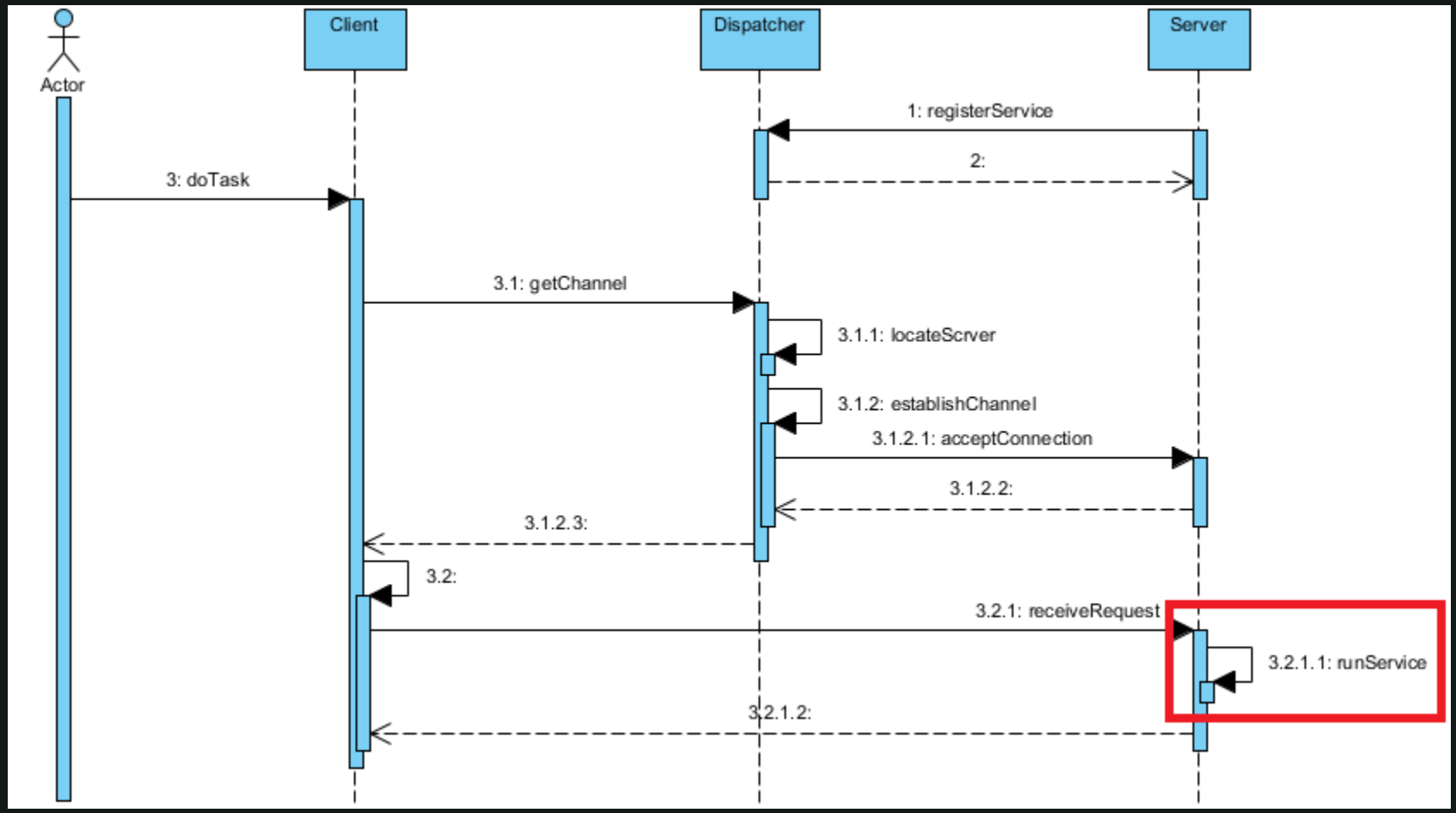
CDS Pattern – Dynamic

- CDS 동작 묘사
  - 클라이언트는 반환된 통신 채널을 사용해 서버에 직접 요청을 보낸다.



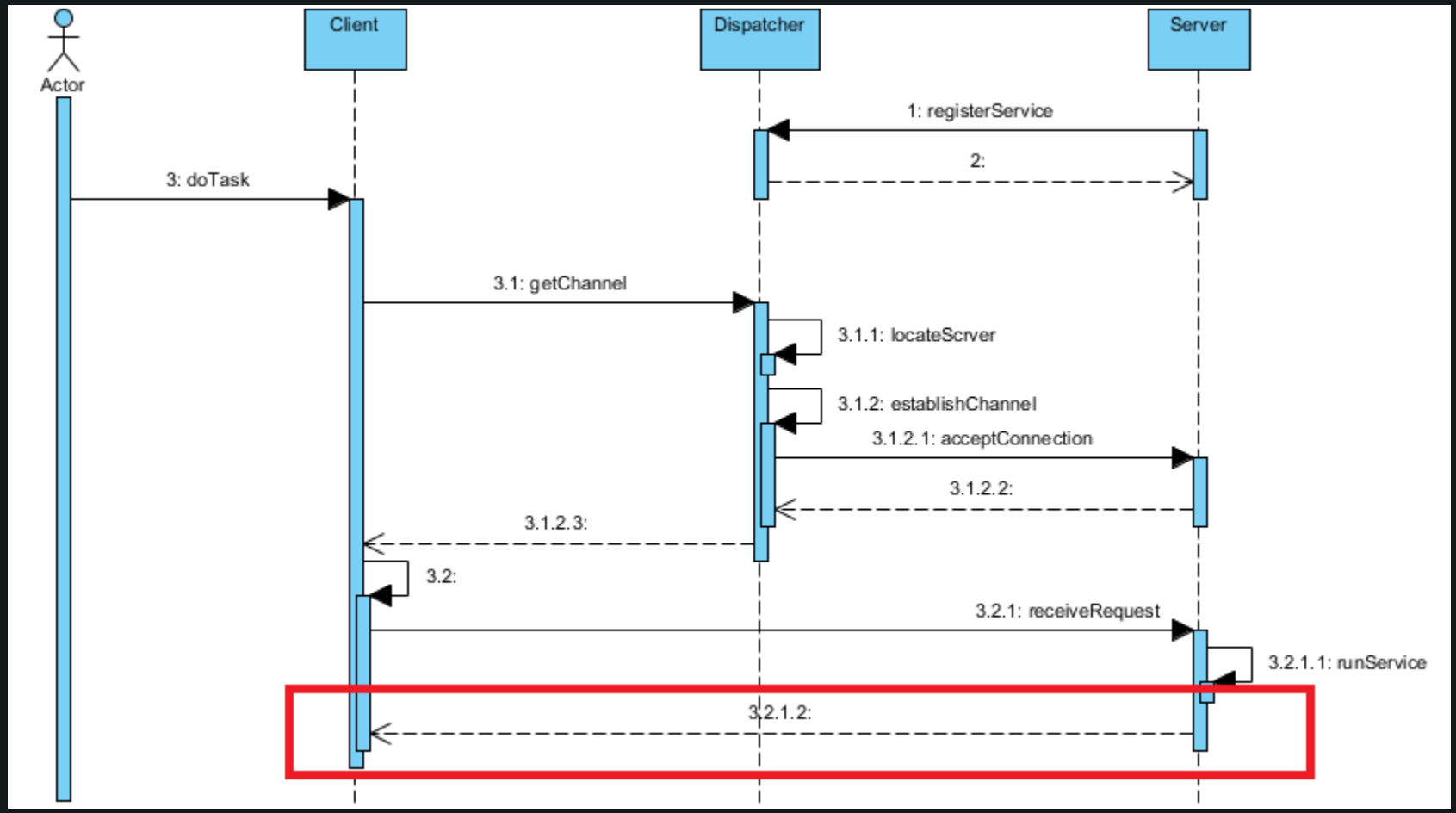
CDS Pattern – Dynamic

- CDS 동작 묘사
  - 서버는 들어온 요청을 받아들인 다음, 적절한 서비스를 실행한다.



CDS Pattern – Dynamic

- CDS 동작 묘사
  - 서버스 실행이 완료되면, 서버는 클라이언트에 결과를 보낸다.



CDS Pattern – Implementation

- 구현(Implementation)

  - **1단계** 애플리케이션을 서버 및 클라이언트로부터 분리한다.

    → 서버 및 클라이언트 컴포넌트를 분리하여 설계한다.

    → 역할이 분리되어야 나중에 유연하게 통합할 수 있다.

  - **2단계** 어떤 통신 기능이 필요한지 파악한다.

    → 동일 머신에서는 공유메모리가 가장 빠르다.

    → 클라이언트 서버간 통신은 소켓을 사용하는 것이 가장 적당하다.

CDS Pattern – Implementation

- 구현(Implementation)

  - **3단계** 컴포넌트 간의 상호작용 프로토콜을 정의한다.

    → 서버와 디스패처 간에는 DSprotocol이 필요하다.
    - 서버가 어떻게 디스패처에 등록되는지 정의한다.
    - 서버에 통신 채널을 설정할 때 필요한 동작을 결정한다.

    → 클라이언트와 디스패처 간에는 CDprotocol이 필요하다.

  - **4단계** 서버의 이름을 어떻게 지을 것인지 결정한다.

    → IP 주소는 위치 투명성을 지원하지 않는다.

    → 고유하게 식별 가능한 이름을 도입해야 한다.

CDS Pattern – Implementation

- 구현(Implementation)

  - **5단계** 디스패처를 설계하고 구현한다.

    → 디스패처가 클라이언트 주소공간 내에 있을 경우 로컬 프로시저 사용

    → 그외의 경우 TCP 포트나 공유 메모리 같은 기능을 사용.

  - **6단계** 클라이언트 컴포넌트 및 서버 컴포넌트를 구현한다.

    → 계획한 해법과 디스패처 인터페이스에 맞게 시스템을 구성한다.

CDS Pattern – Variant

- 변형(Variant)

  - Distributed Dispatcher

    → 네트워크 환경에서 분산 디스패치 컴포넌트를 여러 개를 사용할 수 있다.

    → 원격 디스패처는 요청받은 서버와의 연결을 초기화한 다음 원래 디스패처에 통신 채널을 되돌려 보낸다.

    → 다른 방식은 클라이언트가 원격 디스패처와 직접 통신하는 방법이다.
    - 위치 투명성에 제약을 받는다.
    - 이런 경우엔 Broker 아키텍처 패턴을 먼저 검토한다.

CDS Pattern – Variant

- 변형(Variant)
  - Client-Dispatcher-Service 변형

    → 클라이언트가 서버가 아니라 서비스에 연결된다.

    → 디스패처가 요청을 받으면 자체 리포지토리 안에서 적절한 서비스를 검색한다.

    → 적절한 서비스를 찾으면 연결을 설정한다.

    → 연결 설정에 실패할 경우 동일한 다른 서비스에 연결을 시도한다.

CDS Pattern – Consequence

- 결과(Consequence)

  - 서버의 교환가능성이 보장된다.

  - 위치 투명성과 마이그레이션 투명성이 보장된다.

  - 재구성이 가능하다.

CDS Pattern – Consequence

- 결과(Consequence)

  - 장애 허용성이 보장된다.

  - 명시적이며 우회적인 연결을 설정하기 때문에 효율성이 낮다.

  - 디스패처 컴포넌트의 인터페이스에 발생하는 변경에 민감해진다.

감사합니다.