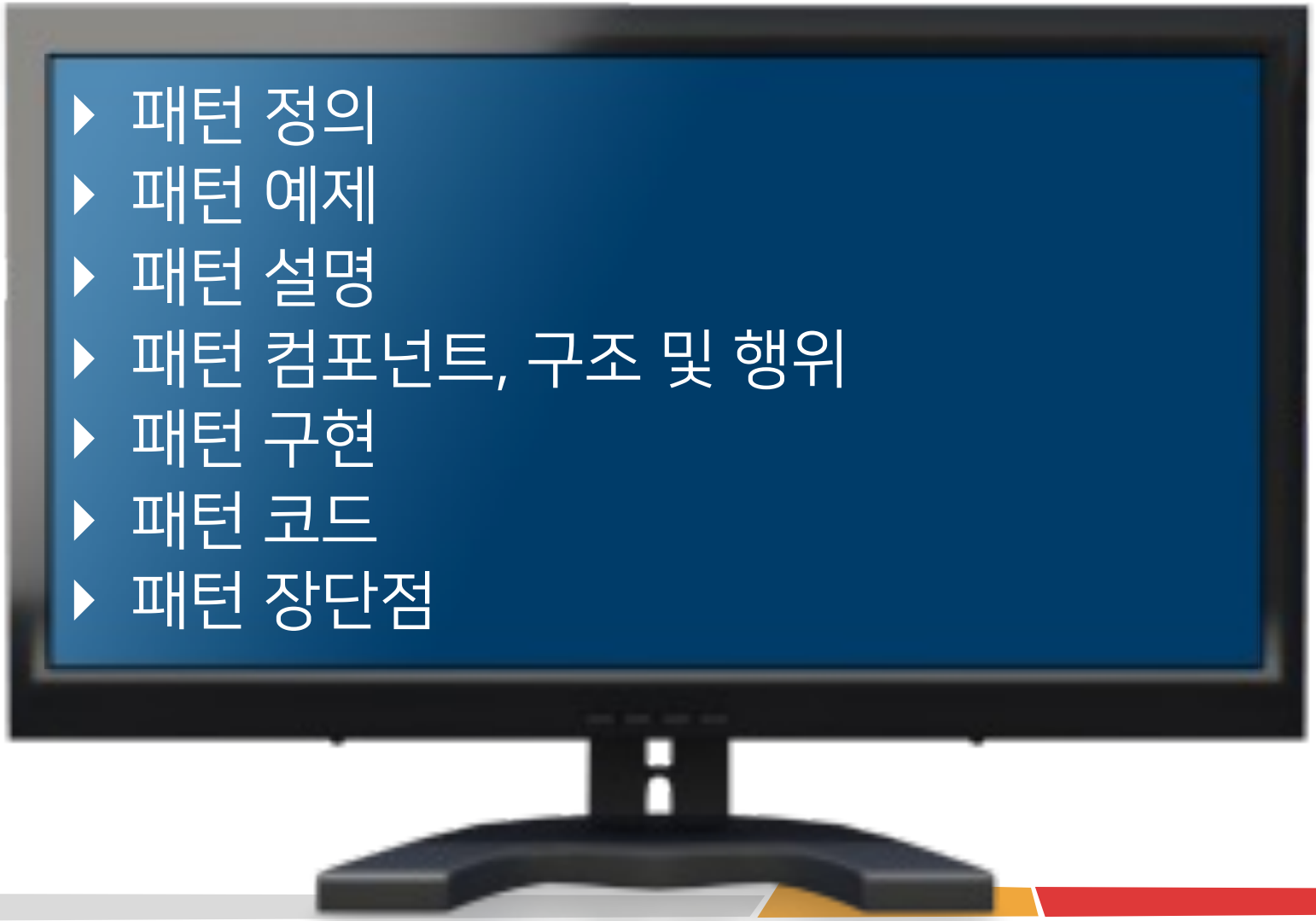
Spring 2023

소프트웨어 아키텍처 패턴: Microkernel

Seonah Lee

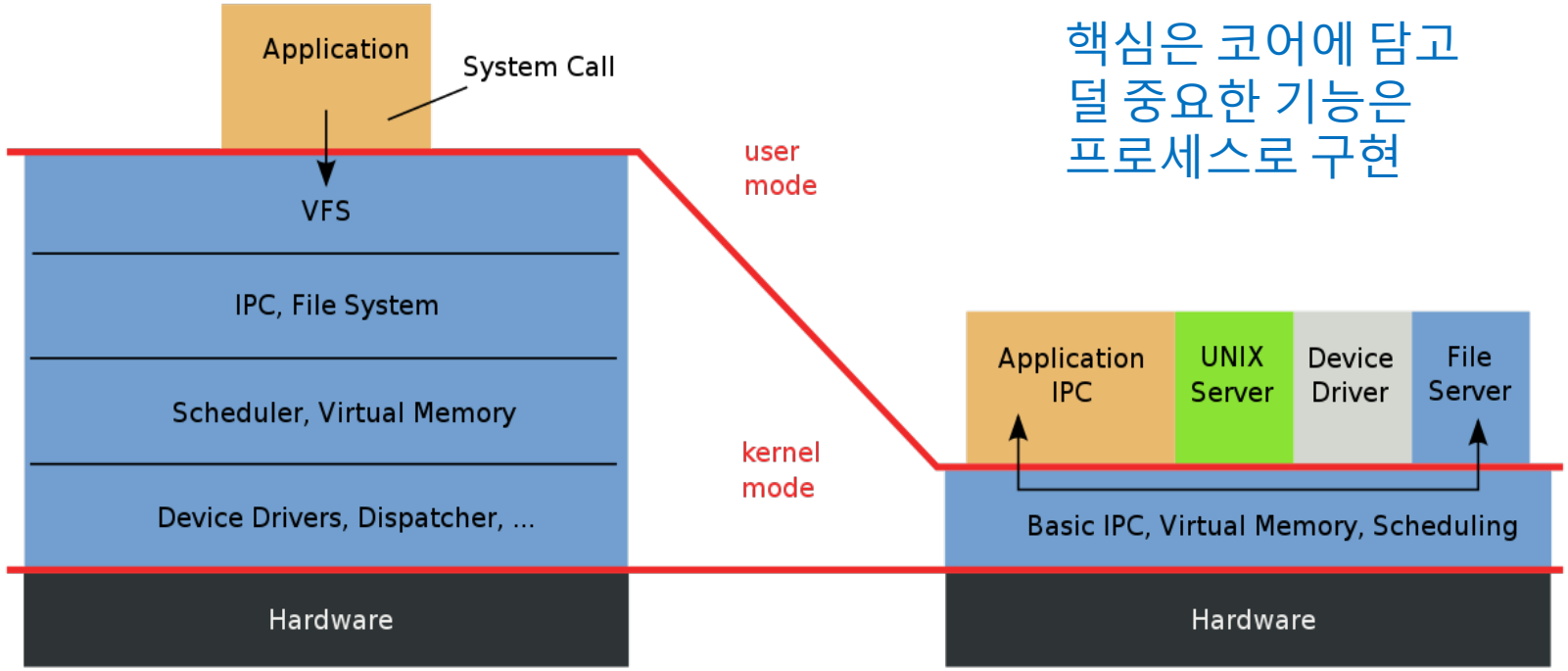
Gyeongsang National University

Microkernel 패턴



Microkernel Pattern: Definition

Monolithic Kernel
based Operating System



Microkernel
based Operating System

핵심은 코어에 담고
덜 중요한 기능은
프로세스로 구현

모노리틱 커널은
시스템콜(:12)을 이용해서
커널 기능을 이용한다.

반면 마이크로커널은메시지를
전달하는 방식으로 자원에
접근한다.

장점:
모듈 단위 개발로 확장과
포팅이 용이

단점:
메시지 전달하는 방식으로
자원에 접근하여 태스크
스위칭에 많은 오버헤드 초래

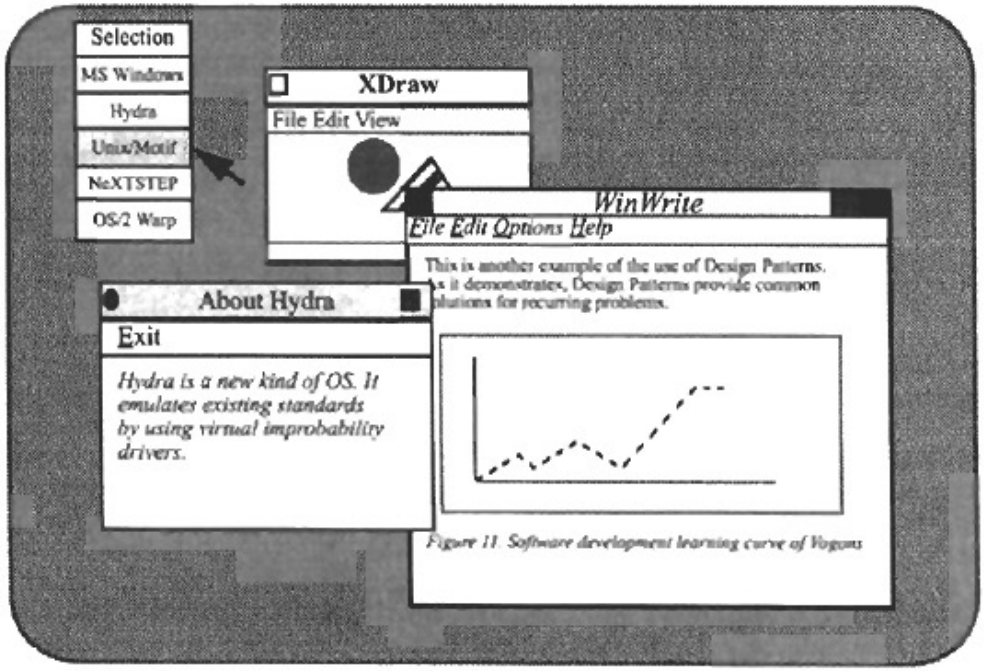
https://ko.wikipedia.org/wiki/마이크로커널

Microkernel Pattern: Definition

- 정의
  - 시스템의 요구사항이 변할 때, 이를 쉽게 시스템에 반영하기 위한 패턴
  - 일반적인 시스템의 기능에서 최소 핵심 기능은 공통으로 두고 변경이 가능한 확장 부분을 플러그인 방식으로 추가할 수 있는 패턴
- 예제
  - 마크**(Mach)**
    - 마이크로커널 기반으로 다른 **OS**를 에뮬레이션할 수 있는 기반 시스템 커널
  - **Window NT**, 코러스, **MKDE**, 아메바 등의 **OS**에 사용

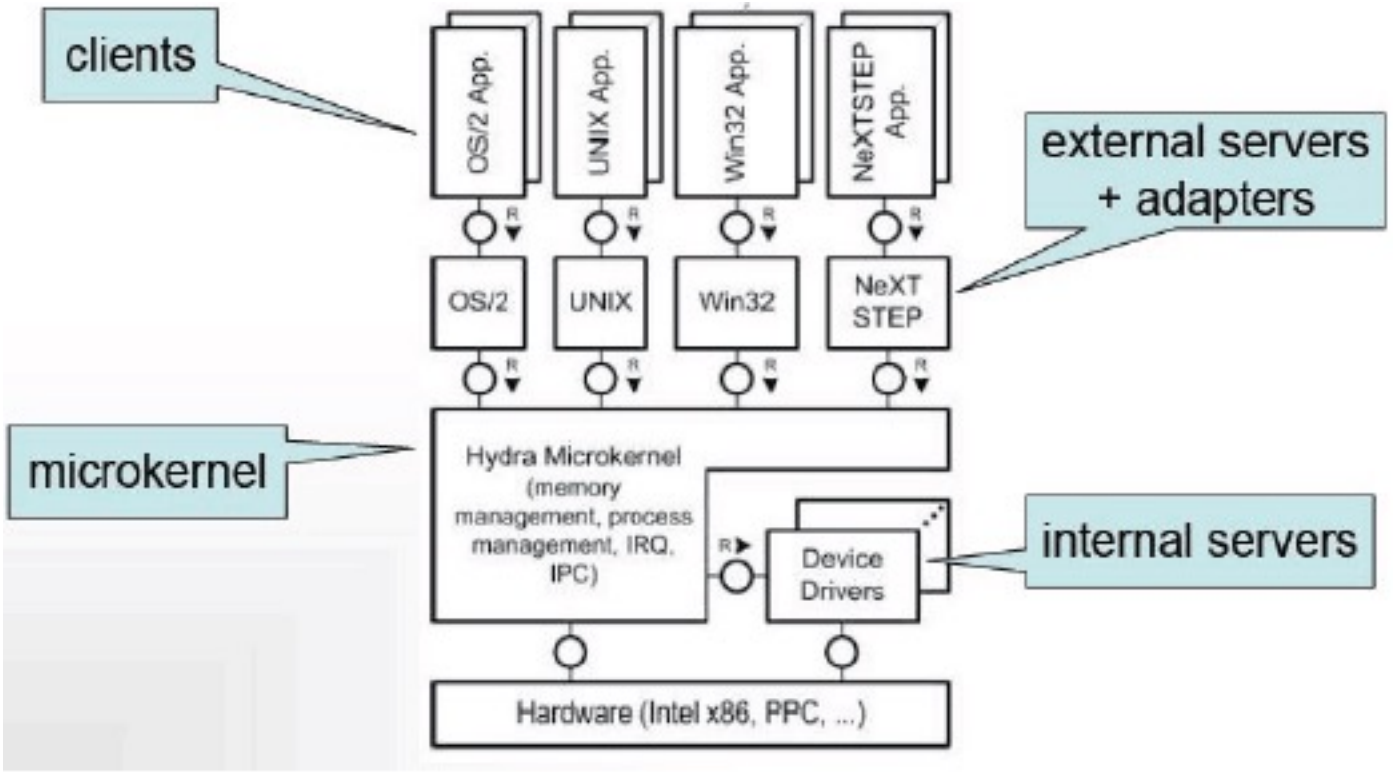
Microkernel Pattern: Example

- 데스크탑 컴퓨터용 새 운영체제를 개발한다고 가정
  - 관련 하드웨어에 쉽게 이식되고, 추가 개발에 쉽게 적응할 수 있어야 함
  - 기 보급된 운영체제 용 애플리케이션을 실행할 수 있어야 함



Microkernel Pattern: Example

- 데스크탑 컴퓨터용 새 운영체제를 개발한다고 가정



Hydra OS overview

Microkernel Pattern: Description

- 정황 (**Context**)
  - 동일한 기능을 기반으로 비슷한 프로그래밍 인터페이스를 제공하는 시스템 개발
- 문제(**Problem**)
  - 애플리케이션 플랫폼(**OS** 포함)은 하드웨어와 소프트웨어의 지속적인 발전과 변화를 반영해야 함
  - 새로운 기능을 쉽게 통합하기 위한 이식성과 확장성, 적용성을 보장해야 함
  - 기존 표준에 적합하게 만들어진 애플리케이션을 실행할 수 있어야 함. 이를 위해 기존 애플리케이션이 기존 표준들을 에뮬레이션해야 함
  - 핵심 기능은 최소한의 메모리 크기를 가진 한 개의 컴포넌트로 분리해야 함

Microkernel Pattern: Description

- 해법 (Solution)
  - **Microkernel** 컴포넌트로 캡슐화 하고 다른 컴포넌트와 통신하는 기능을 가짐
  - **Microkernel** 은 파일이나 프로세스처럼 리소스를 저장하는 역할을 하고 다른 컴포넌트들이 접속할 수 있도록 인터페이스를 제공함
  - **Microkernel** 내에 구현할 필요가 없는 핵심 기능은 **Internal server**로 분리시켜 크기나 복잡성을 줄임
  - **External server**에는 **External server** 자체의 뷰를 구현하고 **Microkernel**의 인터페이스를 이용하여 연동함
  - **Client**는 **Microkernel**의 통신 기능을 이용하여 **External server**와 통신

Microkernel Pattern: Components

External Server

- 자체 클라이언트를 위한 프로그래밍 인터페이스를 제공

Microkernel

- 핵심 메커니즘을 제공
- 통신 기능을 제공
- 시스템의 종속성들을 캡슐화함
- 리소스를 관리하고 제어

Internal Server

- 추가 서비스를 구현
- 특정 시스템에 맞춰진 구현을 캡슐화함

Adapter

- 클라이언트와 통신하는 기능 등의 시스템 종속성을 숨김
- 클라이언트를 대신해 외부 서버가 메소드를 호출

Client

- 특정 외부 서버 하나와 연결된 애플리케이션 역할

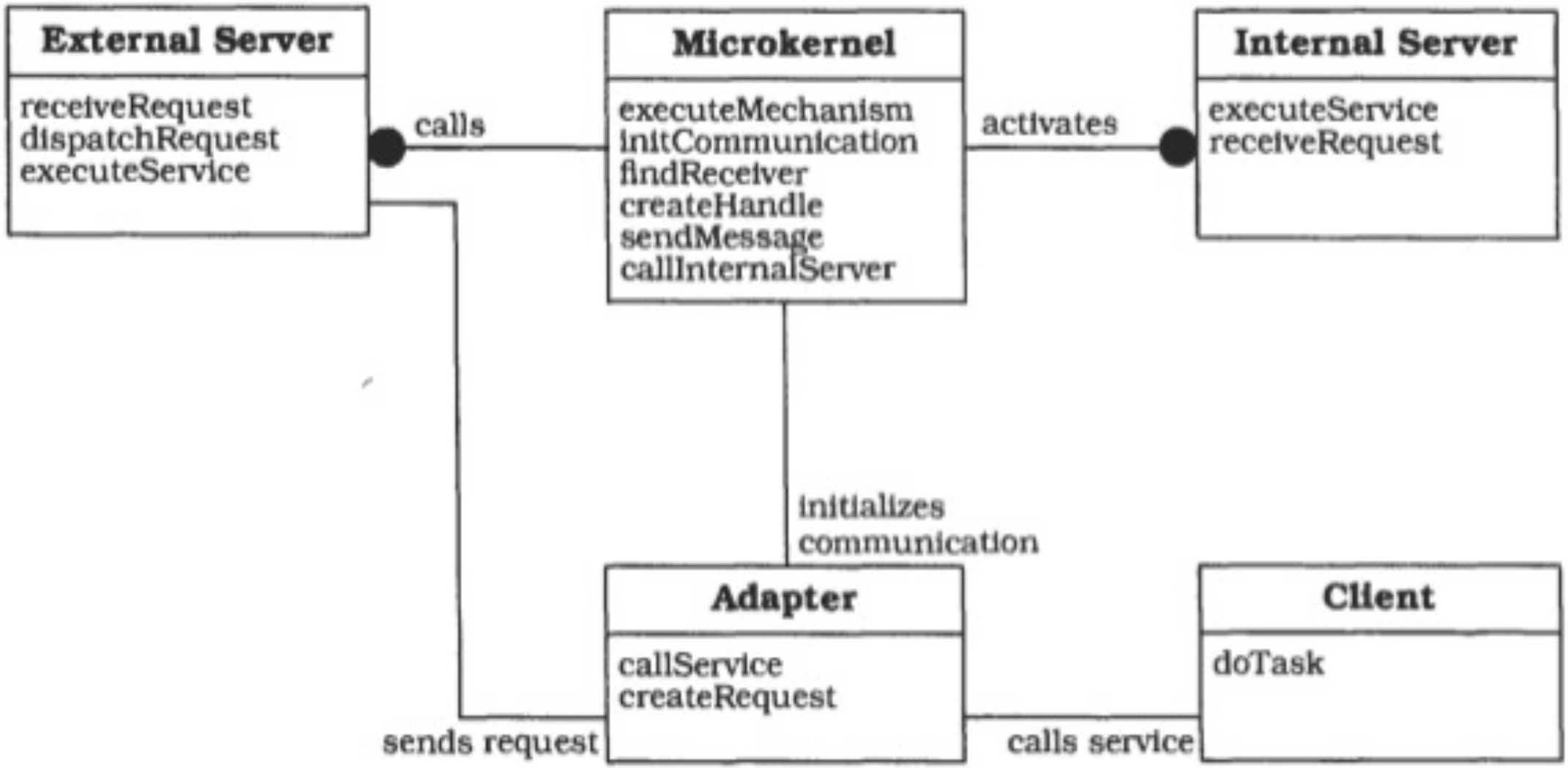
Microkernel Pattern: Components

- 마이크로 커널
  - 핵심 메커니즘 제공
  - 통신 기능 제공
  - 시스템 종속성들을 캡슐화
  - 리소스를 관리하고 제어
- 내부 서버
  - 추가 서비스를 구현(예: 디바이스 드라이버)
  - 특정 시스템에 맞춰진 구현 캡슐화

Microkernel Pattern: Components

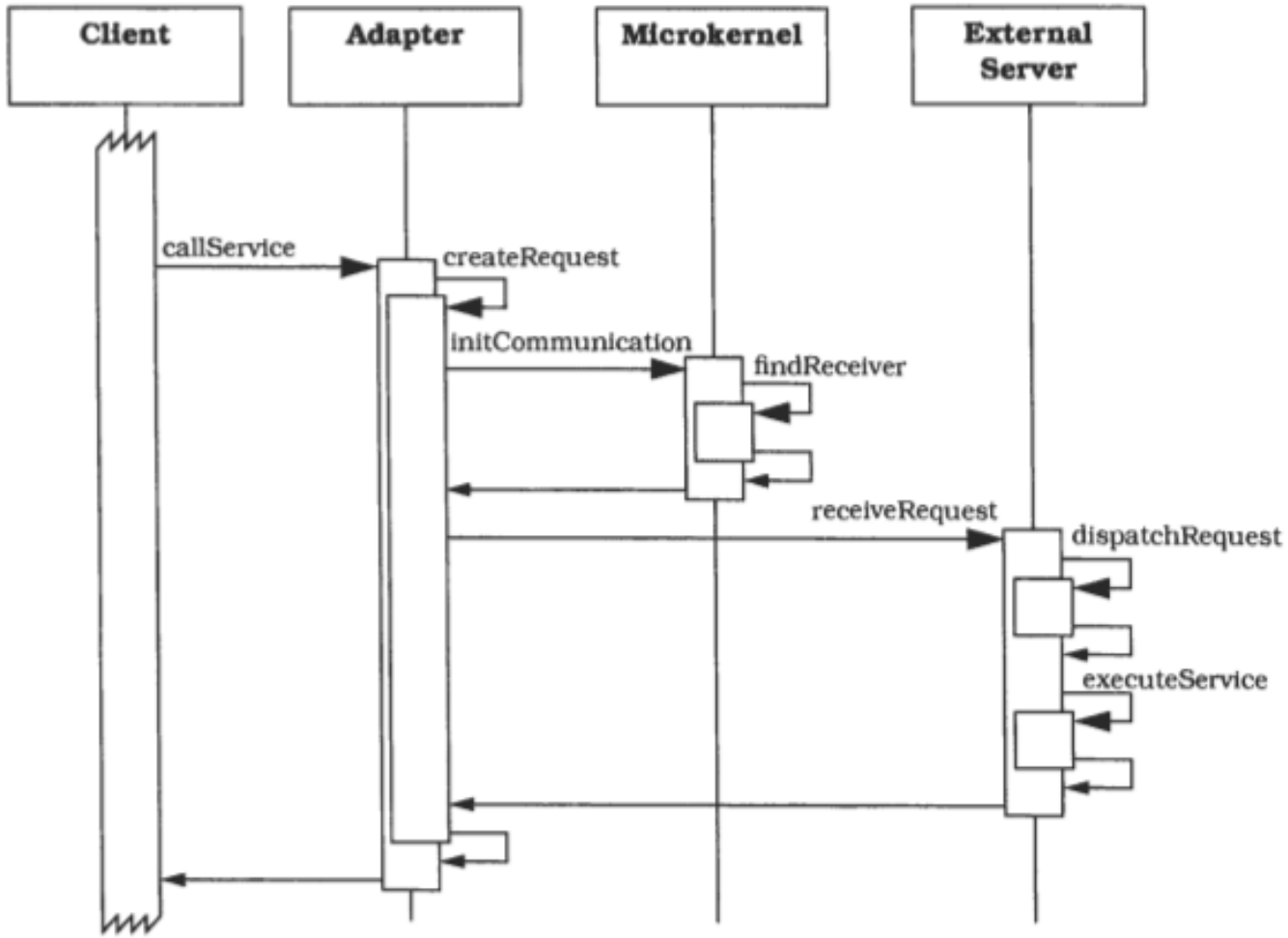
- 외부 서버
  - 자체 클라이언트를 위한 프로그래밍 인터페이스 제공
- 클라이언트
  - 특정 외부 서버 하나와 연결된 애플리케이션
- 어댑터
  - 클라이언트와 통신하는 기능 등의 시스템 종속성을 숨김
  - 클라이언트를 대신해 외부 서버의 메소드를 호출

Microkernel Pattern: Structure



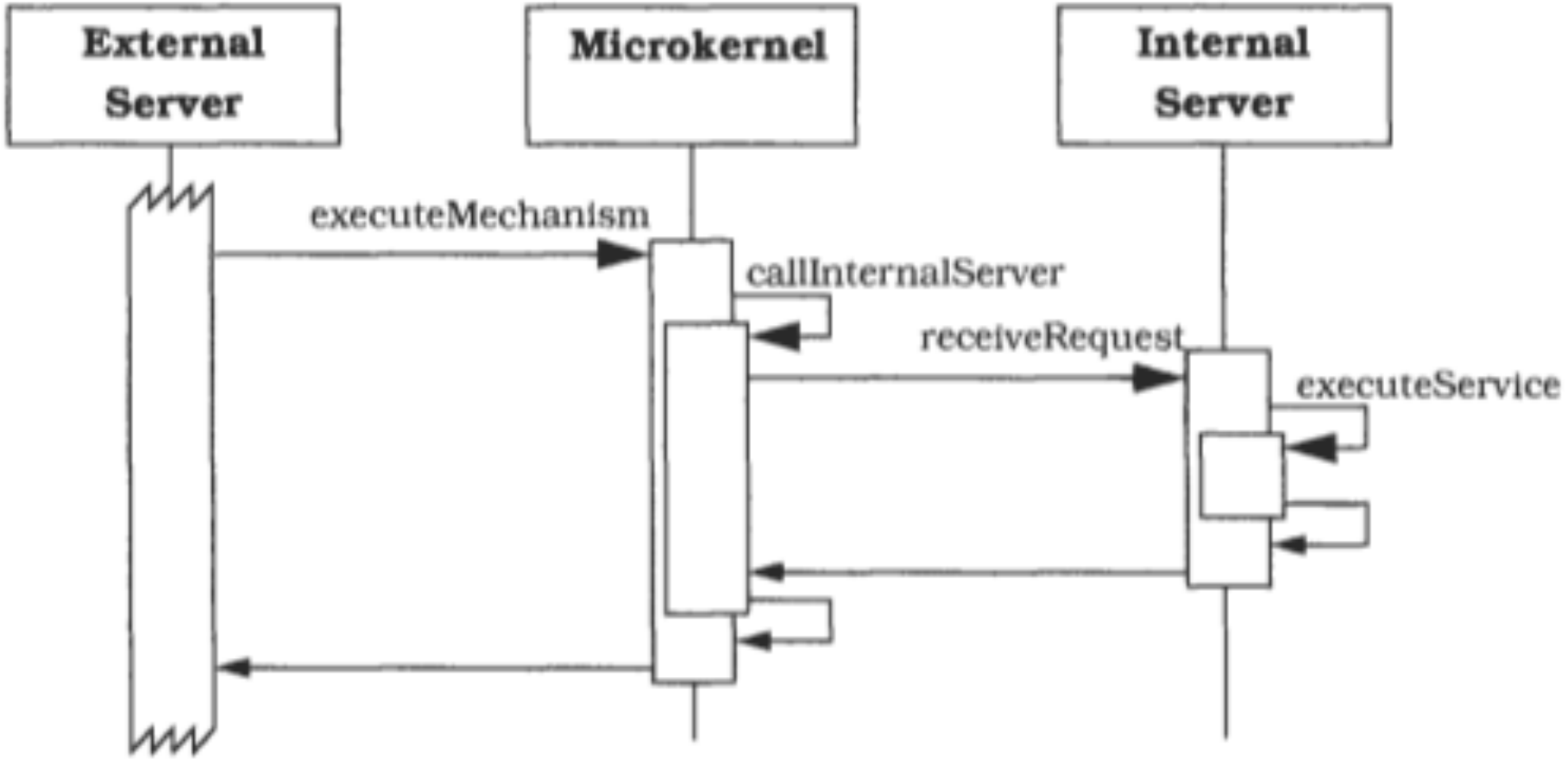
Microkernel Pattern: Behavior

- 클라이언트가 외부 서버에 서비스를 호출할 때의 동작

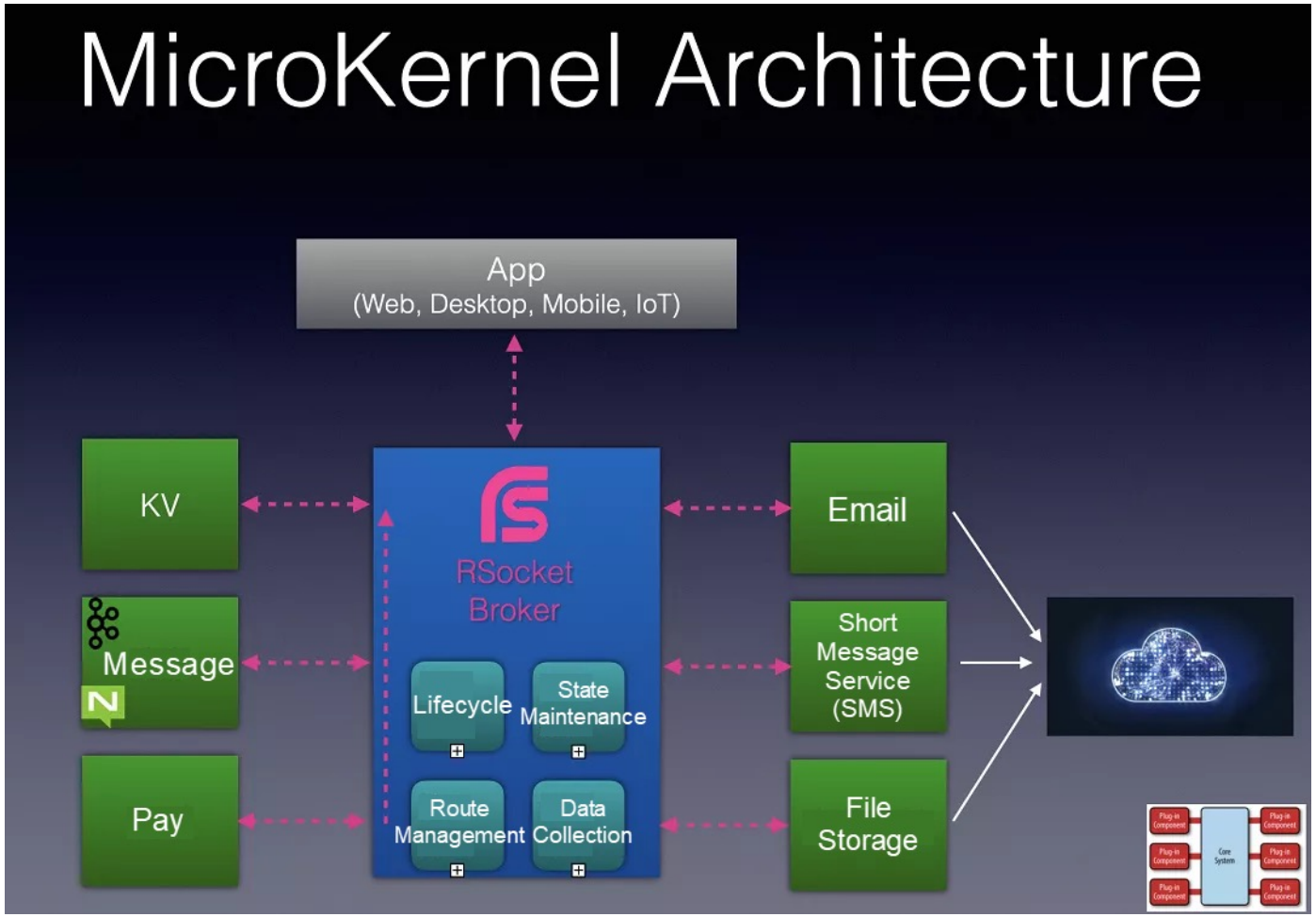


Microkernel Pattern: Behavior

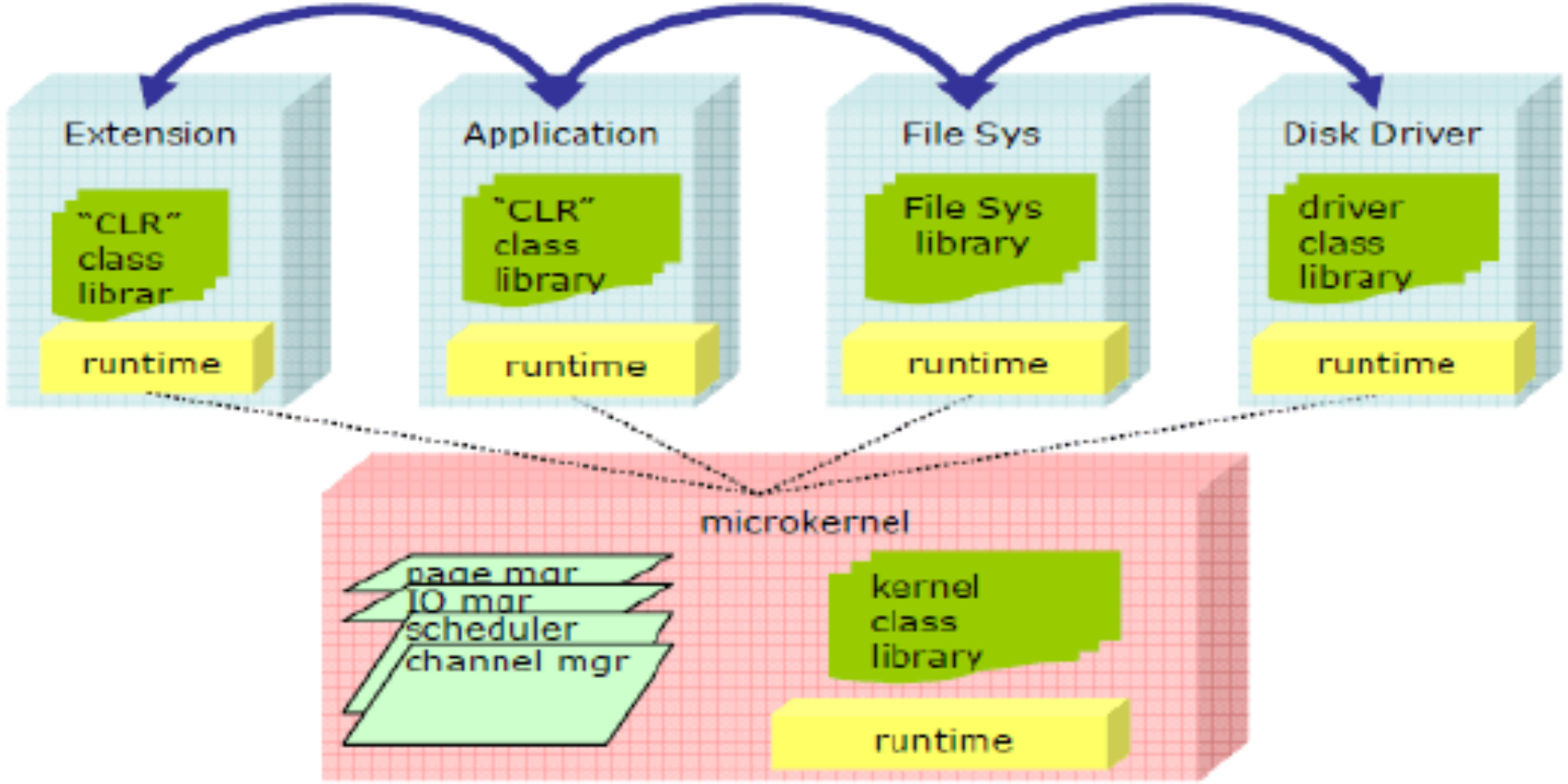
- 외부 서버가 내부 서버에서 제공하는 서비스에 대한 요청을 할 때 하는 동작



Microkernel Pattern: Case Studies

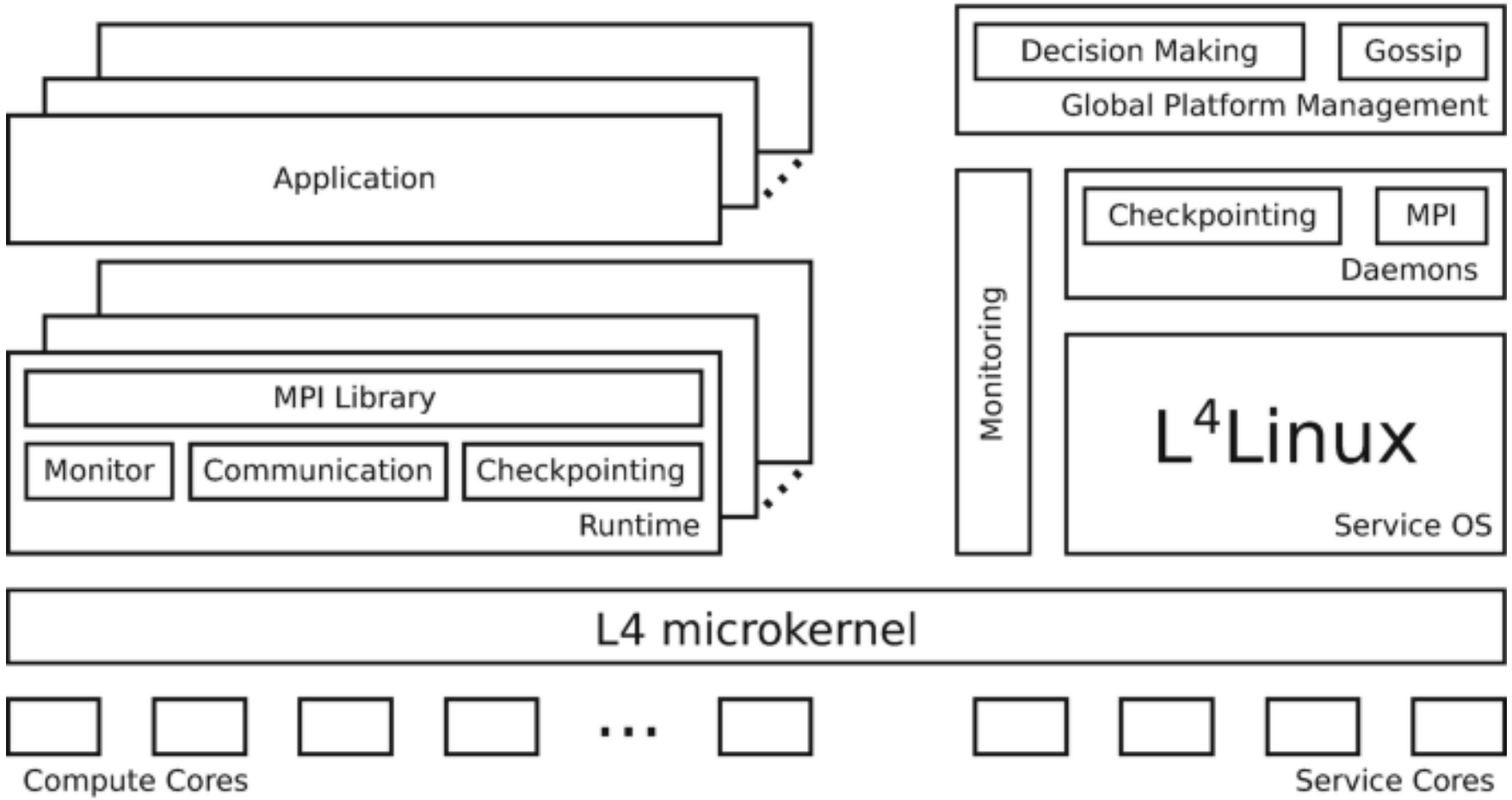


Microkernel Pattern: Case Studies

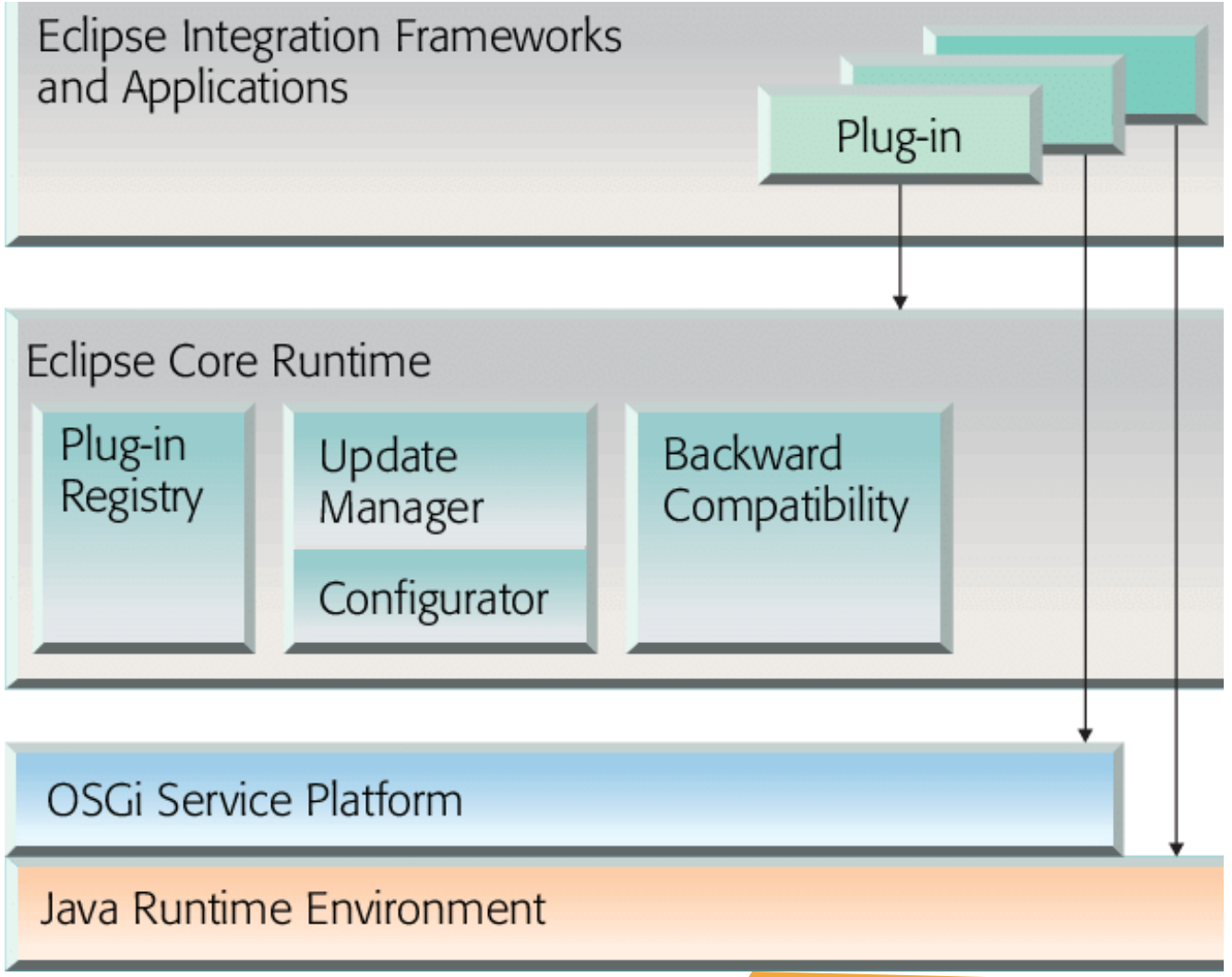


Singularity Architecture

Microkernel Pattern: Case Studies



Microkernel Pattern: Case Studies: Eclipse



https://www.researchgate.net/figure/Eclipse-architecture-overview_fig1_224101700

Microkernel Pattern: Benefits

- 이식성이 보장
  - 시스템을 새로운 환경으로 이동 시, **Microkernal** 만 수정하면 됨
- 유연성과 교환가능성이 확보
  - **Internal/External Server**를 수정하거나 확장
- 유지보수성과 가변성을 향상
  - 정책(외부서버)과 메커니즘(**Microkernal**)을 분리함
- **Distributed Microkernal** 변형 시
  - 확장성(**Scalability**) 확보(새 머신 추가), 신뢰성 지원(서버 문제), 투명성을 제공

Microkernel Pattern: Liabilities

- 시스템 설계가 복잡하고 개발이 어려움
  - **Monolithic** 소프트웨어 시스템과 비교 시 설계와 구현이 훨씬 복잡함
- 성능이 떨어짐
  - 마이크로 커널 시스템은 **internal server**와 **external server** 에 대한 호출로 인하여, 하나의 애플리케이션에서 훨씬 많은 프로세스 간 통신이 필요함

Question?

Seonah Lee
saleese@gmail.com