

# 1주차

## 1. 디자인 패턴

### 1.1 생성 패턴

#### 1) 싱글톤 패턴

##### [정의]

- 하나의 클래스에 하나의 인스턴스만 가지는 패턴

##### [장점]

- 하나의 인스턴스를 만들어 다른 모듈들이 공유하며 사용하기 때문에 인스턴스 생성할 때는 비용이 줄어듦

##### [단점]

- 의존성이 높아짐

### 1.2 팩토리 패턴

##### [정의]

- 객체를 사용하는 코드에서 객체 생성 부분을 떼어내 추상화한 패턴
- 상속관계에 있는 두 클래스에서 상위 클래스가 중요한 뼈대를 결정, 하위 클래스에서 객체 생성에 관한 구체적인 내용을 결정

##### [장점]

- 상위 클래스에서는 인스턴스 생성 방식에 대해 알 필요가 없어 유연성을 갖게 됨
- 객체 생성 로직이 분리 되어 있어 유지 보수성 증가 (리팩토링시에 한 곳만 고치면 된다)

##### [단점]

- 제품 객체의 갯수마다 공장 서브 클래스를 모두 구현해야되서 클래스 폭발이 일어날수 있음

## 1.2 구조 패턴

### 1) 프록시 패턴

#### [정의]

- 대상 객체에 접근하기 전에 그 접근에 대한 흐름을 가로채 대상 객체 앞단의 인터페이스 역할을 하는 디자인 패턴

#### [활용]

- 데이터 검증, 캐싱, 로깅...

### 2) 이터레이터 패턴

#### [정의]

- 이터레이터를 사용해 컬렉션의 요소들에 접근하는 패턴

## 1.3 행위 패턴

### 1) 전략 패턴

#### [정의]

- 객체의 행위를 바꾸고 싶은 경우 '직접' 수정하지 않고 전략이라고 부르는 '캡슐화한 알고리즘'을 컨텍스트 안에서 바꿔주면서 상호 교체가 가능하게 만드는 패턴

#### [활용]

- passport
  - Node.js에서 인증 모듈을 구현할 때 쓰는 미들웨어 라이브러리
  - 여러가지 전략을 기반으로 인증할 수 있게 함

### 2) 옵저버 패턴

#### [정의]

- 주체가 어떤 객체의 상태 변화를 관찰하다가 상태 변화가 있을 때마다 메서드 등을 통해 옵저버 목록에 있는 옵저버들에게 변화를 알려줌

## 1.4 그외

### 1) 노출모듈 패턴

#### [정의]

- 즉시 실행 함수를 통해 private, public 같은 접근 제어자를 만드는 패턴

#### [활용]

- CJS(CommonJS) 모듈 방식
  - 노출모듈 패턴을 기반으로 만든 자바스크립트 모듈 방식

### 2) MVC 패턴

#### [정의]

- 모델(Model), 뷰(View), 컨트롤러(Controller)로 이루어진 패턴

#### [활용]

- 리액트
  - MVC 패턴을 이용한 대표적인 라이브러리
  - 가상 DOM을 통해 실제 DOM을 조작하는 것을 추상화해서 성능을 높임

### 3) MVP 패턴

#### [정의]

- MVC에서 C에 해당하는 컨트롤러가 프레젠테어로 교체된 패턴
- MVC 패턴으로부터 파생

#### [특징]

- 뷰와 프레젠테어는 일대일 관계이기 때문에 MVC 패턴보다 더 강한 결합을 지니고 있음

### 4) MVVM 패턴

#### [정의]

- MVC에서 C에 해당하는 컨트롤러가 뷰모델로 교체된 패턴

#### [특징]

- 뷰모델은 뷰를 더 추상화한 계층
- MVC 패턴과 다르게 커맨드와 데이터 바인딩을 가지는 것이 특징

#### [장점]

- 뷰와 뷰모델 사이의 양방향 데이터 바인딩을 지원
- UI를 별도의 코드 수정없이 재사용할 수 있음
- 단위 테스트가 쉬움

#### [활용]

- 뷰(Vue.js)
  - MVVM 패턴을 가진 대표적인 프레임워크
  - 반응형이 특징
  - 함수를 사용하지 않고 값 대입만으로도 변수가 변경

## 2. 프로그래밍 패러다임

### 2.1 선언형과 함수형 프로그래밍

#### [정의]

- 프로그래머에게 프로그래밍의 관점을 갖게 해주는 역할을 하는 개발 방법론
- 무엇을 풀어내는가에 집중하는 패러다임
- 프로그램은 함수로 이루어진 것이라라는 명제가 담겨 있는 패러다임

### 2.2 객체지향 프로그래밍

#### [정의]

- 객체들의 집합으로 프로그램의 상호 작용을 표현하며 데이터를 객체로 취급하여 객체 내부에 선언된 메서드를 활용하는 방식

### [특징]

- 추상화
  - 복잡한 시스템으로부터 핵심적인 개념 또는 기능을 간추려내는 것을 의미
- 캡슐화
  - 객체의 속성과 메서드를 하나로 묶고 일부를 외부에 감추어 은닉화하는 것
- 상속성
  - 상위 클래스의 특성을 하위 클래스가 이어받아서 재사용, 추가, 확장하는 것
- 다형성
  - 하나의 메서드나 클래스가 다양한 방법으로 동작하는 것
  - ex) 오버로딩, 오버라이딩

### [설계원칙]

- 단일 책임 원칙 (SRP)
  - 모든 클래스는 각각 하나의 책임만 가져야 함
- 개방-폐쇄 원칙 (OCP)
  - 기존 코드는 잘 변경하지 않으면서도 확장은 쉽게 할 수 있어야 함
- 리스코프 치환 원칙 (LSP)
  - 자식 클래스는 부모 클래스처럼 사용 가능해야 함
- 인터페이스 분리 원칙 (ISP)
  - 하나의 일반적인 인터페이스보다 구체적인 여러 개의 인터페이스를 만들어야 함
- 의존 역전 원칙 (DIP)
  - 상위 계층은 하위 계층의 변화에 대한 구현으로부터 독립

## 2.3 절차형 프로그래밍(OOP)

### [정의]

- 로직이 수행되어야 할 연속적인 계산 과정으로 이루어짐

#### [장점]

- 코드의 가독성이 좋음
- 실행 속도 빠름

#### [활용]

- 포트란
  - 대기 과학 관련 연산 작업
  - 머신 러닝의 배치 작업

## 3. 네트워크 기초

### 3.1 처리량과 지연 시간

#### 처리량

- 링크를 통해 전달되는 단위 시간당 데이터양
- 단위로는 bps 사용
- 사용자들이 많이 접속할 때마다 커지는 트래픽, 네트워크 장치 간의 대역폭 등에 영향을 받음

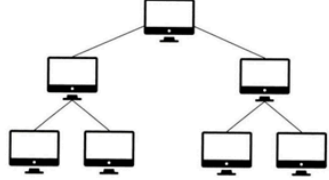
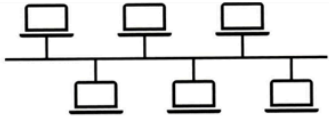
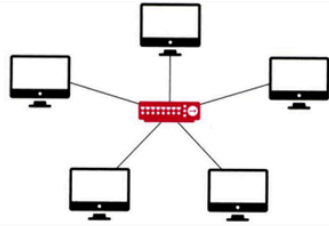
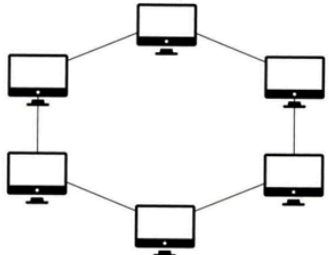
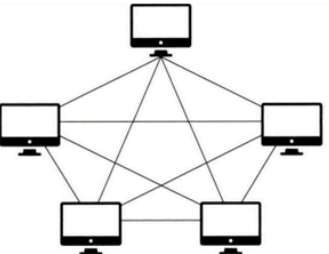
#### 지연시간

- 요청이 처리되는 시간
- 어떤 메시지가 두 장치 사이를 왕복하는데 걸린 시간
- 매체타입, 패킷 크기, 라우터의 패킷 처리 시간에 영향 받음

### 3.2 네트워크 토폴로지와 병목 현상

#### 네트워크 토폴로지

- 노드와 링크가 어떻게 배치되어 있는지에 대한 방식 or 연결형태를 의미

	종류	정의 & 특징
	트리 토폴로지	<ul style="list-style-type: none"> <li>• 트리 형태로 배치한 네트워크 구성</li> <li>• 노드의 추가, 삭제가 쉬움</li> <li>• 특정 노드에 트래픽이 집중될 때 상위 노드에 영향을 끼칠 수 있음</li> </ul>
	버스 토폴로지	<ul style="list-style-type: none"> <li>• 중앙 통신 회선 하나에 여러 개의 노드가 연결되어 공유하는 네트워크 구성</li> <li>• 근거리 통신망(LAN)에서 사용</li> <li>• 설치 비용 ↓ 신뢰성 ↑</li> <li>• 중앙 통신 회선에 노드 추가 및 삭제 쉬움</li> <li>• 스푸핑이 가능함</li> </ul>
	스타 토폴로지	<ul style="list-style-type: none"> <li>• 중앙에 있는 노드에 모두 연결된 네트워크 구성</li> <li>• 노드 추가 및 제거 탐지 쉬움</li> <li>• 패킷의 충돌 가능성 ↓</li> <li>• 중앙 노드에 장애 발생하면 전체 네트워크 사용X</li> <li>• 설치 비용 고가</li> </ul>
	링형 토폴로지	<ul style="list-style-type: none"> <li>• 각각의 노드가 양 옆에 두 노드와 연결하여 고리처럼 하나의 연속된 길을 통해 통신하는 망 구성 방식</li> <li>• 노드 수가 증가되어도 네트워크 상의 손실이 거의 없고, 충돌 가능성 적음</li> <li>• 네트워크 구성 변경이 어려움</li> </ul>
	메시 토폴로지	<ul style="list-style-type: none"> <li>• 그물망처럼 연결되어 있는 구조</li> <li>• 한 단말 장치에 장애가 발생해도 여러 개의 경로가 존재하여 네트워크 계속 사용 및 트래픽 분산 처리 가능</li> <li>• 구축비용과 운용 비용이 고가</li> </ul>

## 병목현상

- 전체 시스템의 성능이나 용량이 하나의 구성 요소로 인해 제한을 받은 현상

#### [네트워크 병목 현상 주된 원인]

- 네트워크 대역폭
- 네트워크 토폴로지
- 서버 CPU, 메모리 사용량
- 비효율적인 네트워크 구성

### 3.3 네트워크 분류

분류	정의
LAN	<ul style="list-style-type: none"><li>• 근거리 통신망</li><li>• 같은 건물이나 캠퍼스 같은 좁은 공간에서 운영</li><li>• 전송 속도 높고, 혼잡하지 않음</li></ul>
MAN	<ul style="list-style-type: none"><li>• 대도시 지역 네트워크</li><li>• 도시 같은 넓은 지역에서 운영</li><li>• 전송 속도 평균, LAN보다 혼잡</li></ul>
WAN	<ul style="list-style-type: none"><li>• 광역 네트워크</li><li>• 국가 또는 대륙 등 넓은 지역에서 운영</li><li>• 전송 속는 낮고, MAN보다 혼잡</li></ul>

### 3.4 네트워크 성능 분석 명령어



명령어 종류	의미
ping	<ul style="list-style-type: none"> <li>• 네트워크 상태를 확인하려는 대상 노드를 향해 일정 크기의 패킷을 전송하는 명령어</li> <li>• 해당 노드의 패킷 수신상태, 도달하기까지 시간, 해당 노드까지 네트워크가 잘 연결되었는지 등을 알 수 있음</li> <li>• ping [IP주소 or 도메인 주소]로 실행</li> </ul>
netstat	<ul style="list-style-type: none"> <li>• 접속되어 있는 서비스들의 네트워크 상태를 표시하는데 사용</li> <li>• 네트워크 접속, 라우팅 테이블, 네트워크 프로토콜 등 리스트를 보여줌</li> <li>• 서비스의 포트가 열려 있는지 확인할 때 사용</li> </ul>
nslookup	<ul style="list-style-type: none"> <li>• DNS에 관련된 내용을 확인하기 위해 쓰는 명령어</li> <li>• 특정 도메인에 매핑된 IP를 확인하기 위해 사용</li> </ul>
tracert	<ul style="list-style-type: none"> <li>• 목적지 노드까지 네트워크 경로를 확인할 때 사용하는 명령어</li> <li>• 목적지 노드까지 구간들 중 어느 구간에서 응답 시간이 느려지는지 등을 확인</li> </ul>

### 3.5 네트워크 프로토콜 표준화

- 다른 장치들끼리 데이터를 주고받기 위해 설정된 공통된 인터페이스를 말함
- 이런 프로토콜은 IEEE 또는 IETF라는 표준화 단체가 정함

## 4. TCP/IP 4계층 모델

### 4.1 계층 구조

## 애플리케이션 계층

- 웹 서비스, 이메일 등 서비스를 실질적으로 사람들에게 제공
- FTP, HTTP, SSH, SMTP, DNS 등 응용 프로그램 사용

## 전송 계층

- 송신자와 수신자를 연결하는 통신 서비스를 제공
- TCP, UDP 등

## 인터넷 계층

- 장치로부터 받은 네트워크 패킷을 IP 주소로 지정된 목적지로 전송하기 위해 사용
- IR, ARP, ICMP 등..

## 링크계층

- 전선, 광섬유, 무선 등으로 실질적으로 데이터를 전달하며 장치 간에 신호를 주고받는 규칙을 정하는 계층
- 물리 계층과 데이터 링크 계층으로 나누기도 함
- 물리 계층은 유선 LAN, 무선 LAN을 통해 0과 1로 이루어진 데이터를 보내는 계층
- 데이터 링크 계층은 이더넷 프레임을 통해 에러 확인, 흐름제어, 접근 제어를 담당하는 계층

## 4.2 PDU

- PDU는 네트워크의 어떠한 계층에서 계층으로 데이터가 전달 될때 한 덩어리의 단위
- 제어 관련 정보들이 포함된 헤더, 데이터를 의미하는 페이로드로 구성
- 계층마다 부르는 명칭이 다름
  - 애플리케이션 계층: 메시지
  - 전송계층: 세그먼트(TCP), 데이터 그램(UDP)

- 인터넷 계층: 패킷
- 링크 계층: 프레임(데이터 링크 계층), 비트(물리 계층)

## 5. 네트워크 기기

### 5.1 네트워크 기기의 처리 범위

- 네트워크 기기는 계층별로 처리 범위를 나눔

애플리케이션 계층	L7 스위치
인터넷 계층	라우터, L3 스위치
데이터 링크 계층	L2 스위치, 브리지
물리 계층	NIC, 리피터, AP

### 5.2 애플리케이션 계층을 처리하는 기기

#### L7 스위치

- 애플리케이션 계층을 처리하는 기기
- 스위치는 여러 장비를 연결하고 데이터 통신을 중재하며 목적지가 연결된 포트로만 전기 신호를 보내 데이터를 전송하는 통신 네트워크 장비임

#### [특징]

- 헬스체크
  - 전송주기와 재전송 회수 등을 설정한 이후 반복적으로 서버에 요청을 보내는 것

- 헬스체크를 통해 서버 또는 비정상적인 서버를 판별
- 로드밸런서를 이용한 서버 이중화
  - 로드밸런서의 대표적인 기능은 서버 이중화
  - 서비스를 안정적으로 운용하기 위해서는 2대 이상의 서버는 필수적

## 5.3 인터넷 계층을 처리하는 기기

### 라우터

- 인터넷 계층을 처리하는 기기
- 라우터는 다른 네트워크에 존재하는 장치끼리 서로 데이터를 주고받을 때 패킷 소모를 최소화하고 경로를 최적화하여 최소 경로로 포워딩하는 라우팅을 하는 장비

### L3 스위치

- 인터넷 계층을 처리하는 기기
- L2 스위치의 기능과 라우팅 기능을 갖춘 장비를 말함
- 라우터는 소프트웨어 기반의 라우팅과 하드웨어 기반의 라우팅으로 나뉘어짐
- 하드웨어 기반의 라우팅을 담당하는 장치가 L3 스위치

## 5.4 데이터 링크 계층을 처리하는 기기

### L2 스위치

- 데이터 링크 계층을 처리하는 기기
- L2 스위치는 장치들의 MAC 주소를 MAC 주소 테이블을 통해 관리
- 연결된 장치로부터 패킷이 왔을 때 패킷 전송을 담당

### 브리지

- 데이터 링크 계층을 처리하는 기기

- 두 개의 근거리 통신망(LAN)을 상호 접속할 수 있도록 하는 통신망 연결 장치
- 장치에서 받아온 MAC 주소를 MAC 주소 테이블로 관리

## 5.5 물리 계층을 처리하는 기기

### NIC

- 물리 계층을 처리하는 기기
- LAN 카드라고 하는 네트워크 인터페이스 카드는 2대 이상의 컴퓨터 네트워크를 구성하는데 사용
- 네트워크와 빠른 속도로 데이터를 송수신할 수 있도록 컴퓨터 내에 설치하는 확장카드

### 리피터

- 물리 계층을 처리하는 기기
- 들어오는 약해진 신호 정보를 증폭하여 다른 쪽으로 전달하는 장치

### AP

- 물리 계층을 처리하는 기기
- 패킷을 복사하는 기기

## 6. IP주소

### 6.1 ARP

- IP 주소로부터 MAC 주소를 구하는 IP와 MAC 주소의 다리 역할을 하는 프로토콜
- ARP를 통해 가상주소인 IP 주소를 실제주소인 MAC 주소로 변환

### 6.2 홉바이홉 통신

- 홉바이홉 통신은 IP 주소를 통해 통신하는 과정
- 통신 장치에 있는 라우팅 테이블의 IP를 통해 시작 주소부터 시작하여 다음 IP로 이동하는 라우팅 과정을 거쳐 패킷이 최종 목적지까지 도달하는 통신

## 라우팅 테이블

- 송신지에서 수신지까지 도달하기 위해 사용
- 라우터에 들어가 있는 목적지 정보들과 그 목적지로 가기 위한 방법들이 있는 테이블을 뜻함

## 게이트웨이

- 서로 다른 통신망, 프로토콜을 사용하는 네트워크 간의 통신을 가능하게 하는 관문 역할을 하는 컴퓨터나 소프트웨어를 뜻함

## 6.3 IP 주소 체계

- IP주소는 IPv4와 IPv6으로 나뉨
- IPv4는 32비트를 8비트 단위로 점을 찍어 표기
- IPv6는 64비트를 16비트 단위로 점을 찍어 표기

## 클래스 기반 할당 방식

- A, B, C, D, E 다섯 개의 클래스로 구분하는 방식
- 클래스 A, B, C는 일대일 통신으로 사용
- 클래스 D는 멀티캐스트 통신
- 클래스 E는 앞으로 사용할 예비용으로 사용

## DHCP

- IP 주소 및 기타 통신 매개변수를 자동으로 할당하기 위한 네트워크 관리 프로토콜임

- 라우터와 게이트웨이 장비에 DHCP 기능이 존재 → 가정용 네트워크에서 IP주소를 할당

## NAT

- 패킷이 라우팅 장치를 통해 전송되는 동안 패킷의 IP 주소 정보를 수정하여 IP 주소를 다른 주소로 매핑하는 방법
- ICS, RRAS, Netfilter → NAT가 가능한 소프트웨어
- NAT를 이용해서 사설 IP를 공인 IP로 변환 or 공인 IP를 사설 IP로 변환

### [단점]

- 호스트 숫자에 따라서 접속이 느려질 수 있음

## 7. HTTP

### 7.1 HTTP/1.0

- 한 연결당 하나의 요청을 처리 → RTT 증가

### [단점]

- RTT 증가
  - RTT는 패킷이 목적지에 도달하고 나서 다시 출발지로 도달하기 까지 걸리는 시간
  - 즉 패킷 왕복 시간
  - 연결할 때마다 RTT 증가 → 서버에 부담, 사용자 응답시간 증가

### RTT 증가 해결방법

- 이미지 스플리팅
  - 많은 이미지 다운로드시에 과부하
  - 많은 이미지가 합쳐 있는 하나의 이미지를 다운

- 이를 기반으로 background-image의 position을 이용해서 이미지 표시
- 코드 압축
  - 개행 문자, 빈칸을 없애서 코드의 크기를 최소화하는 방법
- 이미지 Bask64 인코딩
  - 이미지 파일을 64진법으로 이루어진 문자열로 인코딩하는 방법

## 7.2 HTTP/1.1

- 매번 TCP를 연결하는 것이 아닌 한번 TCP 초기화를 한 이후에 keep-alive라는 옵션으로 여러 개의 파일을 송수신

### [단점]

- HOL Blocking
  - 네트워크에서 같은 큐에 있는 패킷이 첫 번째 패킷에 의해 지연될 때 발생하는 성능 저하 현상
- 무거운 헤더 구조
  - HTTP/1.1의 헤더에는 많은 메타데이터가 압축이 되지 않은 채 들어 있어 무거움

## 7.3 HTTP/2

- HTTP/1.x보다 지연 시간이 줄고, 응답시간이 더 빠름
- 멀티플렉싱, 헤더 압축, 서버 푸시, 요청의 우선순위 처리를 지원하는 프로토콜

### [장점]

- 멀티플렉싱
  - 여러 개의 스크림을 사용하여 송수신하는 것
  - 특정 스트림의 패킷 손실되어도 나머지 스트림은 멀쩡하게 동작함
- 헤더 압축
  - 허프만 코딩 압축 알고리즘을 사용하는 HPACK 압축 형식



- 허프만 코딩 압축은 문자열을 문자단위로 쪼개 빈도수를 세어 빈도가 높은 정보는 적은 비트 수로 표현, 빈도가 낮은 정보는 비트 수를 많이 사용하여 표현 → 전체 데이터의 표현에 불필요한 비트양을 줄임
- 서버 푸시
  - 클라이언트 요청 없이 서버가 바로 리소스를 푸시할 수 있음
  - HTTP/1.1에서는 클라이언트가 서버에 요청을 해야 파일을 다운받을 수 있음

## 7.4 HTTPS

- 애플리케이션 계층과 전송 계층 사이에 신뢰 계층인 SSL/TLS 계층을 넣은 신뢰할 수 있는 HTTP 요청

## SSL/TLS

- 전송 계층에서 보안을 제공하는 프로토콜
- 제 3자가 메시지를 도청하거나 변조하지 못하도록 함

## 7.5 HTTP/3

- HTTP의 세번째 버전
- QUIC이라는 계층위에서 돌아가며, UDP 기반

### [장점]

- 초기 연결 설정시 지연 시간 감소
  - QUIC는 TCP를 사용하지 않기 때문에 통신을 시작할 때 번거로운 3-웨이 핸드셰이크 과정을 거치지 않아도 됨
- 멀티플렉싱

## 질문지

### 싱글톤 패턴에 대해 설명하고, 활용사례에 대해 설명하시오

싱글톤 패턴은 하나의 클래스에 오직 하나의 인스턴스를 가지는 패턴을 말합니다. 이러한 싱글톤 패턴은 하나의 인스턴스를 만들어 다른 모듈들이 해당 인스턴스를 공유하면서 사용합니다. 그렇기 때문에 인스턴스 생성할 때는 드는 비용이 줄어드는 장점이 있습니다. 반면 의존성이 높아진다는 단점도 존재합니다.

싱글톤 패턴은 mongoose 모듈에서 그 활용사례를 볼 수 있습니다. Node.js에서 MongoDB 데이터베이스를 연결할 때는 쓰는 mongoose 모듈에서 사용됩니다. 데이터 베이스를 연결할 때 사용하는 `connect()` 함수는 싱글톤 인스턴스를 반환합니다.

### 오버로딩과 오버라이딩의 차이점을 설명하시오.

오버로딩은 같은 이름을 가진 메서드를 메서드의 타입, 매개변수의 유형, 개수 등에 따라 여러 개 두는 것을 말합니다. 컴파일 중에 발생하는 정적 다형성입니다.

오버라이딩의 경우 오버라이딩은 주로 메서드 오버라이딩을 말하며, 상위 클래스로부터 상속받은 메서드를 하위 클래스가 재정의하는 것을 의미합니다. 이는 런타임 중에 발생하는 동적 다형성 입니다.

### TCP/IP 계층에 대해 설명해보시오

TCP/IP 계층은 애플리케이션 계층, 전송 계층, 인터넷 계층, 링크 계층 총 4개의 계층으로 이루어져 있습니다. 애플리케이션 계층은 FTP, HTTP 등 응용 프로그램이 사용되는 프로토콜 계층 입니다. 전송 계층은 애플리케이션과 인터넷 계층 사이의 데이터가 전달될 때의 중계 역할을 담당합니다. 인터넷 계층은 장치로부터 받은 네트워크 패킷을 IP 주소로 지정된 목적지로 전송하기 위해 사용되는 계층입니다. 링크계층은 실질적으로 데이터를 전달하고, 장치 간에 신호를 주고받는 규칙을 정하는 계층입니다.