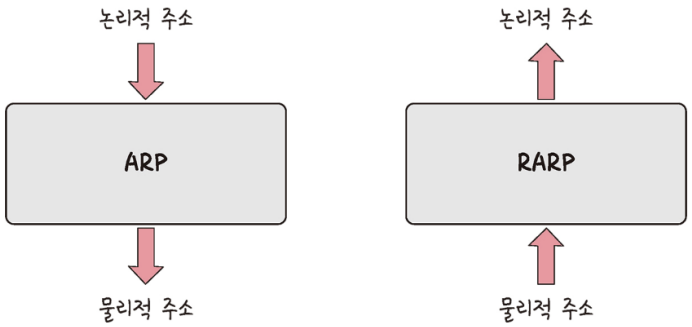
🟩 Chap 02 네트워크_2.4 IP 주소

2.4 IP 주소

2.4.1 ARP

✅ ARP란



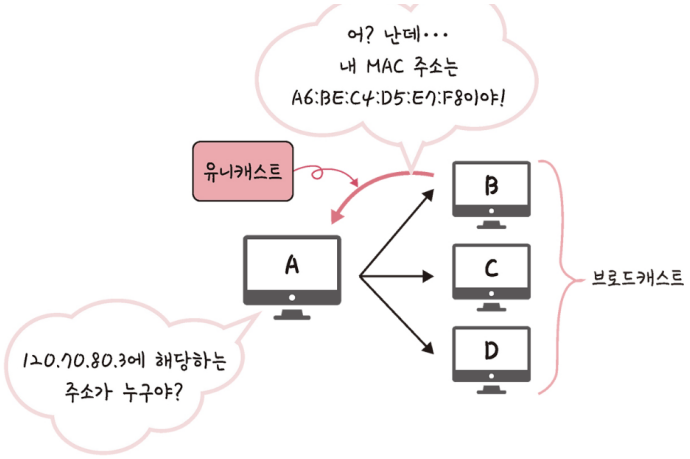
▲ 그림 2-43 ARP와 RARP

= ARP(Address Resolution Protocol)

= IP 주소로부터 MAC 주소를 구하는 IP와 MAC 주소의 다리 역할을 하는 프로토콜

= 가상 주소(IP)를 실제 주소(MAC)로 변환

≠ RARP = 실제 주소(MAC)를 가상 주소(IP)로 변환

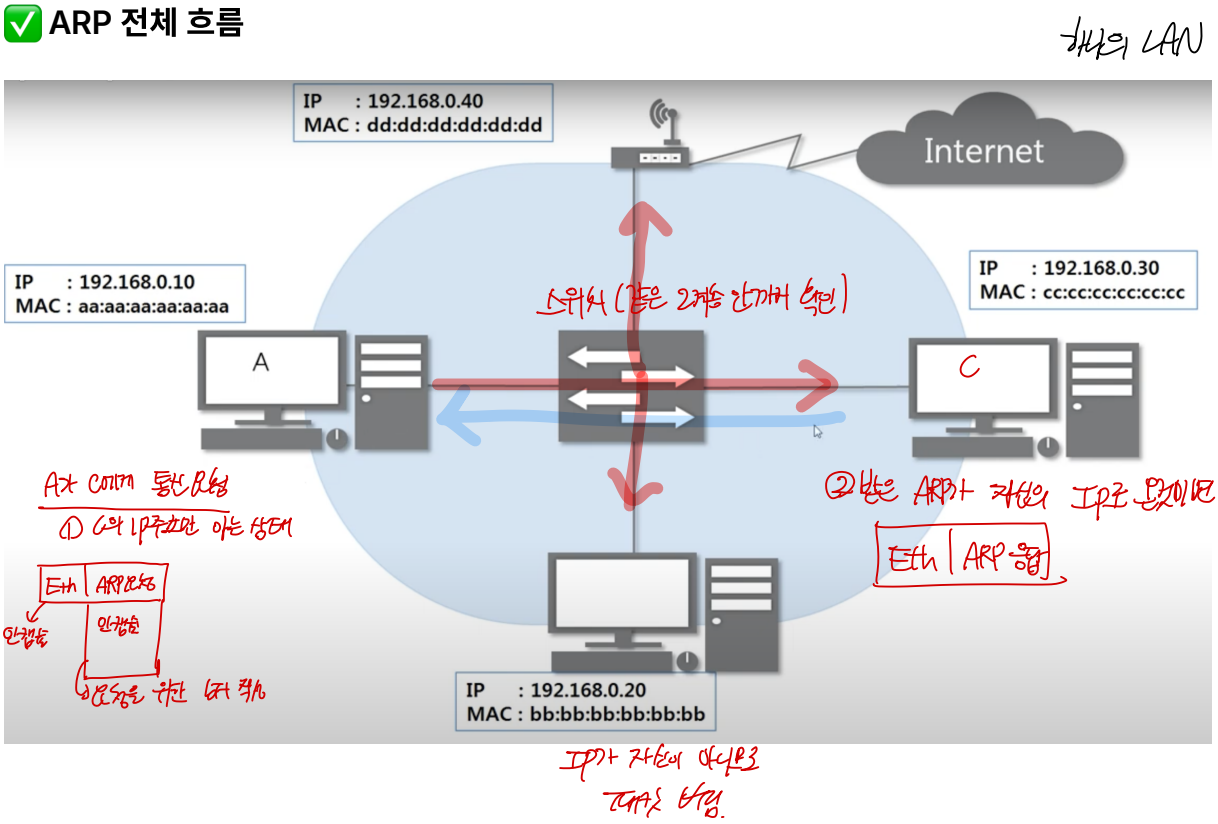


▲ 그림 2-44 ARP의 주소를 찾는 과정

*유니 캐스트 : 고유 주소로 하나의 네트워크 목적지에 1:1 통신 방식

*브로드 캐스트 : 송신 호스트가 전송한 데이터가 네트워크에 연결된 모든 호스트에 전송되는 방식

✅ ARP 전체 흐름



2.4.2 홉바이홉 통신

✅ 홉바이홉 통신이란

= IP 주소를 통해 통신하는 과정

> *hop이란?
> - 통신망에서 각 패킷이 여러 개의 라우터를 건너가는 모습을 표현
> - 통신 자체에 있는 '라우팅 테이블'의 IP를 통해 시작 주소부터 시작하여 다음 IP로 계속 해서 이동하는 '라우　팅' 과정을 거쳐 패킷이 최종 목적지까지 도달하는 통신
>
> *라우팅
> - IP 주소를 찾아가는 과정

✅ 라우팅 테이블

= 송신지에서 수신지까지 도달하기 위해 사용

= 라우터에 들어가 있는 목적지 정보들과 그 목적지로 가기 위한 방법이 들어 있는 리스트

= 게이트웨이와 모든 목적지에 대해 해당 목적지에 도달하기 위해 거쳐야 할 다음 라우터의 정보 가짐

✅ 게이트웨이

= 서로 다른 통신망, 프로토콜을 사용하는 네트워크 간의 통신을 가능하게 하는 관문 역할

= 서로 다른 네트워크상의 통신 프로토콜을 변환해주는 역할

= 라우팅 테이블을 통해 게이트웨이 확인 가능 `netstat -r`

```
C:\Users\jhc>netstat -r
===========================================================================
인터페이스 목록
 17...e0 d5 5e b3 7e 89 ......Intel(R) Ethernet Connection (7) I219-V
  8...f8 63 3f 72 c0 a9 ......Intel(R) Wireless-AC 9462
  4...f8 63 3f 72 c0 aa ......Microsoft Wi-Fi Direct Virtual Adapter
  2...fa 63 3f 72 c0 a9 ......Microsoft Wi-Fi Direct Virtual Adapter #2
  9...f8 63 3f 72 c0 ad ......Bluetooth Device (Personal Area Network)
  1...........................Software Loopback Interface 1
 45...00 15 5d 42 d9 de ......Hyper-V Virtual Ethernet Adapter
===========================================================================

IPv4 경로 테이블
===========================================================================
활성 경로:
네트워크 대상      네트워크 마스크     게이트웨이      인터페이스    메트릭
          0.0.0.0          0.0.0.0  121.165.151.254  121.165.151.200     25
    121.165.151.0    255.255.255.0             연결됨    121.165.151.200    281
  121.165.151.200  255.255.255.255             연결됨    121.165.151.200    281
  121.165.151.255  255.255.255.255             연결됨    121.165.151.200    281
        127.0.0.0        255.0.0.0             연결됨          127.0.0.1    331
        127.0.0.1  255.255.255.255             연결됨          127.0.0.1    331
  127.255.255.255  255.255.255.255             연결됨          127.0.0.1    331
     172.25.192.0    255.255.240.0             연결됨       172.25.192.1   5256
     172.25.192.1  255.255.255.255             연결됨       172.25.192.1   5256
   172.25.207.255  255.255.255.255             연결됨       172.25.192.1   5256
        224.0.0.0        240.0.0.0             연결됨          127.0.0.1    331
        224.0.0.0        240.0.0.0             연결됨    121.165.151.200    281
        224.0.0.0        240.0.0.0             연결됨       172.25.192.1   5256
  255.255.255.255  255.255.255.255             연결됨          127.0.0.1    331
  255.255.255.255  255.255.255.255             연결됨    121.165.151.200    281
  255.255.255.255  255.255.255.255             연결됨       172.25.192.1   5256
===========================================================================
영구 경로:
  없음

IPv6 경로 테이블
===========================================================================
활성 경로:
 IF  메트릭 네트워크 대상            게이트웨이
  1    331 ::1/128                  연결됨
 17    281 fe80::/64                연결됨
 45   5256 fe80::/64                연결됨
 17    281 fe80::453:e251:319f:36ff/128
                                    연결됨
 45   5256 fe80::3030:e4dc:c066:2087/128
                                    연결됨
  1    331 ff00::/8                 연결됨
 17    281 ff00::/8                 연결됨
 45   5256 ff00::/8                 연결됨
===========================================================================
영구 경로:
  없음
```

▲ 그림 2-47 netstat -r 명령어 구동 모습

2.4.3 IP 주소 체계

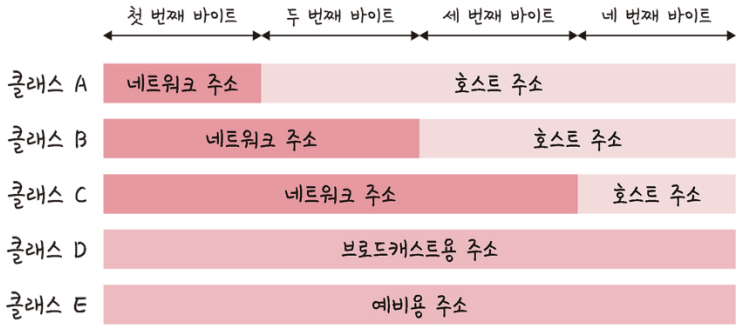
✅ IPv4, IPv6

	IPv4	IPv6
주소 크기	32비트 (8비트씩 4개 10진수 표시)	128비트 (16비트 8개 16진수 표시)
주소 공간	주소 공간 한정 → 고갈 문제 있음	넓은 주소 공간 제공 → 충분히 수용 가능
구성 방식	NAT 사용 → 여러 장치가 하나의 공용 IP 주소 공유	NAT 필요 없음 → 각 장치가 고유한 공용 IP 주소 가질 수 있음
헤더 구조	비교적 간단 → 확장성, 유연성 낮음 가변	복잡한 구조 → 확장성, 유연성 높음 고정
브로드캐스트	브로드캐스트 지원	없음 → 멀티캐스트, 애니캐스트 사용
자동 구성	DHCP를 통해 자동 IP 설정 가능	자동 구성 기능 내장 DHCP 없어도 자동 IP 설정 가능
보안	IPSec 프로토콜 별도 설치	IPSec 자체 지원
지원 및 호환성	현재 대부분 인터넷에서 사용	IPv4 상호 운용을 위한 이중 스택 or 터널링 같은 기술 필요

✅ 클래스 기반 할당 방식

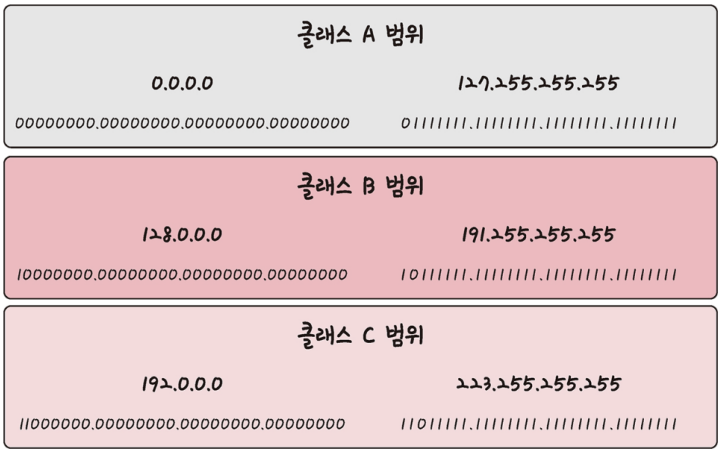
= A,B,C,D,E 5개의 클래스로 구분하는 방식

= 네트워크 주소 + 호스트 주소

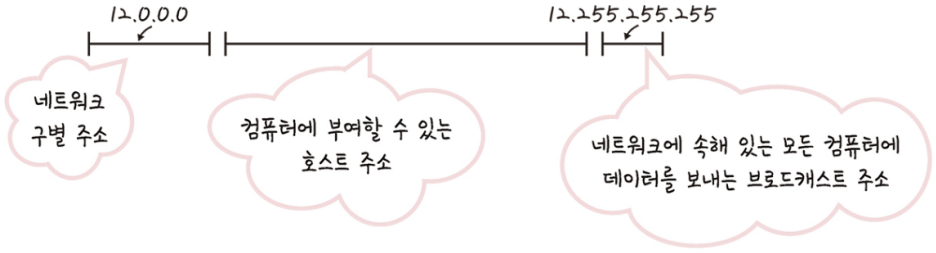


▲ 그림 2-49 클래스 기반 할당 방식

- 클래스 A,B,C : 일대일 통신
- 클래스 D : 멀티캐스트 통신
- 클래스 E : 앞으로 사용할 예비용



▲ 그림 2-50 클래스 기반 할당 방식 상세 내역



▲ 그림 2-51 네트워크 주소와 브로드캐스트용 주소

문제점 : 버리는 주소 많음

해결 : DHCP, IPv6, NAT 등장

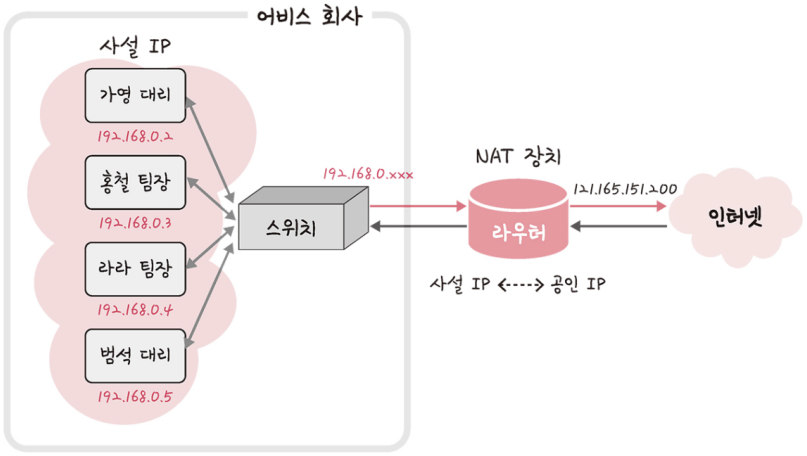
✅ DHCP

= Dynamic Host Configuration Protocol)

= IP 주소 및 기타 통신 매개변수를 자동으로 할당하기 위한 네트워크 관리 프로토콜

= 네트워크 IP 주소를 수동을 설정 필요 없이 인터넷 접속시 자동으로 IP 주소 할당

✅ NAT



= Network Address Translation

= 패킷이 라우팅 장치를 통해 전송되는 동안 패킷의 IP 주소 정보 수정 후 다른 주소로 매핑

= IP패킷의 TCP/UDP 포트 숫자와 소스 및 목적지의 IP등을 재기록 → 라우터를 통해 네트워크 트래픽을 주고 받는 기술

= 사설 네트워크에 속한 여러 개의 호스트가 하나의 공인 IP 주솔르 사용하여 인터넷에 접속하기 위함

→ NAT 장치를 통해 사설 IP를 공인 IP로 변환하거나 공인 IP를 사설 IP로 변환하는데 쓰임

공유기와 NAT

인터넷 공유기에는 NAT 기능 탑재

→ 여러 대의 호스트가 하나의 공인 IP 주소 사용하여 인터넷 접속 가능

→ 인터넷 공유기로 여러 PC 연결해 사용 가능

NAT를 이용한 보안

내부 IP와 외부 IP가 다르므로 기본적인 보안 가능

NAT 단점

여러명 동시 접속 → 실제 접속하는 호스트 숫자에 따라 접속 속도 느려질 수 있음

2.5 HTTP

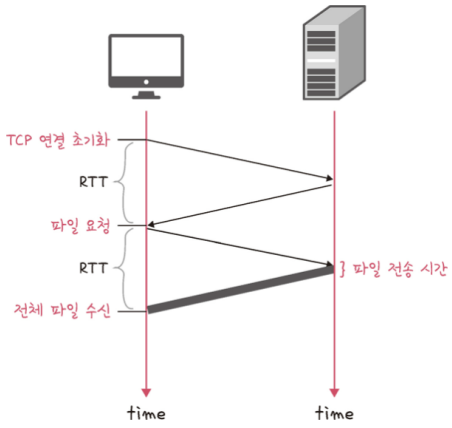
> HTTP는 애플리케이션 계층.
> 웹 서비스 통신에 사용
> HTTP/1.0 부터 시작, 현재는 HTTP/3

2.5.1 HTTP/1.0

= 하나의 연결당 하나의 요청 처리

→ RTT 증가

✅ RTT(Round Trip Time) 증가



▲ 그림 2-54 RTT 증가

= 패킷이 목적지에 도달하고 나서 다시 출발지로 돌아오기까지 걸리는 시간

= 패킷 왕복 시간

⇒ RTT 증가 → 서버 부담, 사용자 응답 시간 길어짐
⇒ 해결 방안 : 스플리팅, 코드 압축, 이미지 Base64 인코딩

✅ 해결1) 이미지 스플리팅

: 작은 아이콘들을 각각의 이미지 파일로 만들 경우 과부하 발생

→ 하나의 이미지 파이로 합쳐 사용

```
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;

public class ImageSplitting {
    public static void main(String[] args) throws Exception {
        File file = new File("large_image.jpg"); // 원본 이미지 파일 경로
        BufferedImage image = ImageIO.read(file);

        int rows = 4; // 행 수
        int cols = 4; // 열 수
        int chunks = rows * cols;

        int chunkWidth = image.getWidth() / cols;
        int chunkHeight = image.getHeight() / rows;

        int count = 0;
        BufferedImage[] imgs = new BufferedImage[chunks];
        for (int x = 0; x < rows; x++) {
            for (int y = 0; y < cols; y++) {
                imgs[count] = new BufferedImage(chunkWidth, chunkHeight, imag

                // 이미지 부분을 잘라서 배열에 저장
                int dx = chunkWidth * y;
                int dy = chunkHeight * x;
                for (int i = 0; i < chunkWidth; i++) {
                    for (int j = 0; j < chunkHeight; j++) {
                        imgs[count].setRGB(i, j, image.getRGB(dx + i, dy + j)
                    }
                }
                count++;
            }
        }

        // 분할된 이미지 저장
        for (int i = 0; i < imgs.length; i++) {
            ImageIO.write(imgs[i], "jpg", new File("img" + i + ".jpg"));
        }

        System.out.println("이미지 분할 완료!");
    }
}
```

✅ 해결2) 코드 압축

: 코드 압축을 통해 개행문자, 빈칸 없앰 → 코드의 용량 자체를 줄이는 방식

자바스크립트

```
const express = require('express')
const app = express()
const port = 3000
app.get('/', (req, res) => {
    res.send('Hello World!')
})

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
})
```

앞의 코드를 다음과 같은 코드로 바꾸는 방법입니다.

자바스크립트

```
const express=require("express"),app=express(),port=3e3;app.get("/",(e,p)=>{p.send("Hello World!")}),app.listen(3e3,()=>{console.log("Example app listening on port 3000")});
```

✅ 해결3) 이미지 Base64 인코딩

: 이미지 파일을 64진법으로 이루어진 문자열로 인코딩하는 방법

: [장점] 서버와의 연결을 열고 이미지에 대해 서버에 HTTP 요청 필요 없음

: [단점] Base64 문자열로 변환할 경우 37%정도 크기가 커짐