

# 데이터통신과 네트워킹

Data Communication  
& Networking Ch. 12



## CHAPTER

---

# 12

---

## 네트워크 계층 작업과 프로토콜

---

### Section

- 01 네트워크 분할
- 02 주소 변환 관련 프로토콜
- 03 기타 네트워크 계층 프로토콜

## 1. 서브넷 마스크의 특징

- 사물 인터넷 때문에 IP 주소는 빠른 속도로 고갈.
- 조직의 규모에 따라 C 클래스의 256개는 너무 작고, B 클래스의 65535개는 너무 큼 -> 이러한 문제를 해결하는 방법은 하나의 클래스를 서브넷(하부넷)으로 잘라 여러 개로 나누어 사용하는 것 -> 서브넷으로 분할해서 사용할 때 서브넷 마스크가 중요한 역할.
- 서브넷 마스크에서 1로 표시된 부분은 네트워크 주소, 0으로 표시된 부분은 호스트 주소를 의미.

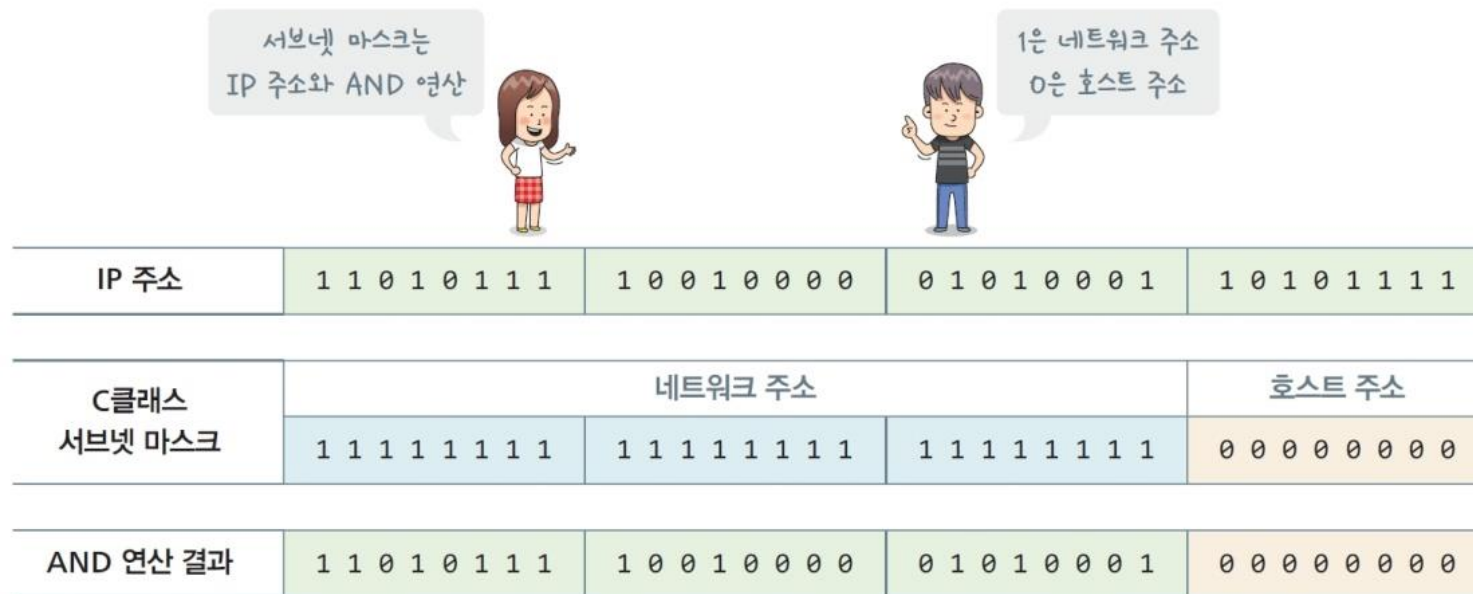


그림 12-1 C 클래스 서브넷 마스크 연산 과정

# 네트워크 분할

- 서브넷 마스크의 특성상 앞에서부터 1이 연달아 나타남.
- IP 주소 옆에 1의 개수를 써주면 서브넷 마스크를 간단하게 표현 할 수 있음.
- C클래스의 서브넷 마스크는 1이 24개이다. IP 주소 228.152.23.8이 C 클래스라면 228.152.23.8/24로 표시 -> 서브넷 마스크의 1이 앞에서부터 24개 있다는 의미.
- B클래스의 서브넷 마스크는 /16, A클래스의 서브넷 마스크는 /8



그림 12-2 IP 주소와 서브넷 마스크 동시 표시 방법

## 2. 네트워크 분할 방법

- C 클래스는 원래 256개의 호스트를 가질 수 있는 네트워크 -> 228.255.75로 시작하는 C 클래스를 A 회사와 B 회사에게 반씩 나누어 준다고 가정.
- A 회사는 IP 주소의 마지막 자리인 4번째 자리(호스트 주소)는 이진수로 00000000에서 00000001, 00000010 .... 01111111까지 사용.
- 호스트 주소의 맨 앞 비트가 0으로 시작하는 주소가 A 회사에 속하는 주소. 십진수로는 228.255.75.0에서 228.255.75.127까지의 주소이며 128개의 호스트 주소를 사용할 수 있음.

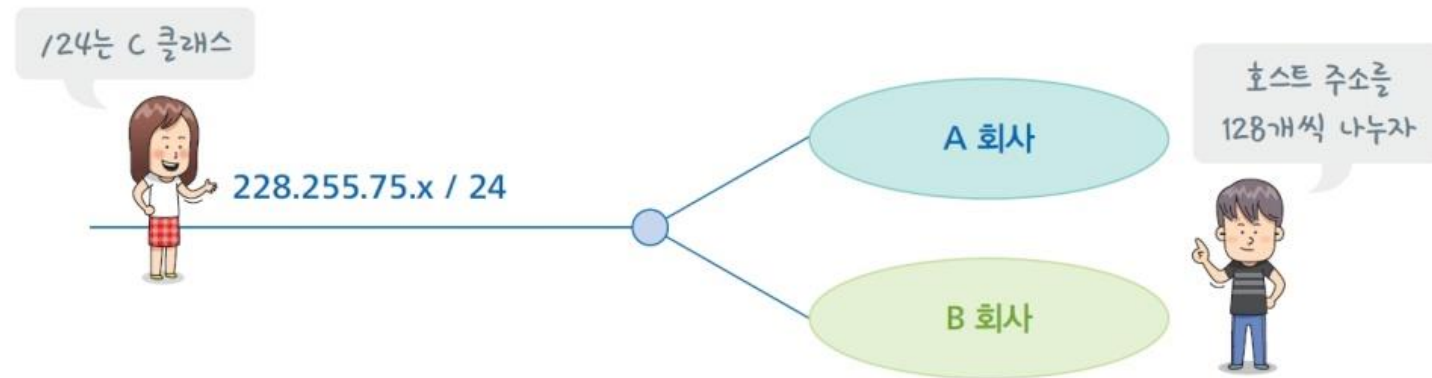



그림 12-3 C 클래스 네트워크 분할


# 네트워크 분할

- B 회사는 2진수 10000000에서 10000001, 10000010 .... 11111111까지의 호스트 주소를 사용.
- 호스트 주소의 맨 앞 비트가 1로 시작하는 주소가 B 회사에 속하는 주소이며 전체 128개의 호스트 주소를 사용. 십진수로는 228.255.75.128에서 228.255.75.255까지의 주소.
- A 회사의 호스트 주소는 2진수로 00000000에서 00000001, 00000010 .... 01111111와 같이 맨 앞의 첫 비트가 0으로 시작, B 회사의 호스트 주소는 2진수로 10000000에서 10000001, 10000010 .... 11111111와 같이 맨 앞의 첫 비트가 1로 시작.

호스트 0에서 127까지는  
A 회사가 사용  
호스트 맨 앞 비트는 0



호스트 128에서 255까지는  
B 회사가 사용  
호스트 맨 앞 비트는 1



	1번째 바이트	2번째 바이트	3번째 바이트	4번째 바이트	10진수
A 회사	228	255	75	0 0 0 0 0 0 0 0	0
	228	255	75	0 1 1 1 1 1 1 1	127
B 회사	228	255	75	1 0 0 0 0 0 0 0	28
	228	255	75	1 1 1 1 1 1 1 1	255

그림 12-4 C 클래스를 2개의 서브넷으로 나누는 경우



# 네트워크 분할

- 2개로 분할시 서브넷 마스크는 2진수로 11111111 11111111 11111111 10000000이 되어야 함.
- 4번째 바이트의 서브넷 마스크가 00000000이 아닌 10000000으로 변경됨 -> 이는 4번째 바이트의 맨 처음 비트도 네트워크 주소라는 뜻.
- 2진수 10000000를 10진수로 바꾸면 128 -> 따라서 C 클래스를 2개의 서브넷으로 나눈 경우 서브넷 마스크는 255.255.255.128이 됨. 간단하게 표시하면 /25가 됨.



그림 12-5 C 클래스를 2개 나눈 경우의 서브넷 마스크

## 3. 서브넷 분할 원리

- 네트워크 주소를 나타내는 1의 개수가 늘어날 때 마다 2의 지수승으로 네트워크는 분할.
- C클래스를 2개의 서브넷으로 분할하면 서브넷 마스크는 11111111 11111111 11111111 10000000이 됨. 10진수로 255.255.255.128이며 /25라 표시.
- C 클래스 네트워크를 4개로 분할: 서브넷 마스크가 255.255.255.192. 이는 /26으로 표시.

네트워크 주소를 나타내는  
1의 개수가 증가할 때마다  
2의 지수승으로 분할



1개의 서브넷이  
가질 수 있는 호스트 수는  
2의 지수승으로 감소



1번째 바이트	2번째 바이트	3번째 바이트	4번째 바이트							10 진수	간단 표시	비고	1개 서브넷 최대 호스트수
11111111	11111111	11111111	0	0	0	0	0	0	0	0	/24	서브넷 분할 없음	256
11111111	11111111	11111111	1	0	0	0	0	0	0	128	/25	2(2 <sup>1</sup> )개로 분할	128
11111111	11111111	11111111	1	1	0	0	0	0	0	192	/26	4(2 <sup>2</sup> )개로 분할	64
11111111	11111111	11111111	1	1	1	0	0	0	0	224	/27	8(2 <sup>3</sup> )개로 분할	32
11111111	11111111	11111111	1	1	1	1	0	0	0	240	/28	16(2 <sup>4</sup> )개로 분할	16

그림 12-6 C 클래스 분할과 서브넷 마스크



# 네트워크 분할

- B 클래스의 서브넷 마스크는 255.255.0.0이며 /16.
- B 클래스를 2개로 나누면 서브넷 마스크는 255.255.128.0이며 /17 -> 1개의 서브넷이 가질 수 있는 최대 호스트 수는 32768개.
- 서브넷 마스크가 255.255.224.0(/19)인 네트워크는 B 클래스를 8개로 분할한 것 -> 1개의 서브넷이 가질 수 있는 최대 호스트 개수는 8192개.

1번째 바이트	2번째 바이트	3번째 바이트						4번째 바이트	10 진수	간단 표시	비고	1개 서브넷 최대 호스트수
11111111	11111111	0	0	0	0	0	0	0	0	/16	서브넷 분할 없음	65536
11111111	11111111	1	0	0	0	0	0	0	128	/17	2개로 분할	32768
11111111	11111111	1	1	0	0	0	0	0	192	/18	4개로 분할	16384
11111111	11111111	1	1	1	0	0	0	0	224	/19	8개로 분할	8192
11111111	11111111	1	1	1	1	0	0	0	240	/20	16개로 분할	4096

그림 12-7 B 클래스 분할과 서브넷 마스크

# 네트워크 분할

- 네트워크를 분할 할 때 한 번에는 2의 배수로만 분할 할 수 있다고 해서 네트워크가 짝수로만 만들어지는 것은 아님.
- 네트워크에서 분할을 여러 번 사용하면 홀수개의 서브넷도 만들 수 있음.

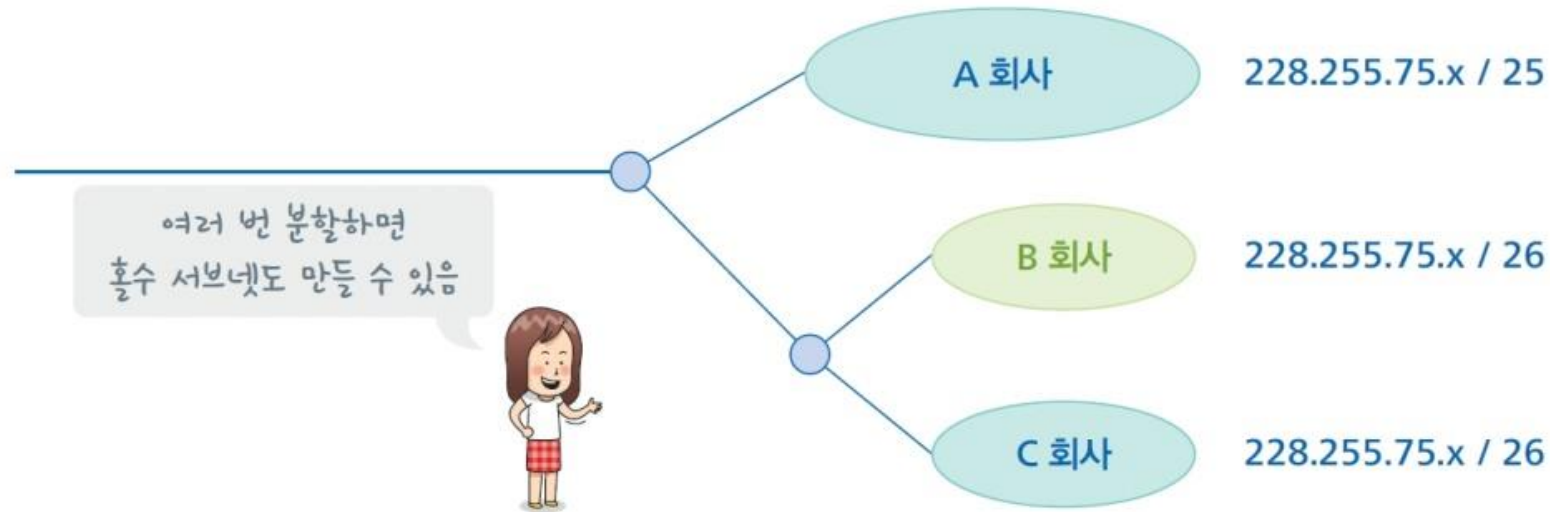
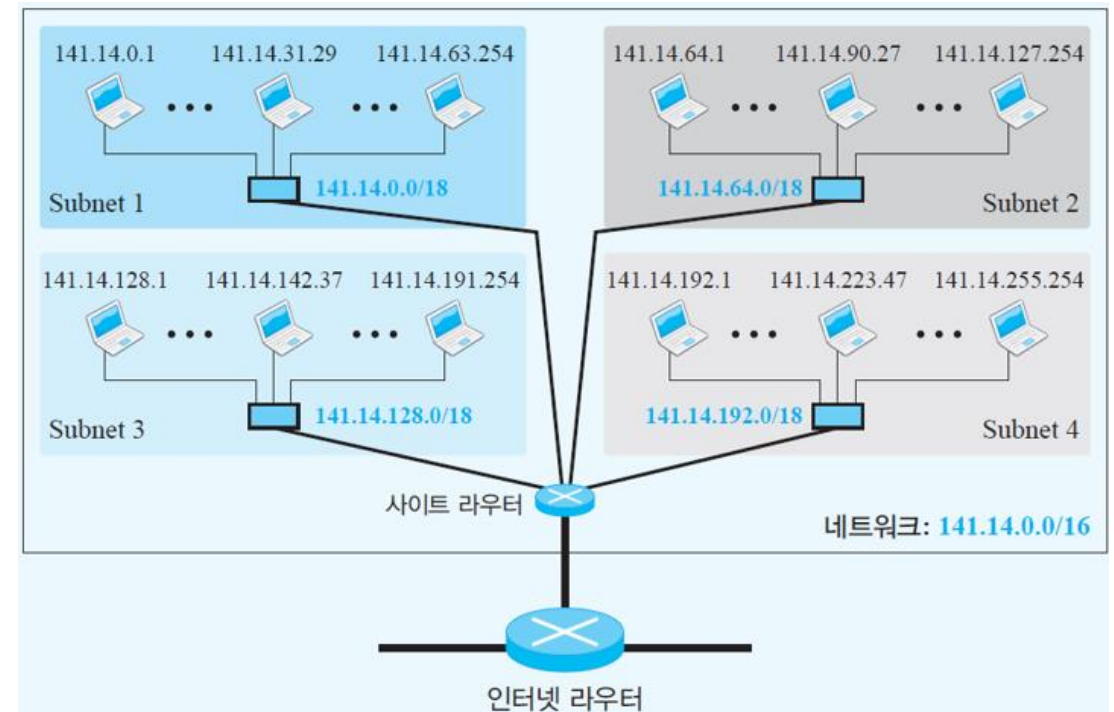
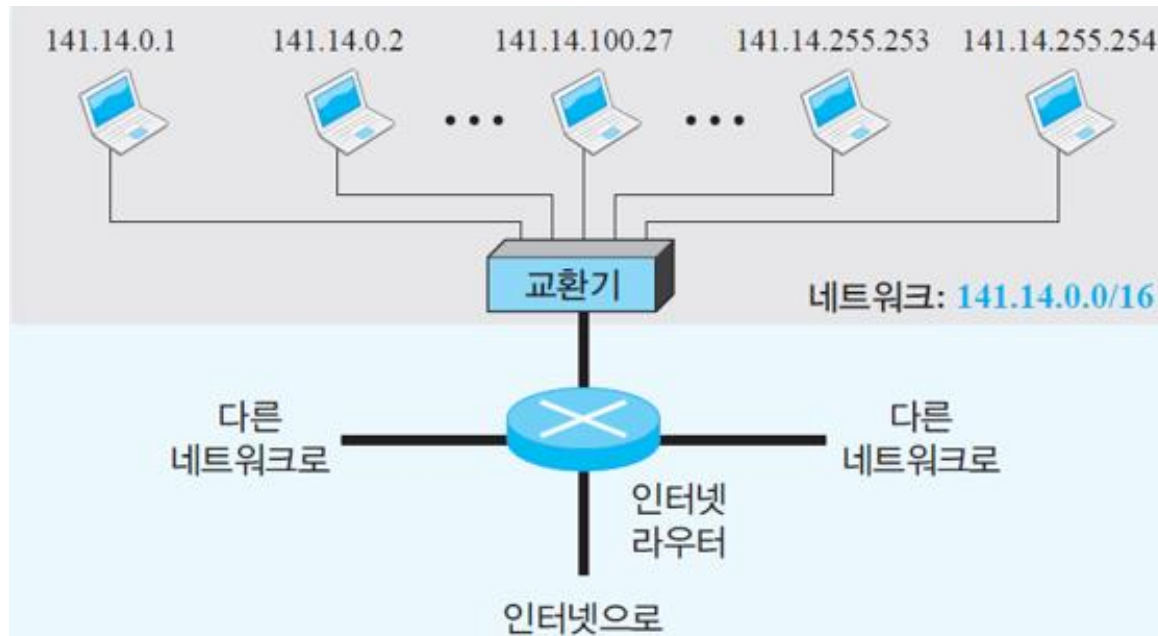


그림 12-8 홀수 개로 구성된 네트워크

# 클래스 기반 네트워크 분할 예제

## 예제

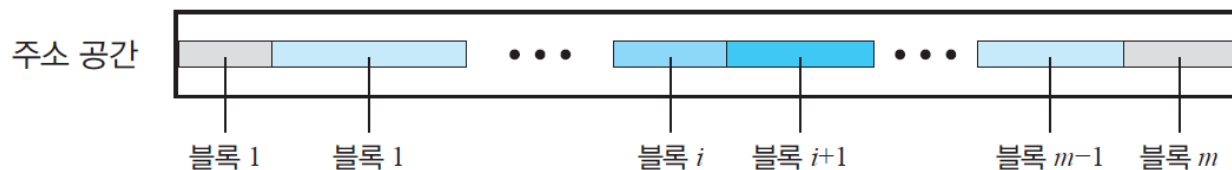
- 왼쪽 그림의 네트워크를 서브네팅한 네트워크를 보여준다. 전체 네트워크는 여전히 같은 네트워크를 통하여 연결되어 있다.
- 네트워크를 사설 라우터를 이용하여 4개의 서브네트워크로 나누었다.
  - 인터넷에서 여전히 하나의 네트워크로 보이지만 내부적으로 네트워크는 4개의 서브네트워크로 되어있다.
  - 각 서브네트워크는  $2^{14}$  개의 호스트를 가질 수 있다. 네트워크는 4개의 서로 다른 학부(빌딩)을 가진 대학 캠퍼스에 속할 수 있다.
  - 서브네팅 후에, 각 학부는 자신의 서브네트워크를 갖지만, 여전히 전체 캠퍼스는 인터넷에서 하나의 네트워크이다. 그림에서 /16과 /18은 netid와 subnetid의 길이를 나타낸다.



# 클래스 없는 주소 지정

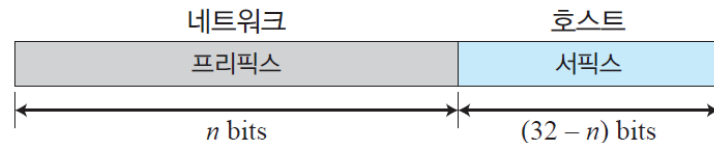
- 클래스 없는 주소 지정
  - 클래스 기반 주소 지정에서 서브네팅과 슈퍼네팅은 실제로 주소 고갈 문제를 해결하지 못함
  - 인터넷이 성장함에 따라, 장기적으로 보다 큰 주소 공간을 확보하는 것이 필요
  - 장기적 해결책으로서 IPv6가 개발
  - 그러나 동일한 주소 공간을 사용하면서 각 기관에 대해 주소를 분배하는 방법이 필요.
  - 단기 해결책은 IPv4 주소를 사용하면서 클래스 없는 주소지정(classless addressing)이다.
- 가변길이 블록
  - 각 기관은 가변길이 블록으로  $2^1, 2^2, 2^3, 2^4, 2^5 \dots 2^{32}$ 개의 주소를 갖는 블록을 지정 가능

그림 4.25 클래스 없는 주소지정 방식에서 가변 길이 블록



- 2단계 주소 체계
  - 프리픽스(Prefix): netid와 동일 기능
  - 서픽스(suffix): hostid와 동일 기능
  - 클래스 없는 주소 지정 방식에서 프리픽스의 길이는 0 ~ 32

그림 4.26 프리픽스와 서픽스



# 클래스 없는 주소 지정

- 네트워크 마스크
  - 클래스 기반 주소지정 방식에서 정의된 것과 같은 개념
  - 네트워크 마스크는 32비트 숫자
  - 왼쪽부터  $n$ 개의 연속적인 비트는 1로 설정
  - 그 나머지 비트는 0으로 설정
- 다음은 슬래시 표현으로 나타낸 주소
  - A. 주소 12.23.24.78/8에서, 네트워크 마스크는 255.0.0.0이다. 즉, 8개의 1과 24개의 0으로 구성. 프리픽스 길이는 8이고, 서픽스 길이는 24이다. (A 클래스와 동일)
  - B. 주소 130.11.232.156/16에서, 네트워크 마스크는 255.255.0.0이다. 즉, 16개의 1과 16개의 0으로 구성. 프리픽스 길이는 16이고, 서픽스 길이는 16이다. (B 클래스와 동일)
  - C. 주소 167.199.170.82/27에서, 네트워크 마스크는 255.255.255.224이다. 즉, 27개의 1과 5개의 0으로 구성. 프리픽스 길이는 27이고, 서픽스 길이는 5이다. (C 클래스와 동일)
- 예제
  - ISP가 1,000개의 주소를 갖는 블록을 요청하였다. 블록은 다음과 같다.
  - 1,000은 2의 거듭제곱이 아니기 때문에, 1,024개의 주소가 할당된다( $1,024 = 2^{10}$ )
  - 블록의 프리픽스 길이는  $n = 32 - \log_2 1,024 = 22$ 이다.
  - 18.14.12.0 (1,024로 나누어짐)가 선택된다. 할당된 블록은 18.14.12.0/22이다. 첫 번째 주소는 18.14.12.0/22 이고, 마지막 주소는 18.14.15.255/22이다. (세 번째 바이트가 12~15, 즉 4개를 할당)

# 클래스 없는 주소 지정

- 서브네팡
  - 3단계 계층은 서브네팡을 이용하여 구성
  - 하나의 블록을 할당 받은 기관(또는 ISP)은 블록을 여러 개의 서브-블록으로 나눈 후에 각 서브-블록을 서브네팡에 할당
  - 서브네팡은 여러 개의 서브-서브네팡으로 나뉘어질 수 있다
  - 서브-서브네팡은 다시 서브-서브-서브네팡으로 나뉘어질 수 있다.
- 서브네팡 설계
  - 기관이 할당 받은 주소의 총 개수는  $N$ 이고, 프리픽스 길이는  $n$ 이며, 각 서브네팡에게 할당된 주소의 개수는  $N_{\text{sub}}$ , 각 서브네팡의 프리픽스 길이는  $n_{\text{sub}}$ , 그리고 서브네팡의 총 개수는  $s$ 라고 하자.
  - 각 서브네팡에 속하는 주소의 개수는 2의 거듭제곱이어야 한다.
  - 각 서브네팡을 위한 프리픽스 길이는 다음과 같이 구한다.

$$n_{\text{sub}} = n + \log_2(N/N_{\text{sub}})$$

- 각 서브네팡에 속하는 시작 주소는 해당 서브네팡에 속하는 주소의 개수로 나누어 질 수 있어야 한다.
- 먼저 큰 네트워크에 주소를 설정하고 크기가 작은 네트워크의 주소는 그 다음에 설정하는 등의 단계로 주소 설정



# 클래스 없는 주소 지정

- 예제 1
  - 기관이 블록 130.34.12.64/26을 할당 되었다. 기관은 각각 동일한 개수의 호스트를 갖는 4개의 서브넷을 구성하고자 한다. 서브넷을 설계하고 각 서브넷에 대한 정보를 구하여라.
  - 전체 네트워크에 대한 주소의 개수는  $N = 2^{32-26} = 64$ 이다. 네트워크의 첫 번째 주소는 130.34.12.64/26 이고 마지막 주소는 130.34.12.127/26이다. 이제 서브넷을 설계해 보자:
  - 첫번째 조건을 만족하도록 각 서브넷에서 16개의 주소를 할당한다.(16은 2의 거듭 제곱).
  - 각 서브넷에 대한 서브넷 마스크는

$$n_1 = n_2 = n_3 = n_4 = n + \log_2(N/N_i) = 26 + \log_2 4 = 28$$

- 각 서브넷은 첫 번째 주소부터 시작해서 16개의 주소를 할당한다.

# 클래스 없는 주소 지정

- 예제 2
  - ISP가 190.100.0.0/16(65,536개의 주소)로 시작하는 주소 블록을 할당 받았다. ISP는 이 주소를 다음과 같이 세 그룹의 고객에게 분배하고자 한다:
    - 첫 번째 그룹은 64개의 고객; 각각은 256개 정도의 주소가 필요
    - 두 번째 그룹은 128개의 고객; 각각은 128개 정도의 주소가 필요
    - 세 번째 그룹은 128개의 고객; 각각은 64개 정도의 주소가 필요
    - 서브-블록을 설계하라, 각 서브-블록에게 주소를 할당한 후에 얼마나 많은 수의 주소가 이용 가능한가?
  - 두 단계로 문제를 해결해 보자.
    - 첫 번째 단계에서, 주소의 서브블록을 각 그룹에 할당한다. 각 그룹에 할당된 주소의 전체 개수와 각 서브블록에 대한 프리픽스 길이는 다음과 같다.

$$\text{그룹 1 : } 64 \times 256 = 16,384$$

$$n_1 = 16 + \log_2(65536/16384) = 18$$

$$\text{그룹 2 : } 128 \times 128 = 16,384$$

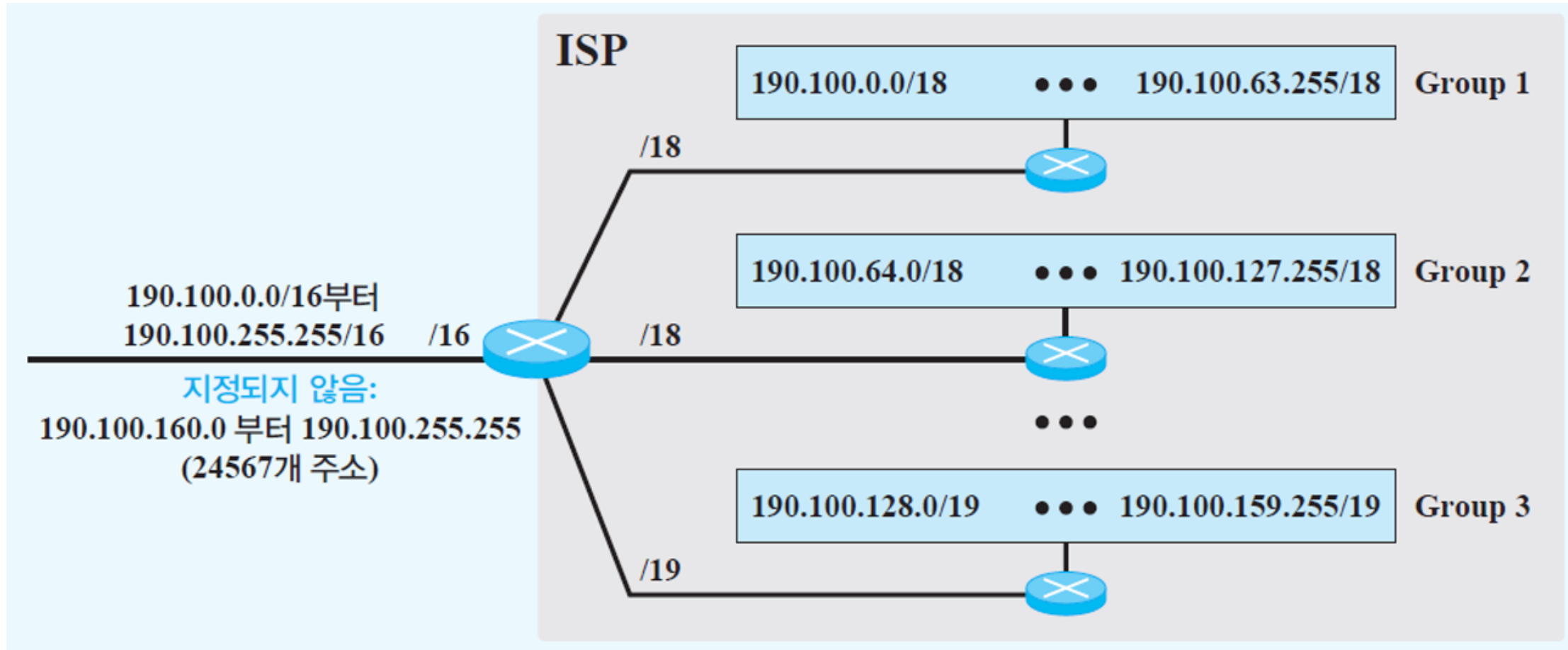
$$n_2 = 16 + \log_2(65536/16384) = 18$$

$$\text{그룹 3 : } 128 \times 64 = 8192$$

$$n_3 = 16 + \log_2(65536/8192) = 19$$

# 클래스 없는 주소 지정

- 첫 번째 단계의 설계



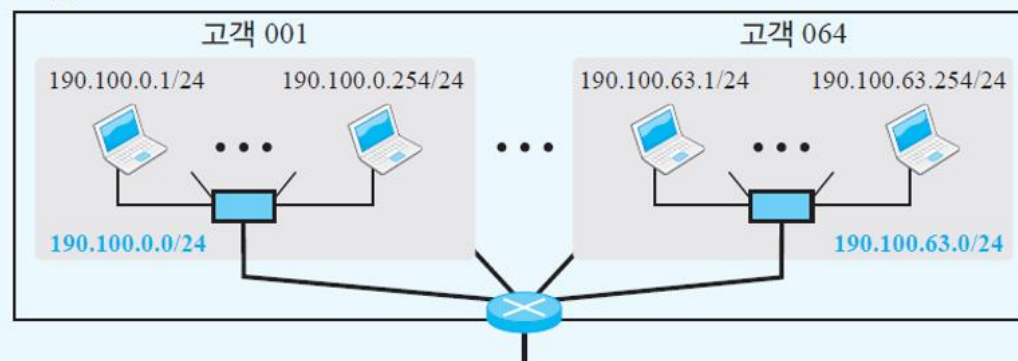
# 클래스 없는 주소 지정

- 두 번째 단계에서, 각 고객에 대한 첫 번째 주소는 서브넷 주소로 사용하고, 마지막 주소는 특수 목적 주소로 예비용으로 둔다.

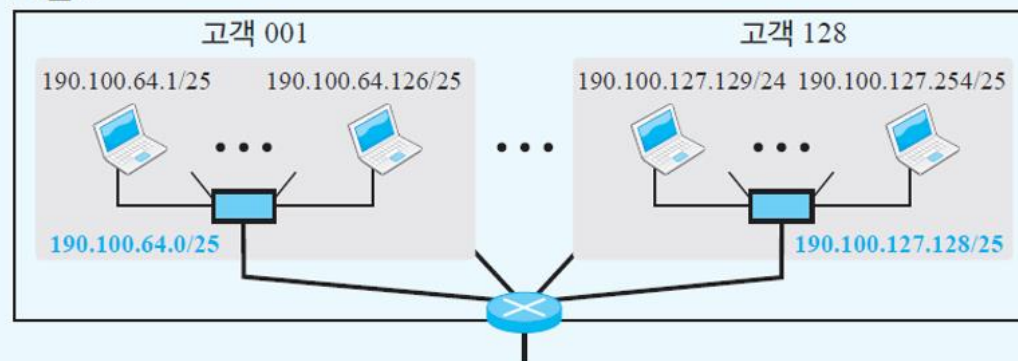
그룹:  $n = 18$ :

서브넷:  $n = 18 + \log_2 (16385/256) = 24$

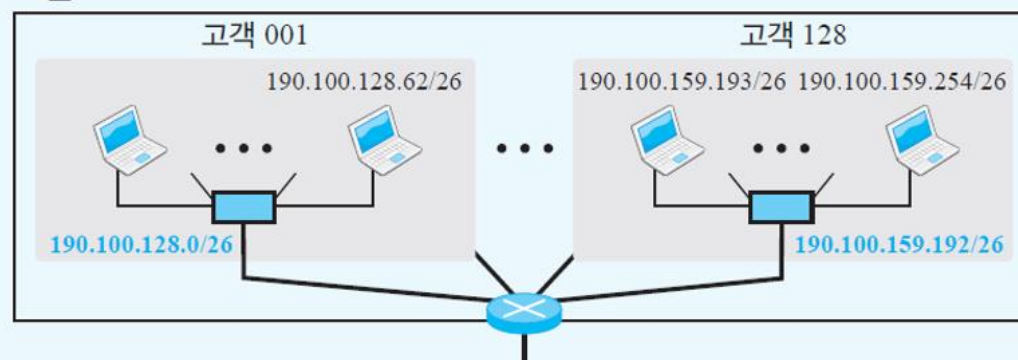
그룹 1



그룹 2



그룹 3



그룹:  $n = 19$

서브넷:  $n = 19 + \log_2 (8192/64) = 26$

# 주소 변환 관련 프로토콜

## 1. 주소 변환의 필요성

- 228.255.75.1에서 228.255.75.4까지의 호스트가 있는 어떤 C 클래스 네트워크를 가정.
- 228.255.75.2가 228.255.75.3에게 패킷을 보내려면 해당 호스트의 MAC 주소가 필요.
- IP 주소는 인터넷에서 목적지 LAN까지 도달하기 위해 필요한 주소 -> 도착한 LAN의 안에서 호스트를 찾는다면, IP 주소로 LAN 내부에서 통신을 하는 경우에는 MAC 주소가 필요.

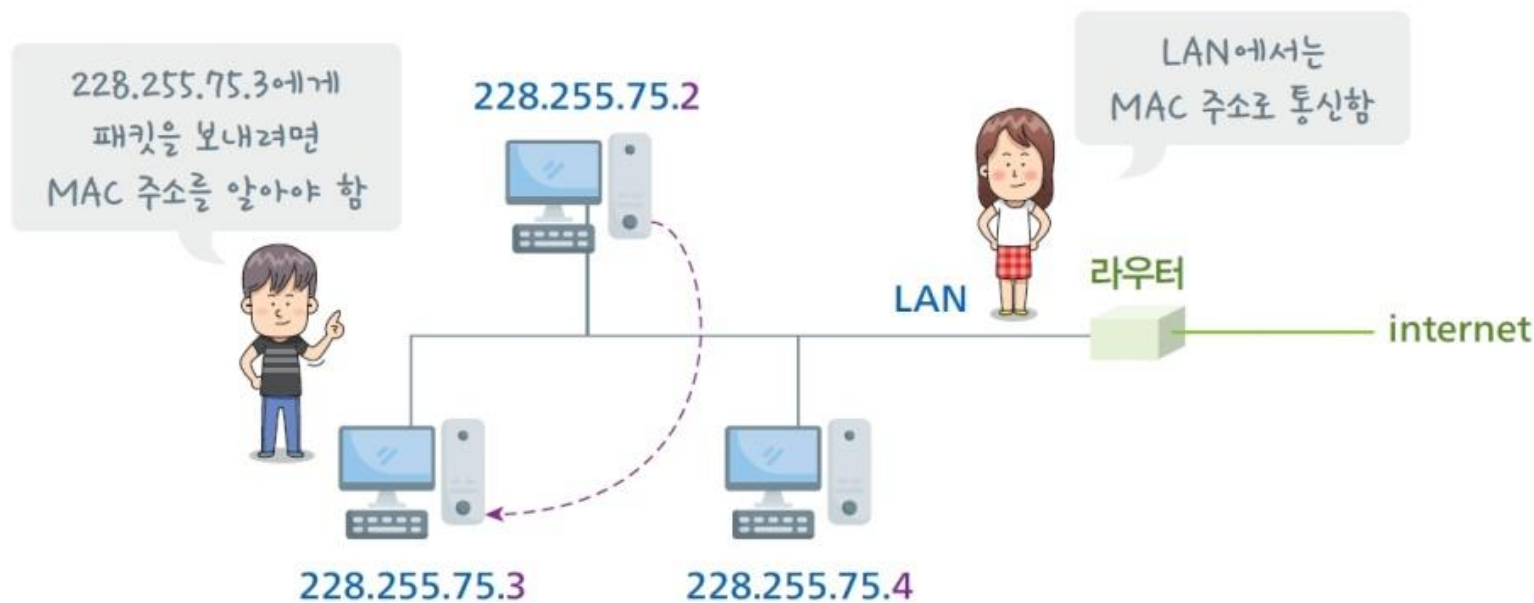


그림 12-9 IP 주소와 MAC 주소가 동시에 필요한 이유

# 주소 변환 관련 프로토콜

## 2. 주소 변환 프로토콜 - ARP

- 주소변환 서버 없이도 IP 주소를 MAC 주소로 바꾸는 프로토콜이 **ARP** Address Resolution Protocol
- 특정 IP의 MAC 주소를 알고 싶은 호스트는 다른 호스트들에서 질문 -> 해당 질문은 LAN 전체에 브로드캐스팅 -> 질문을 받은 호스트 중 해당 IP를 가진 노드는 자신의 MAC 주소를 알려줌.
- ARP의 성능을 높이기 위하여 각 호스트들은 획득한 ARP 정보를 캐시<sup>cache</sup>에 보관.

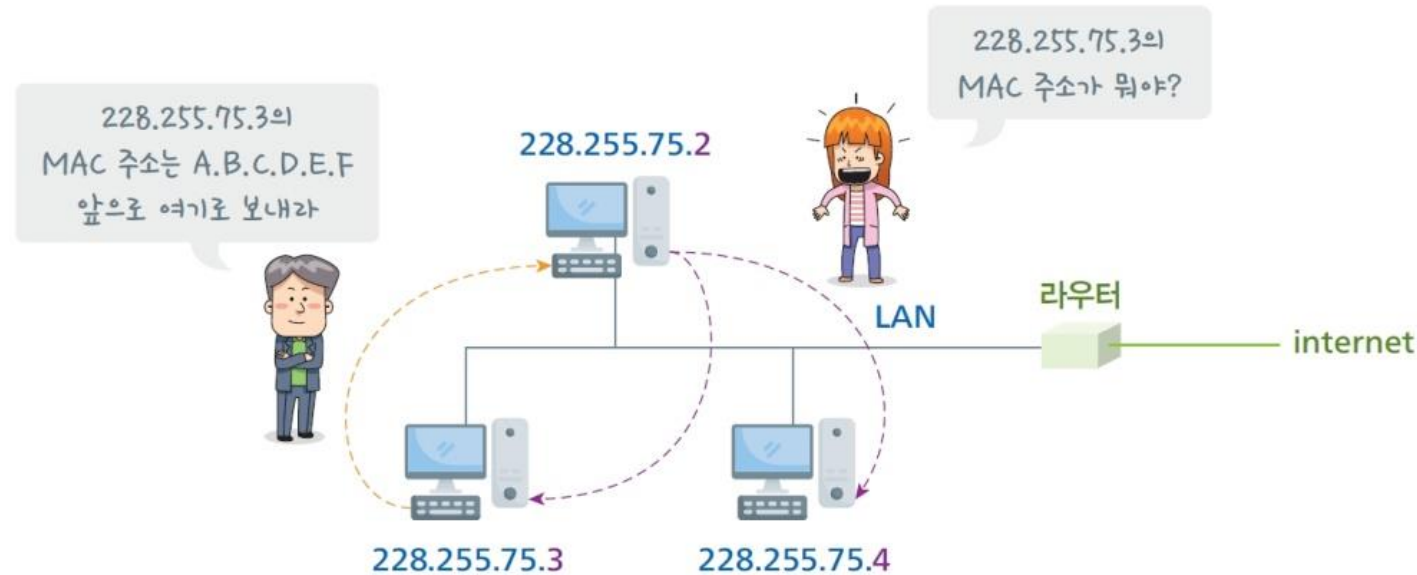


그림 12-10 ARP(Address Resolution Protocol) 작동 방법



# 주소 변환 관련 프로토콜

- ARP가 가지고 있는 캐시 데이터를 보고 싶다면, 윈도우 명령 프롬프트(cmd)에서 arp -a라고 검색.
- 해당 컴퓨터는 IP 주소 192.168.0.1, 192.168.0.5, 192.168.0.6에 대한 MAC 주소를 캐시에 저장.
- 192.168.0.255는 브로드캐스트 주소이며, 물리주소는 항상 ff-ff-ff-ff-ff-ff로 나타남.

인터페이스: 192.168.0.12 --- 0x13		
인터넷 주소	물리적 주소	유형
192.168.0.1	58-86-94-15-12-8f	동적
192.168.0.5	b4-2e-99-db-fb-80	동적
192.168.0.6	4a-2e-38-f8-63-83	동적
192.168.0.255	ff-ff-ff-ff-ff-ff	정적

그림 12-11 윈도우 cmd에서 arp -a 결과 화면

# 주소 변환 관련 프로토콜

- 기존의 컴퓨터가 사용하던 IP를 228.255.75.7로 바꿀 경우 -> IP 충돌이 발생한 경우, 충돌을 일으킨 양쪽의 컴퓨터에 "IP 충돌이 발생하였습니다."와 같은 경고 메시지가 나타남.
- 충돌이 발생한 경우, 해당 IP를 먼저 사용하고 있던 호스트에게 우선권이 주어짐.
- 충돌이 없다는 것을 확인한 이후에 자신의 IP 주소와 MAC 주소를 전체에게 브로드캐스트 함 -> 이때 사용하는 ARP를 Gratuitous ARP, 약자로 GARP라 부름.
- GARP는 응답이 필요없는 ARP.

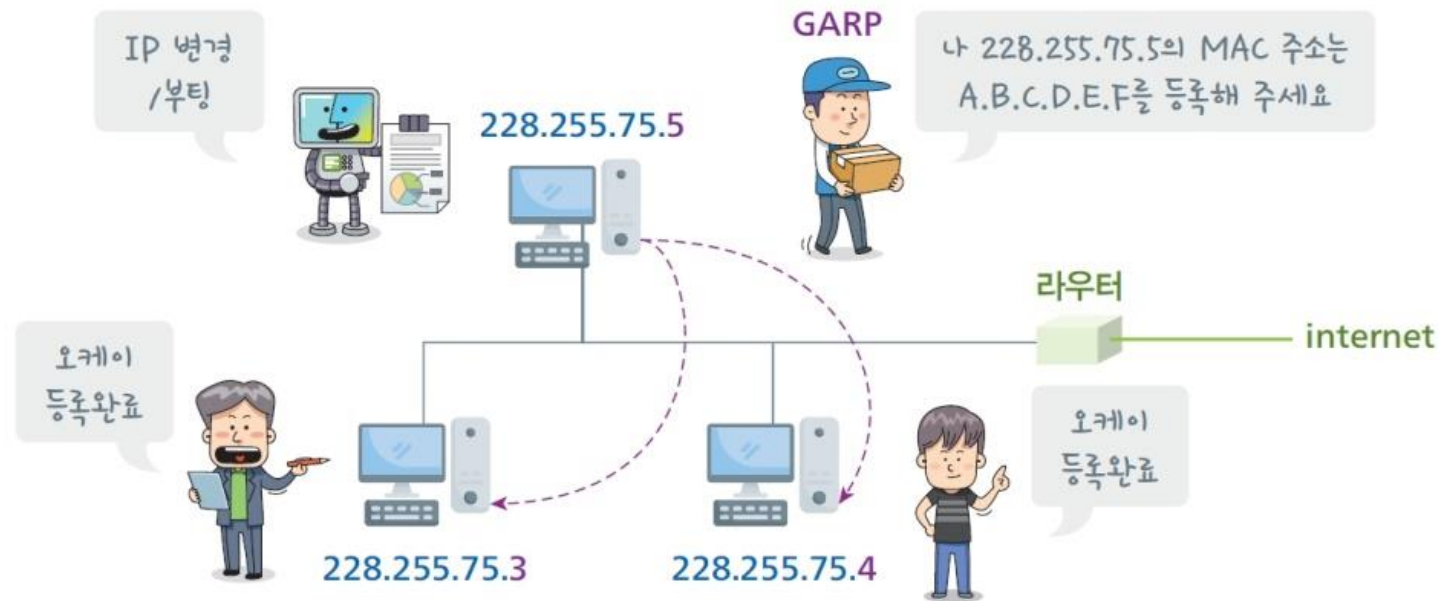


그림 12-12 Gratuitous ARP(GARP) 작동 방식

# 주소 변환 관련 프로토콜

## 3. 역주소 변환 프로토콜 – RARP 와 BOOTP

- 연산능력이나 저장장치가 없는 더미dummy 터미널도 IP 주소가 필요.
- 이런 단말기가 IP 주소를 얻기 위해 사용하는 프로토콜이 RARP이며, Reverse ARP의 약자.
- RARP는 MAC 주소로부터 IP 주소를 얻는 프로토콜.

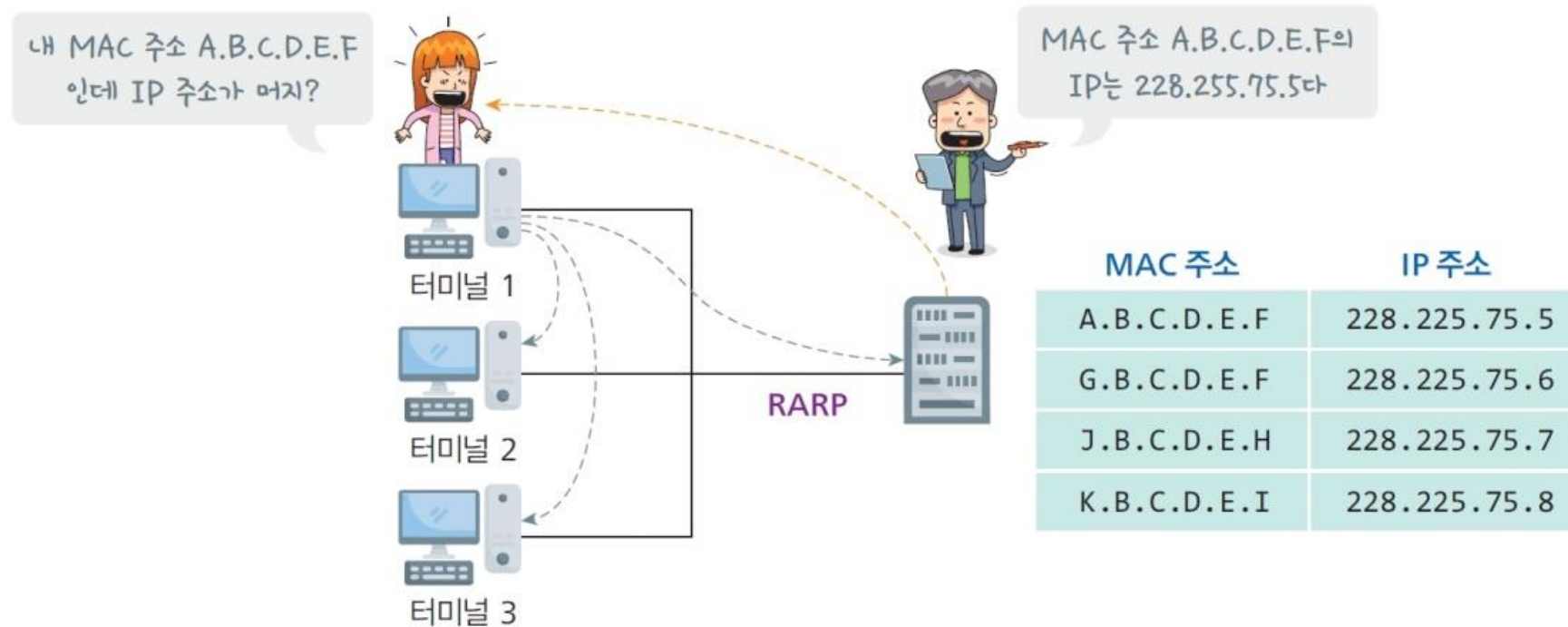


그림 12-13 Reverse ARP(RARP) 작동 방식

# 주소 변환 관련 프로토콜

- RARP의 단점은 브로드캐스트 메시지가 LAN에만 전파된다는 것
- RARP는 LAN에서만 사용할 수 있는 프로토콜이기 때문에 라우터를 넘어서 메시지를 전송할 수 없음.
- 라우터를 넘어서 주소변환을 할 수 있도록 만든 것이 부트스트랩 프로토콜, BOOTP<sup>bootstrap protocol</sup>
- BOOTP는 UDP라고 불리는 전송계층의 프로토콜을 사용한다. 그래서 라우터를 넘어서 메시지가 전달될 수 있음.

# 주소 변환 관련 프로토콜

## 4. 유동 호스트 설정 프로토콜 - DHCP

- 고정 IP 방식의 문제: 모든 호스트의 IP 주소와 관련 값을 일일이 수동으로 설정해야 한다는 것과 IP 주소 관리.
- 유동 IP 방식에서는 컴퓨터가 부팅할 때 IP 주소, 서브넷 마스크, 게이트웨이, DNS 주소를 자동으로 설정 -> 이때 사용하는 프로토콜이 DHCP(Dynamic Host Configuration Protocol)
- DHCP의 작동 방식은 BOOTP와 유사 -> 메시지를 전체에게 브로드캐스트 하면 DHCP 서버가 사용하지 않는 주소와 서브넷 마스크, 게이트웨이 주소, DNS 주소를 해당 컴퓨터에게 전달.

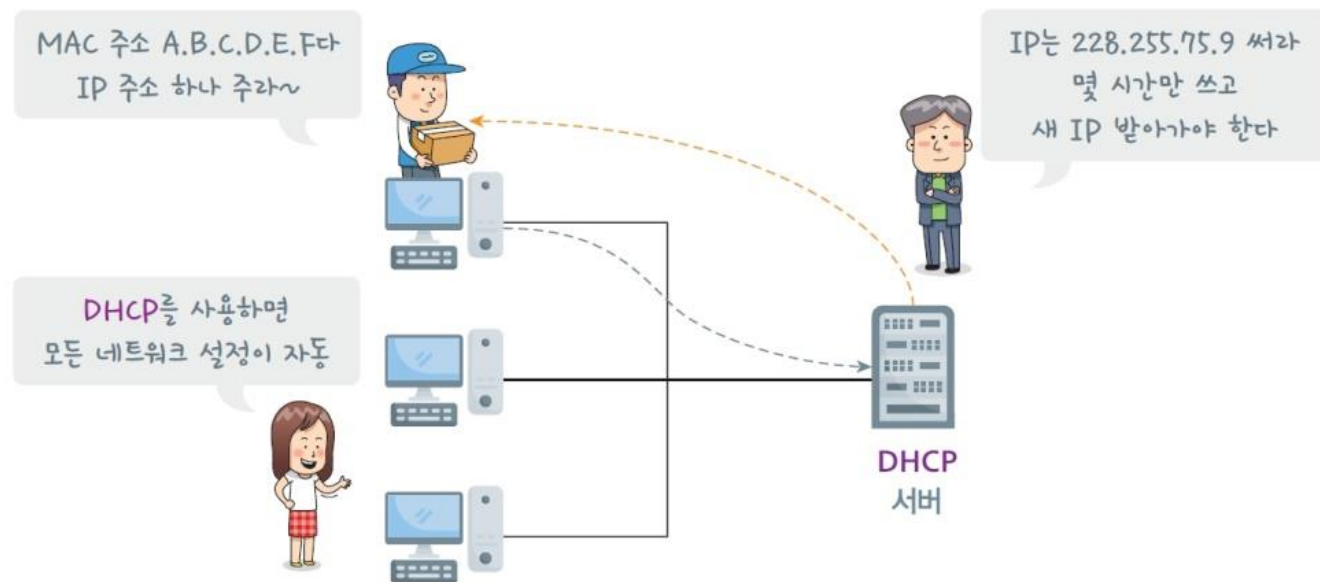


그림 12-14 DHCP(Dynamic Host Configuration Protocol) 작동 방식

# 주소 변환 관련 프로토콜

- 주소관련 프로토콜의 특징 요약

표 12-1 주소관련 프로토콜 특징 요약

프로토콜	특징
ARP Address Resolution Protocol	IP 주소 → MAC 주소 변환
GARP Gratuitous ARP	IP 주소, MAC 주소 등록
RARP Reverse ARP	MAC 주소 → IP 주소 변환(다른 LAN 불가)
BOOTP bootstrap protocol	MAC 주소 → IP 주소 변환(다른 LAN 가능)
DHCP Dynamic Host Configuration Protocol	MAC 주소 → IP 주소 변환 (자동 IP 할당 / 주기적 IP 변경)



## 1. 네트워크 관리 프로토콜

- 감시와 메시지 전송에 사용되는 프로토콜이 ICMP(Internet Control Message Protocol)
- 다음 표는 ICMP의 주요 메시지 타입과 의미를 요약한 것.

표 12-2 ICMP 주요 메시지

Type	의미	설명
0	Echo reply	Echo 응답(ping 응답)
3	destination unreachable	도달 불가
5	redirect	경로 변경
8	Echo request	Echo 요청(ping 요청)
11	time exceeded	TTL 초과
30	trace route	라우팅 경로 추적

# 기타 네트워크 계층 프로토콜

- ICMP를 직접 경험할 수 있는 방법은 ping을 사용하는 것 -> ping 명령어 다음에 IP 주소를 쓰면 해당 호스트로 패킷을 보내고, 해당 호스트는 ping에 대한 답신<sup>replay</sup> 메시지를 보냄.
- ping 명령어에서 호스트까지 메시지를 보내는 것이 ICMP 8번 Echo request -> Echo request를 받은 호스트는 이에 대한 응답으로 ICMP 0번 Echo replay를 보냄.
- ping을 사용하면 해당 호스트가 살아있는지 혹은 죽었는지를 알 수 있을 뿐 아니라, 서버까지 패킷이 오고가는데 걸리는 시간도 알 수 있음.

```
C:\User\zoch>ping 117.52.93.2
```

```
Ping 117.52.93.2 32바이트 데이터 사용:
```

```
117.52.93.2의 응답: 바이트=32 시간=5ms TTL=51
```

```
117.52.93.2의 응답: 바이트=32 시간=5ms TTL=51
```

```
117.52.93.2의 응답: 바이트=32 시간=5ms TTL=51
```

```
117.52.93.2의 응답: 바이트=32 시간=5ms TTL=51
```

```
117.52.93.2에 대한 Ping 통계:
```

```
패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
```

```
왕복 시간(밀리초):
```

```
최소 = 4ms, 최대 = 8ms, 평균 = 5ms
```

그림 12-15 ping 사용 예

# 기타 네트워크 계층 프로토콜

- ping과 유사하지만 tracert는 해당 호스트까지 가는 경로를 추적하는 명령어.
- tracert 명령은 패킷이 어떤 라우터를 거쳐서 목적지 까지 도달하는지에 대한 경로를 알려줌 -> 이때 사용하는 것이 ICMP 30번 trace route.

```
C:\User\zoch>tracert 117.52.93.2
```

최대 30홉 이상의  
slrclub.com [117.52.93.2](으)로 가는 경로 추적:

1	2 ms	<1 ms	<1 ms	192.168.0.1
2	2 ms	1 ms	1 ms	203.252.21.4
3	3 ms	2 ms	2 ms	172.16.101.3
4	16 ms	9 ms	6 ms	203.252.23.251
5	4 ms	3 ms	2 ms	61.98.80.213
6	3 ms	4 ms	5 ms	10.67.253.106
7	6 ms	3 ms	3 ms	10.222.22.240
8	7 ms	5 ms	7 ms	10.222.30.66
9	5 ms	5 ms	3 ms	10.222.32.115
10	6 ms	5 ms	5 ms	118.130.13.65
11	8 ms	5 ms	6 ms	1.208.165.50
12	4 ms	5 ms	4 ms	182.162.152.106
13	6 ms	6 ms	6 ms	110.45.163.90
14	5 ms	6 ms	7 ms	10.70.10.2
15	5 ms	4 ms	5 ms	slrclub.com [117.52.93.2]

추적을 완료했습니다.

- ICMP는 다양한 곳에 사용.
  - 3번 destination unreachable은 패킷이 목적지 주소를 찾지 못할 때 발생하는 메시지.
  - 11번 time exceeded는 패킷의 TTL이 0이 되는 경우 발생하는 메시지. IP 헤더 분석에서 설명하였지만, TTL이 0이 된 패킷은 폐기됨. 따라서 11번 time exceeded 메시지를 받았다는 것은 패킷이 루프를 돌거나 TTL 값이 너무 작게 설정 되어 있다는 의미.
  - 5번의 redirect는 패킷의 경로가 잘못 설정 되었다고 판단될 때 보내는 메시지. 패킷을 보낸 호스트의 경로 설정에 문제가 있다고 판단되는 경우에 발생하는 메시지.

# 기타 네트워크 계층 프로토콜

- 멀티캐스팅에 사용되는 프로토콜이 IGMP(Internet Group Message Protocol)
- 각 채널에는 해당 채널을 시청중인 호스트의 주소가 그룹으로 묶여 있음 -> 채널을 시청하려는 호스트는 IGMP join 메시지를 이용하여 채널 그룹에 등록 -> A는 IGMP join 7번 채널, B는 IGMP join 8번 채널.
- 그러다가 A가 7번에서 8번 채널로 옮길 때, IGMP leave 메시지를 사용하여 7번 채널 그룹에서 빠지고, IGMP join 8번 채널을 사용하여 8번 채널 그룹에 다시 등록.



그림 12-17 IGMP(Internet Group Message Protocol) 작동 방식

## 내부 라우팅 - OSPF

- OSPF는 Open Shortest Path First의 약자로 최단경로 프로토콜이며 내부 라우팅 프로토콜 -> OSPF가 연결상태 라우팅 알고리즘을 기반으로 만들어 짐.
- OSPF의 작동방식은 앞의 연결상태 라우팅 알고리즘에서 자세히 설명하였음.
- 다음 표는 OSPF가 사용하는 메시지 타입과 설명.

표 12-3 OSPF 메시지 타입

타입	설명
Hello	이웃 라우터를 찾을 때 사용
Link state update	네트워크 시간(거리) 정보를 보낸
Link state ACK	Link state update에 대한 ACK
Database description	발신자의 테이블 개수를 알려줌
Link state request	이웃에게 정보 요청



# 기타 네트워크 계층 프로토콜

- 연결상태 라우팅 알고리즘의 초기에는 이웃 라우터의 정보를 수집 -> 이때 사용하는 메시지가 Hello.
- 이웃 라우터의 정보가 모이면, 라우터들은 주기적으로 자신의 테이블을 전체 라우터들에게 플러딩 -> 이때 사용하는 메시지가 Link state update.
- Link state update를 받으면 이에 대한 응답을 해주어야 하는데 이것이 Link state ACK.
- 많은 수의 라우터와 ACK를 주고받기 때문에 연결상태 라우팅 알고리즘의 테이블에는 SEND 플래그와 ACK 플래그를 유지 -> 이는 받은 메시지와 보낸 메시지를 기억 했다가 못 보낸 메시지가 있으면 빠른 시간안에 응답하기 위해서임.
- Database description 메시지는 링크상태 테이블의 수를 알려달라는 메시지. 테이블의 수를 비교하여 자기가 가지고 테이블의 정보와 같은지 혹은 다른지를 확인. 만약 자신의 테이블과 다르다면 Link state request를 보내어 다른 라우터의 테이블을 받아 비교.

# 기타 네트워크 계층 프로토콜

- OSPF의 장점은 전체 네트워크 구조를 가지고 있어서 네트워크 상태 변화에 빠르게 적응 할 수 있다는 것과 거리벡터 라우팅과 같이 무한루프가 발생하지 않는다는 것.
- OSPF의 단점은 최단경로 알고리즘을 사용하기 때문에 CPU 소모가 많고, 전체 네트워크 구조를 가지고 있기 때문에 메모리를 많이 사용한다는 것.

# 기타 네트워크 계층 프로토콜

## 외부 라우팅 - BGP

- BGP는 Border Gateway Protocol의 약자로 외부 라우팅에 사용되는 프로토콜.
- 외부 라우팅에서는 시간(거리)정보는 중요하지 않음 -> 그 보다는 어느 네트워크에 연결 가능한지가 가장 중요한 문제 -> 그래서 단순한 BGP를 사용.
- BGP는 거리벡터 라우팅 알고리즘과 유사하지만 다른 점이 있음.
  - 시간(거리)에 대한 처리는 이루어지지 않음. 오직 통신 가능성만을 고려함.
  - 라우팅 정보에는 목적지 라우터 정보만 유지함.
- 무한 루프의 문제는 다른 라우터로부터 잘못된 거리 정보를 받을 때 발생 -> BGP가 시간(거리)를 계산하지 않기 때문에 거리벡터 라우팅의 문제였던 무한 루프(무한 숫자 세기)의 문제는 발생하지 않음.
- BGP에서는 해당 네트워크가 살아 있는지 혹은 죽었는지에 대한 값만을 받기 때문에 특정 라우터로부터 응답이 없는 경우, 즉시 다른 경로를 찾음.