



# NETWORK

IPv4 Classful, Classless Address에서의  
Subnetting, 게이트웨이를 거쳐 NAT까지

한시현



# Contents


0. 시작하기 앞서
1. Classful Addressing (Subnetting)
2. Classless Addressing (Subnetting)
3. Private Network
4. Gateway
5. NAT (network address translation)



시작하기 앞서..  
IP 주소란?



# Internet Protocol Version 4

- 인터넷에 연결된 각 장치를 구분하기 위해 TCP/IP 프로토콜의 IP Layer (Network Layer)에서 사용하는 식별자를 IP Address(인터넷 주소)라고 한다.
  - IP 주소는 Unique(유일해야 한다)하고, Universal(주소의 형식이 정해져 있어야 한다)하다.
  - IPv4는 32 bit 주소이다.
- 

# 주소 공간

- IPv4같은 프로토콜은 주소 공간을 가지고 있다.
- 주소 공간이란 프로토콜에서 사용되는 주소의 총 갯수이다.
- bit는 0과 1로 이루어져 있으므로, 32 bit의 주소를 가지는 IPv의 주소 공간(만들 수 있는 주소의 갯수)은  $2^{32}$  (4,294,967,296)개이다.
- 공간이 유한하므로 사용자의 수가 계속 증가한다면 언젠가는 이 주소 공간이 고갈 될 텐데, 이를 방지하기 위한 방법들을 나중에 볼 것이다.

# 표기법

- 주소 표기법으로는 2진수로 표현하는 2진 표기법과 10진수로 표현하는 점 10진 표기법이 잘 알려져 있다.
- 아래 둘은 동일한 주소를 다르게 표기한 것이다.
- 2진 표기법  
10000001 00001011 00001011 11101111
- 점 10진 표기법  
129.11.11.239
- 2진수를 10진수로, 10진수를 2진수로 바꾸는 계산을 통해 자유롭게 계산할 수 있다.
- .을 기준으로 나뉜 공간 하나당 8비트를 부여하면 된다.

# 주의사항

- 111.56.**0**45.78 : 다음과 같이 0이 앞에 오면 안된다.
- 75.45.**301**.14 : 8bit를 점 10진 표기법으로 최대 표현 가능한 수는 255이다.
- 221.34.7.8.**20** : 4개의 덩어리만 가능하다.
- **11100010**.23.14.67 : 혼용해서 사용하면 안된다.

# 여담

- 16진수로 나타내는 16진 표기법이라는 방법도 있다.
- 이는 4비트씩 끊어서 계산한다.
- 10000001 00001011 00001011 11101111
- -> 0x810B0BEF (0x는 16진수임을 나타내는 기호)



# IP 주소의 구조

- IP 주소의 구조에 대해 이야기해보자. 여기에는 2가지가 있다.
- Classful addressing (클래스 기반 주소 지정)
- Classless addressing (클래스 없는 주소 지정)
- IP 주소는 초기에는 클래스라는 개념을 사용했으나, 이후에 클래스가 없는 주소 지정이라는 것이 나오며 원래의 구조를 대체하였다.
- 우선 우리는 클래스 기반 주소 지정에 대해 살펴보겠다.

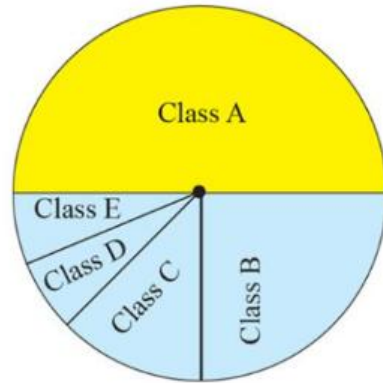


# Classful Addressing

# Classful addressing(클래스 기반 주소 지정)

- 클래스 기반 주소 지정이란 사용할 수 있는 양(크기)을 정해놓았다는 것이다.
- Classful addressing에서는 IP 주소 공간을 5개의 클래스 (A,B,C,D,E)로 나눈다.

# 클래스의 주소 공간 점유율



Class A:  $2^{31} = 2,147,483,648$  addresses, 50%

Class B:  $2^{30} = 1,073,741,824$  addresses, 25%

Class C:  $2^{29} = 536,870,912$  addresses, 12.5%

Class D:  $2^{28} = 268,435,456$  addresses, 6.25%

Class E:  $2^{28} = 268,435,456$  addresses, 6.25%

- 전체 주소 공간:  $2^{32}$
- Class A :  $2^{31}$ (전체의 절반)
- Class B :  $2^{30}$  (전체의 4분의 1, A의 절반)
- Class C :  $2^{29}$  (전체의 8분의 1, B의 절반)
- Class D :  $2^{28}$  (전체의 16분의 1, C의 절반)
- Class E :  $2^{28}$  (전체의 16분의 1, C의 절반)

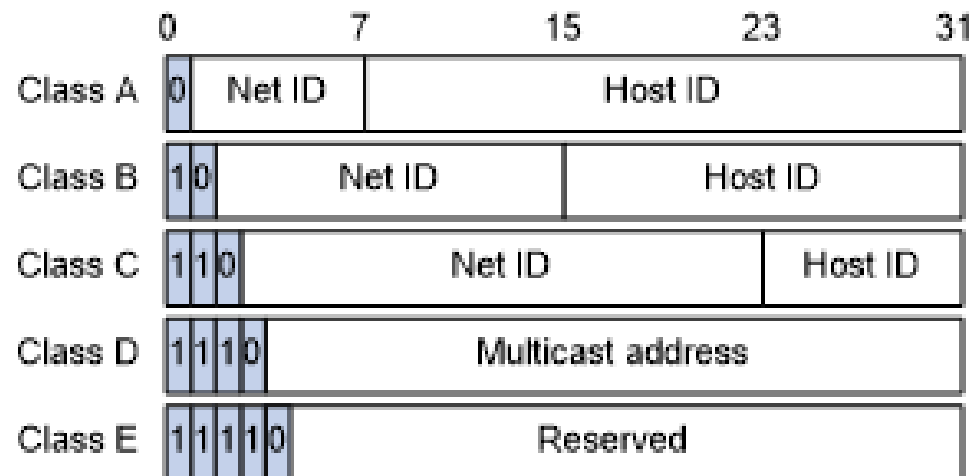
# 그래서, 주소의 클래스를 어떻게 찾는데?

	Octet 1	Octet 2	Octet 3	Octet 4		Byte 1	Byte 2	Byte 3	Byte 4
Class A	0.....				Class A	0-127			
Class B	10.....				Class B	128-191			
Class C	110.....				Class C	192-223			
Class D	1110....				Class D	224-239			
Class E	1111....				Class E	240-255			
	Binary notation					Dotted-decimal notation			

- 첫 번째 비트를 보면 된다.
- Class A : 0~으로 시작 (0-127)
- Class B : 10~으로 시작 (128-191)
- Class C : 110~으로 시작 (192-223)
- Class D : 1110~으로 시작 (224-239)
- Class E : 1111~으로 시작 (240-255)

# Netid & Hostid

- Classful addressing에서 클래스 A,B,C의 IP 주소는 netid와 hostid로 나뉜다.
- Netid : 네트워크의 ID(식별자)
- Hostid : 네트워크에 연결된 기기의 ID(식별자)
- 이들의 길이는 클래스에 따라 다른데, 이는 아래와 같다.

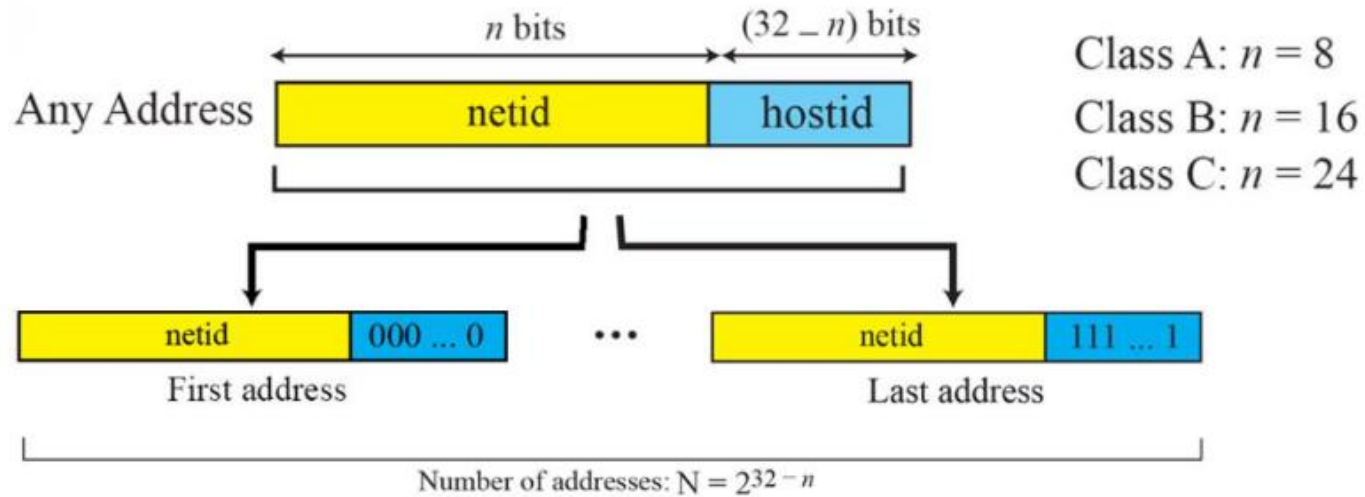


클래스 D와 E의 경우는  
멀티캐스팅과 예약된 주소(특수 목적)로  
사용하기 위해 설계된 클래스이다.

# Class & Block

- 일반적으로 netid에 포함된 network address가 같은 address를 블록(Block)이라고 한다.
- Class A의 첫 번째 byte인 0.~.~.~ 블록과 마지막 byte인 127.~.~.~ 블록은 Special block이다.
- 첫 주소 : ~.0.0.0 (Network Address, 네트워크라는 것을 알려주는 주소)
- 끝 주소 : ~.255.255.255 (Special, 이는 예약된 자리이다.)
- Class D의 경우, 멀티캐스팅을 하기 위해 설계되었고, 오직 1개의 블록만을 갖는다.
- Class E의 경우, 예약된 주소로 사용하기 위해 설계되었고, 역시 1개의 블록만을 갖는다.

# Classful addressing에서의 정보 추출

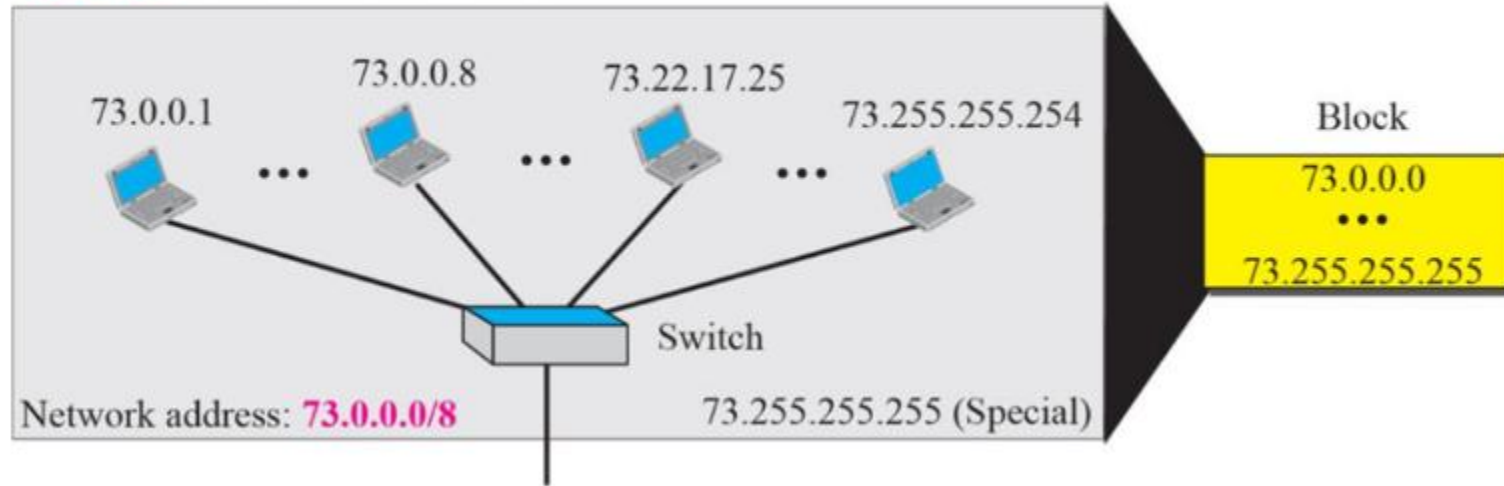


- Netid를 통해 첫 번째 주소와 마지막 주소를 찾을 수 있다.
- 블록 안에 몇 개의 주소가 있는지도 알 수 있다.
- 라우터는 Netid를 통해 네트워크를 찾고, 해당 네트워크에서 Hostid를 통해 기기를 찾는다.



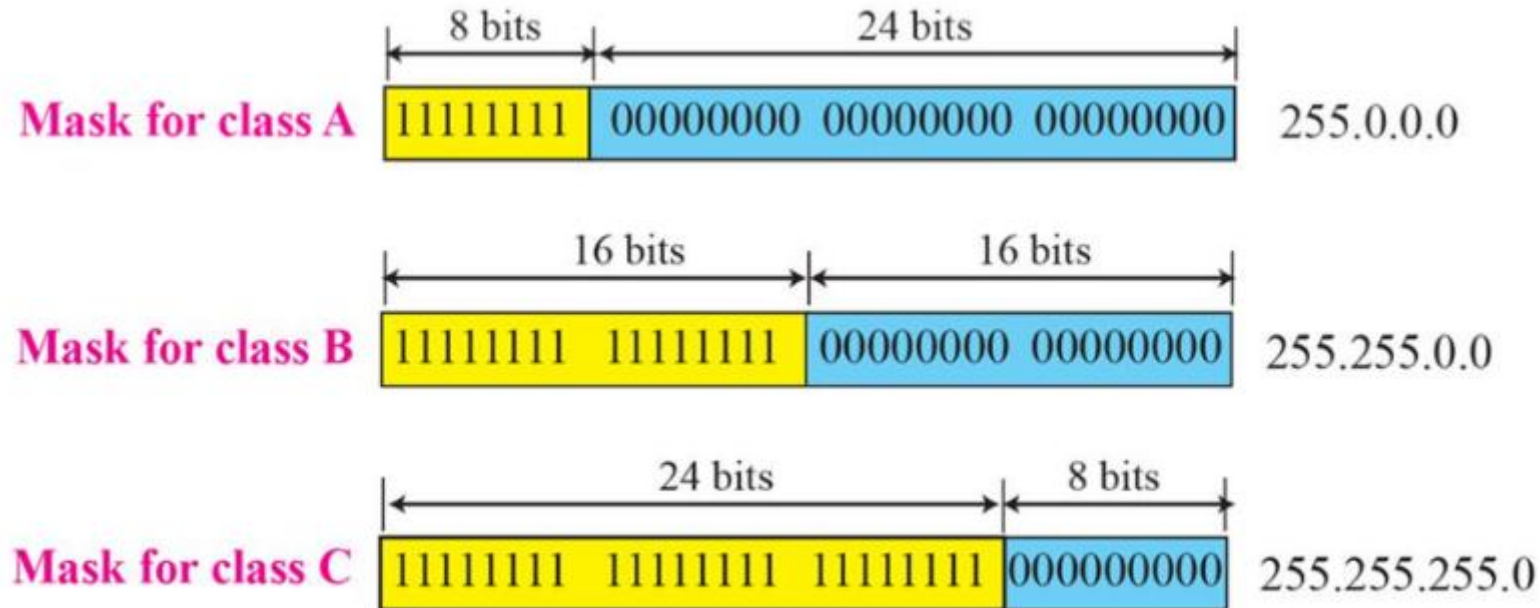
# Classful addressing에서의 정보 추출 예시

Netid 73: common in all addresses



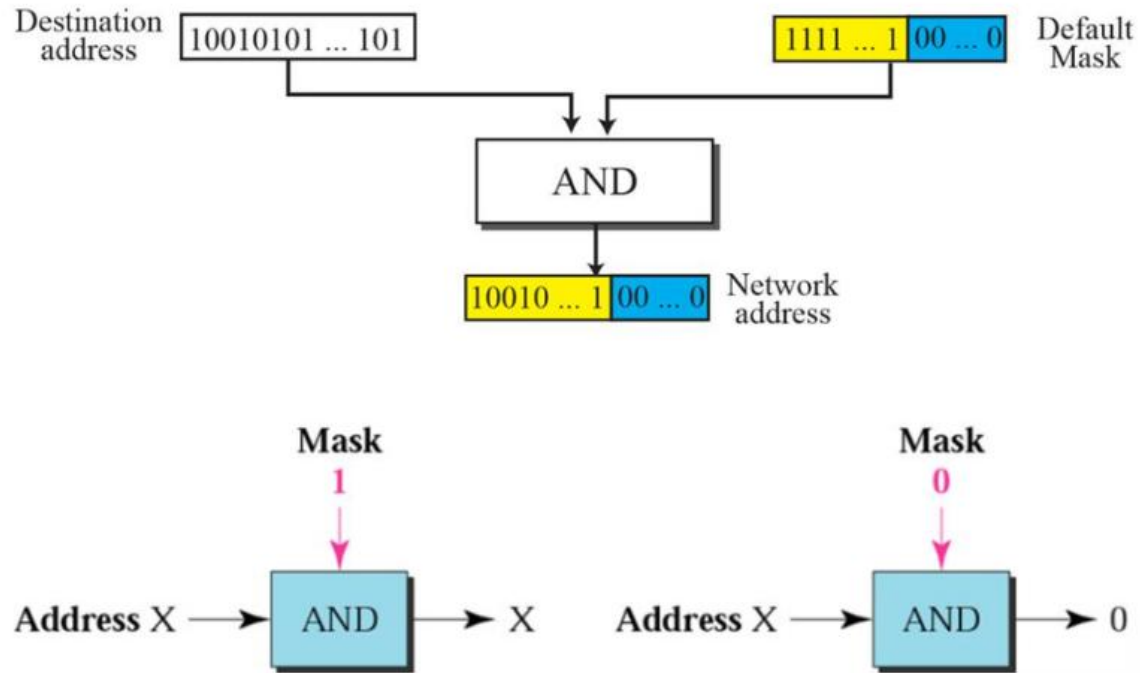
- 그림에서 netid는 73이다. (Class A)
- Network address는 첫 번째 주소인 73.0.0.0/8 이고, 마지막 주소인 73.255.255.255는 나중에 사용하기 위해(special) 이라 쓰이지 않는다.
- 기기가 할당받을 수 있는 ip address의 범위는 73.0.0.1 ~ 73.255.255.254 이다.

# Network Mask



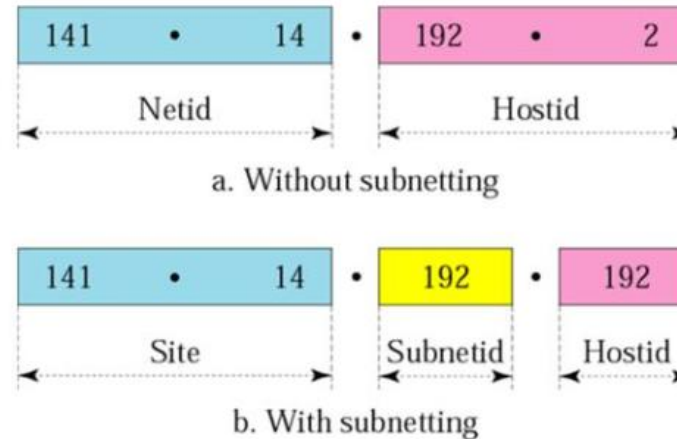
- 패킷에 담긴 destination address(목적지 주소)안에서 network address를 빠르게 찾기 위해 Network Mask를 사용한다.
- 각 클래스 별로 네트워크 마스크가 정해져 있는데, 이를 Default Mask 라고 한다.

# Network Mask



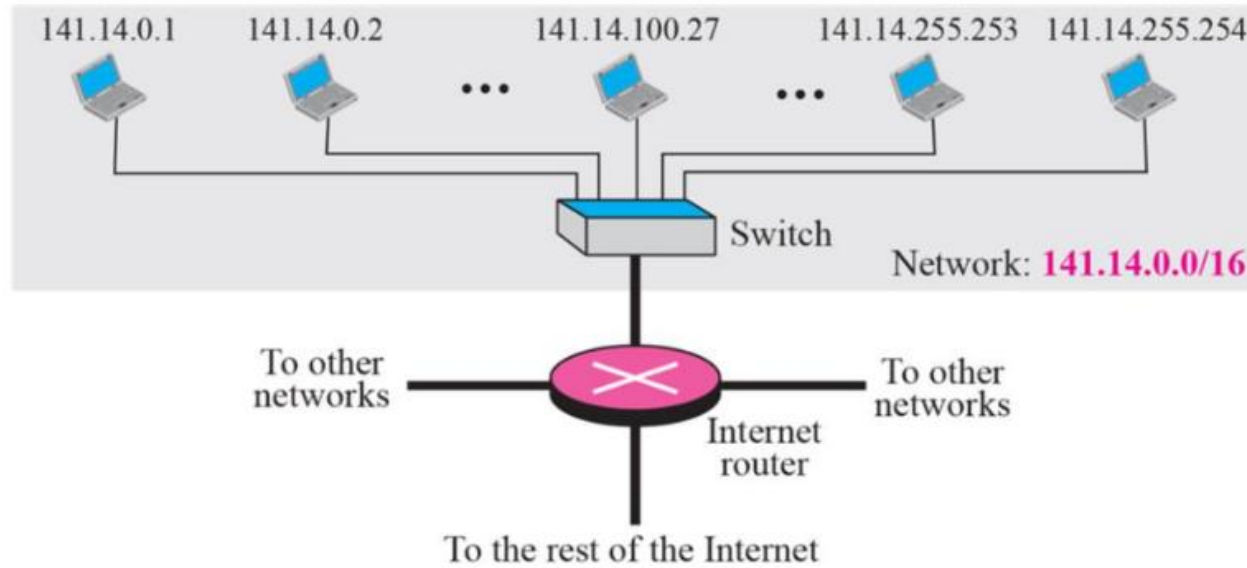
- 위와 같이 목적지 주소에 Network mask(Default mask)와 AND 연산을 사용해서 Network address를 찾는다.

# Subnetting



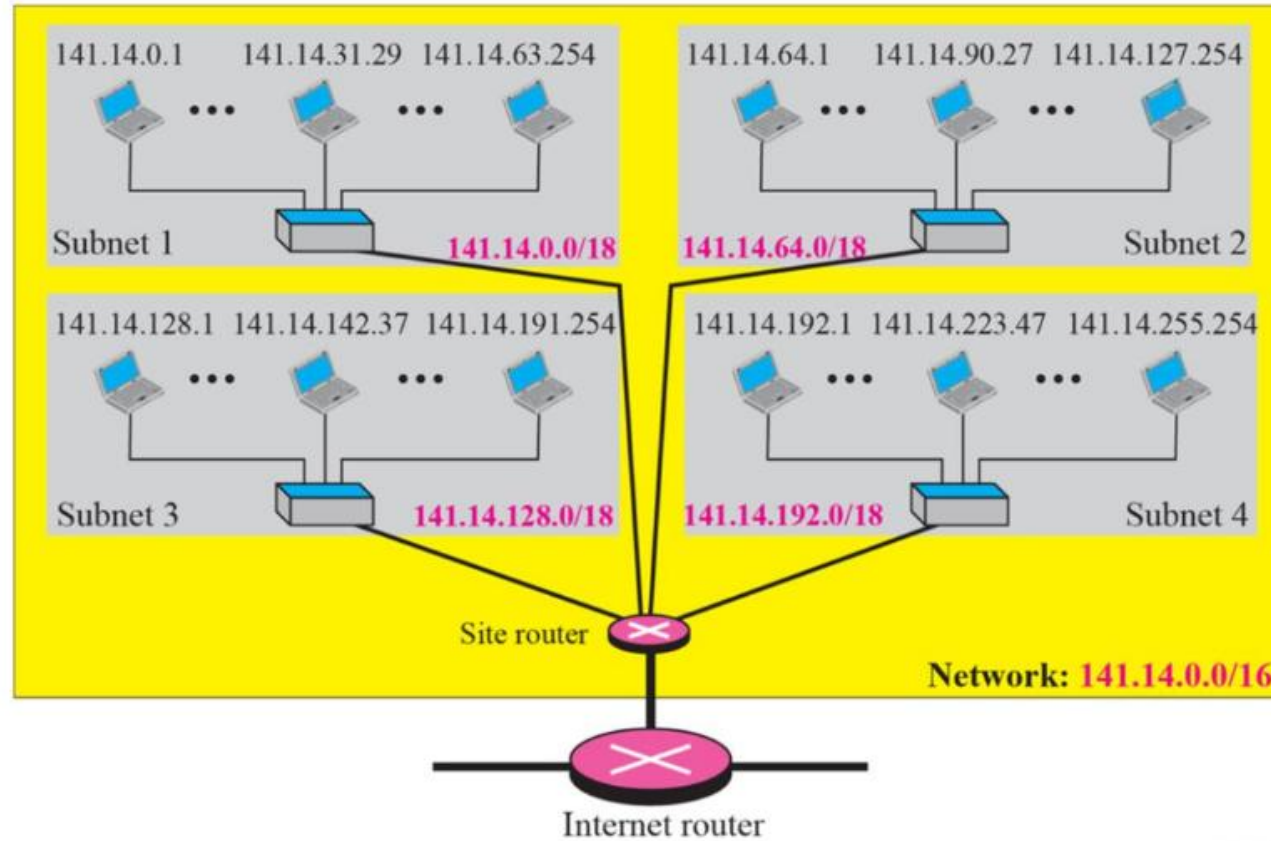
- 큰 네트워크(블록)를 여러 개의 작은 네트워크(블록)로 분할하는 것을 Subnetting이라 한다.
- 나누어진 네트워크(블록)을 Subnet, 나누어진 네트워크의 netid를 subnetid라고 한다.
- 이 Subnetting은 다음과 같은 이유로 사용된다.
  - 네트워크 관리의 효율성 증가
  - 보안 강화
  - IP 주소 낭비 감소
  - 트래픽을 분리하여 네트워크 성능 향상

# Subnetting 예시

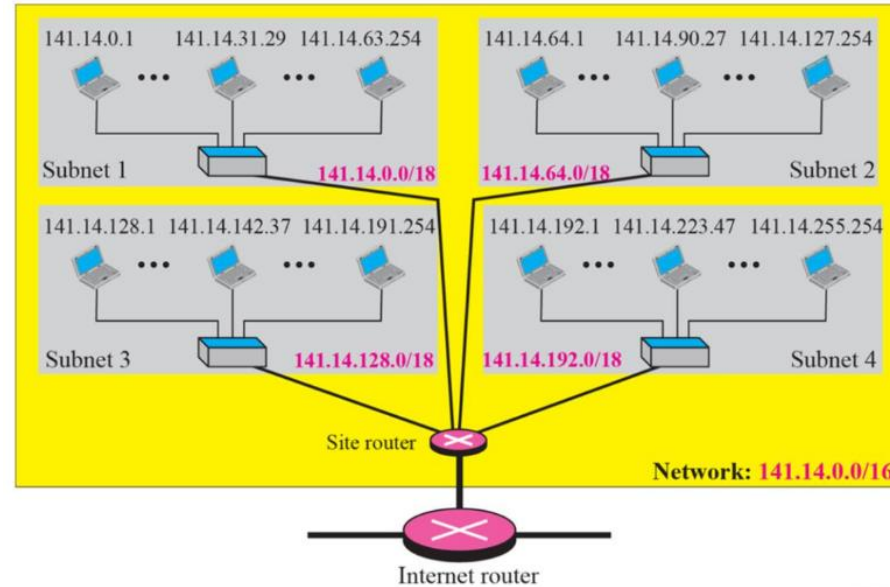


- 그림의 Class는 B이므로 hostid는  $2^{16}$ 개 존재한다.
- 이 때 관리해야 할 주소가 너무 많으니 분할해서 관리하겠다는 것이다.

# Subnetting 예시

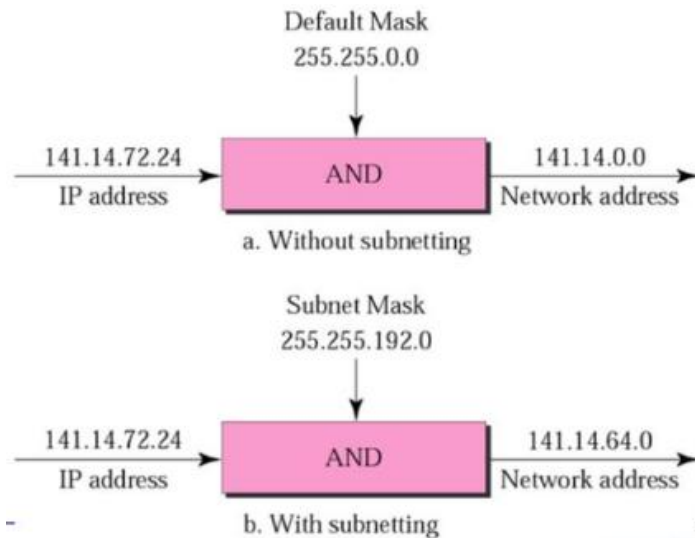
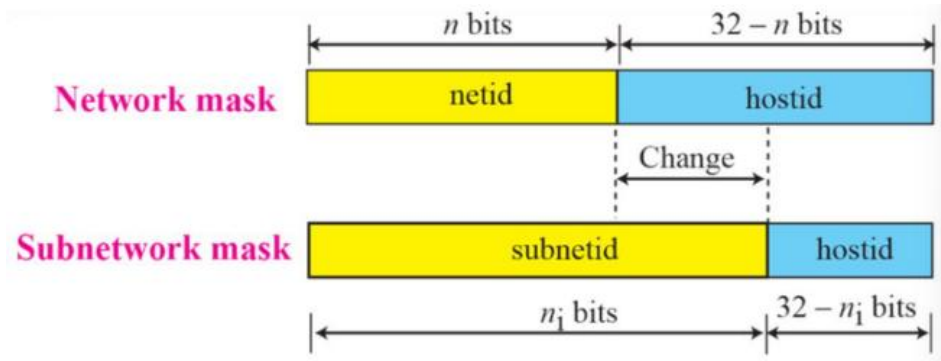


# Subnetting 예시



- 주소를 4개의 그룹으로 쪼갬다.
- 그룹이 4개이므로 이를 구분하기 위해서는 2bit가 필요하다. [00 , 01 , 10 , 11]
- 그림은 원래 Class B였으므로 netid에 할당된 bit 수가 16개였으나 서브넷을 4개로 분할하면 구분을 위해 2 bit가 추가로 필요하니 총 (16+2) 18 bit이다.
- 즉, 서브넷으로 분할하면 각 서브넷은 또 각자 고유한 network address를 필요로 하므로 더 많은 network address를 구분하기 위해 hostid에 할당될 bit가 줄어들고 이를 netid에 할당하는 것이다.

# Subnetwork Mask (Subnet Mask)



- 서브넷을 사용하면 Subnet Mask라는 것을 사용하게 된다.
- 서브넷 마스크란, IP 주소를 네트워크 부분과 호스트 부분으로 나누기 위해 사용되는 32비트의 비트 마스크로
- 즉, 서브넷 구조에서 사용되는 네트워크 마스크이다.
- Network Mask (Default Mask)를 쓸 때보다 훨씬 유연하고 세분화된 네트워크 범위를 정의할 수 있다.
- 즉, 효율적인 IP 할당이 가능해진다.



# Classful addressing의 문제점

- IP 주소 낭비
  - 클래스 A, B, C로 IP 주소를 고정된 크기로 나누었기 때문에, 실제 필요 이상의 많은 IP 주소를 할당하게 되는 경우가 많다.
- 네트워크의 비효율적인 크기 할당
  - 고정된 네트워크 크기가 있었기 때문에, 실제로 적은 수의 주소를 사용하는 네트워크가 큰 주소 범위를 차지하게 될 수 있다.
- 위 문제점들은 곧 주소 고갈 문제로 이어진다.

# 문제 해결 방법

- 주소라는 것은 한정적이기 때문에, 사용자가 많아지면 결국 고갈 될 것이다.
- 이를 해결하기 위한 2가지 방법이 있다.
- long-term solution : IPv6
- short-term solution : classless addressing
- Classless addressing은 IPv4의 주소 고갈 문제를 해결하기 위해 고안된 short-term solution이다.
- 현재는 Classful Addressing이 거의 사용되지 않으며, 대부분의 네트워크에서는 이 Classless Addressing (CIDR) 방식이 사용되고 있다.



# Classless Addressing

# Classless addressing(클래스 없는 주소 지정)

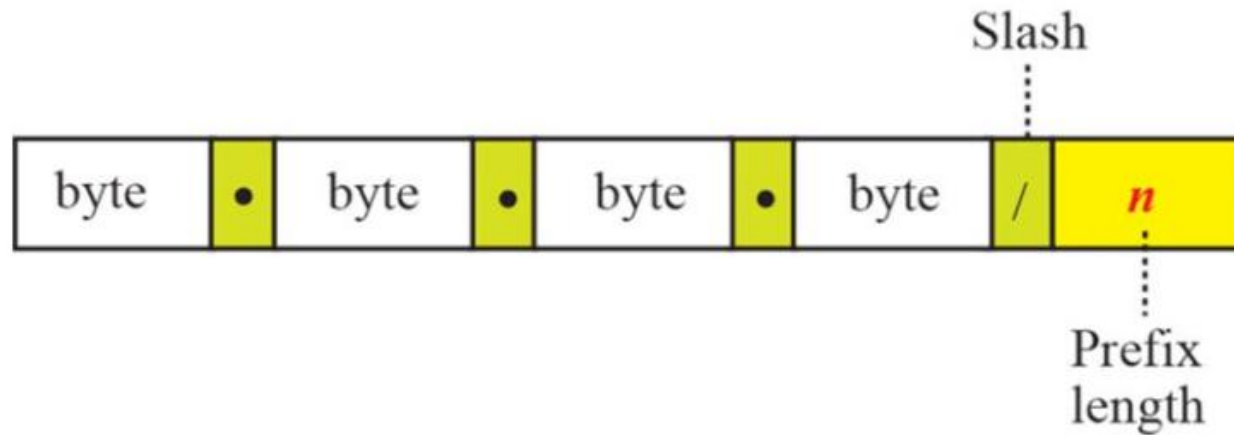
- 주소 공간이 정해져 있었던 Classful addressing에서는 netid와 hostid를 할당하는 비트의 크기가 고정되어 있었다.
- 즉, 추가 네트워크를 만들어서 사용하고 싶을 경우, subnetting을 통해 netid에 할당된 bit를 늘려줘야 했다.
- 그런데 classless addressing에서의 주소 공간(netid + hostid)은 변할 수 있는 가변 길이 블록으로 나뉜다.
- 단, 블록에 포함되는 주소의 수는 2의 제곱수가 되도록 설정해야 한다.
  - (ex. 10개 x -> 16개 o)

# 2단계 주소 체계 Prefix & Suffix



- prefix = netid
- suffix = hostid
- classless addressing은 IPv4에서의 solution이므로, classful addressing과 같은 32 bit 주소이다.

# / 표기법 (Slash notation)



- / 뒤의 *n*은 다음을 나타낸다.
- classful addressing : netid의 길이 (8, 16, 24)
- classless addressing : prefix의 길이
  - class가 없기 때문에 prefix의 길이가 정해져야 주소가 어느 블록에 속한지 알 수 있다.
- 주의) *n*이 잘못 설정되면 주소가 속한 블록이 아예 달라질 수 있다.

# Classless Addressing에서의 Subnetting

- 이전에 classful addressing에서 네트워크 수는 적은데 그 안에 있는 host의 수가 많다면 네트워크를 쪼개서 여러 네트워크로 분할하는 서브네팅을 했었다.
- Classless addressing에서도 subnetting이 존재한다.
  - 네트워크 안의 네트워크 안의 네트워크... 이런 식의 액자식 구성을 효율적으로 관리하기 위함
- 할당 받은 주소 덩어리를 분할해서 subnet들로 나눌 수 있다.
- prefix의 길이가 증가해서 subnet prefix 길이를 결정한다.

# Subnetting 조건

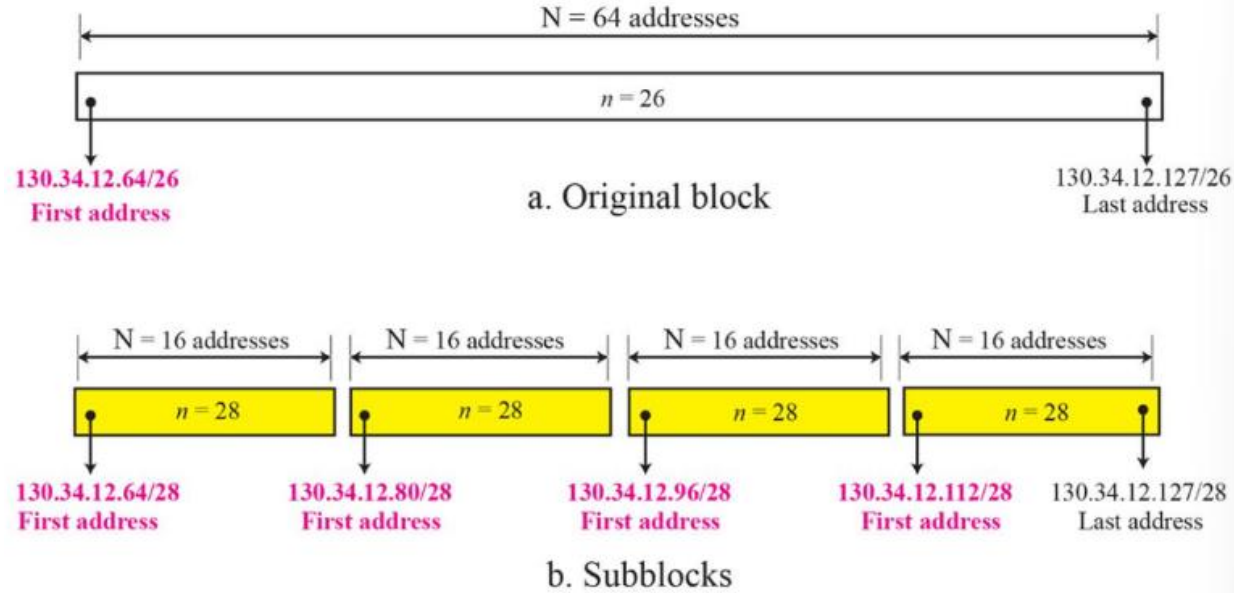
- 블록에 포함된 주소의 수는 2의 제곱이 되도록 설정해야 한다.
- 서브넷의 prefix 길이는 다음 식을 이용해 구해야 한다.
- $n$  = 원래 prefix 길이
- $N$  = 주소의 수 ( $2^{\text{suffix}}$  길이)
- $N_{\text{sub}}$  = 서브넷에 부여된 주소 수
- $n_{\text{sub}}$  = 서브넷의 prefix 길이
- 큰 네트워크(prefix가 큰 블록)부터 주소 할당

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

(사실 식 안써도 구할 수 있다.)

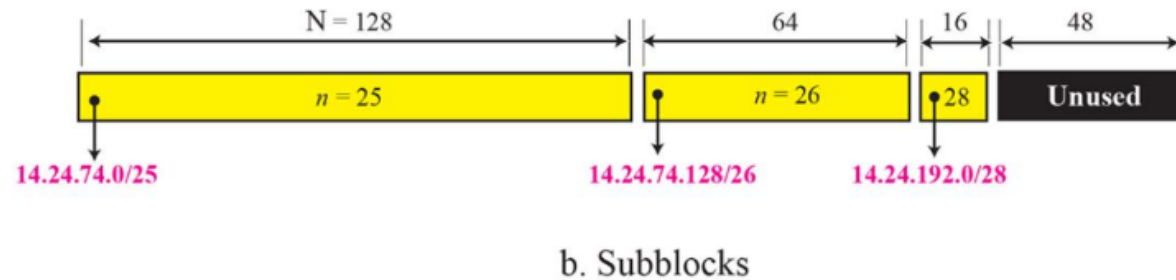
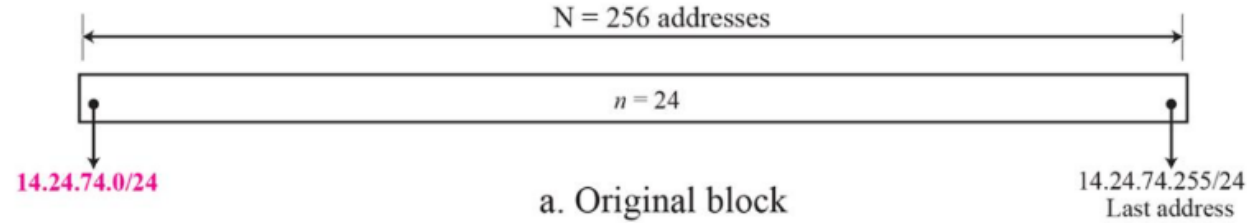


# Subnetting 예시 1



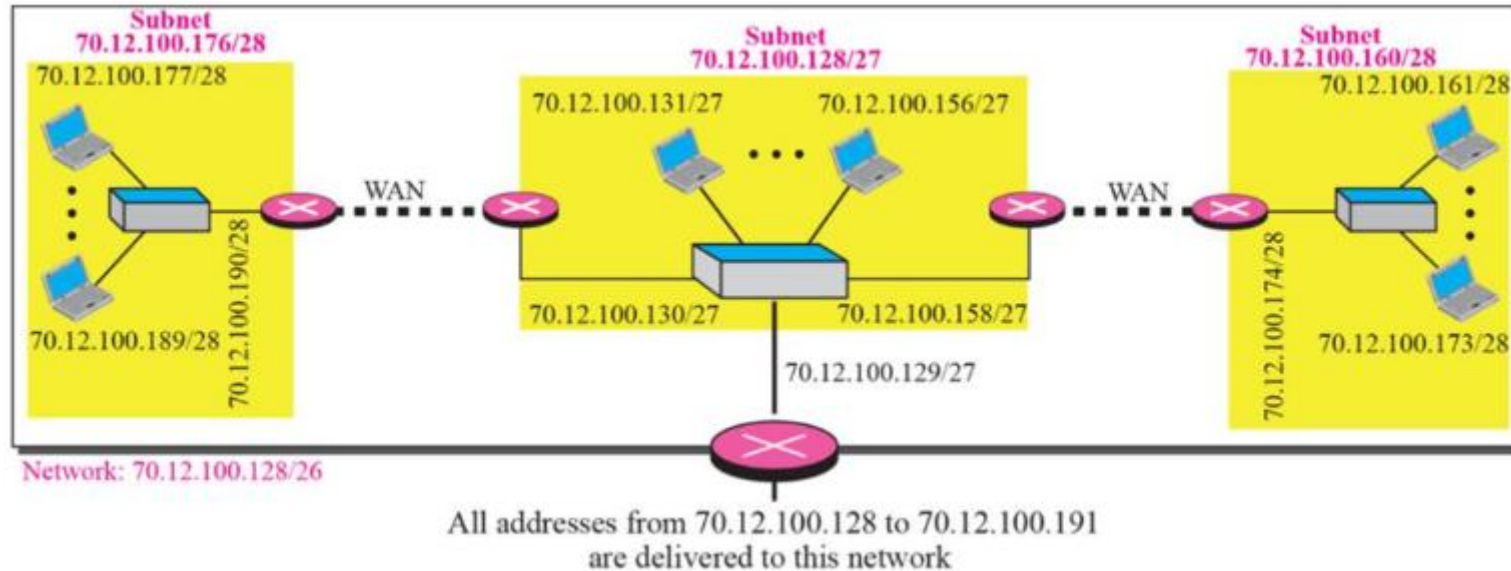
- 4개의 서브블록과 각 서브블록 안에 같은 수의 host를 요청했을 때

# Subnetting 예시 2



- 서브블록의 안의 주소의 수가 120, 60, 10개가 되도록 요청했을 때
- 큰 네트워크( $n$ 의 값이 크다)부터 할당한다.  
120  $\rightarrow$  128  
60  $\rightarrow$  64  
10  $\rightarrow$  16 이 되도록 할당한다.

# Subnetting 예시 3



- 중앙 office에 32개, 왼쪽, 오른쪽 office에 16개씩의 주소를 요청했을 때
- 왼쪽, 오른쪽을 나눌 때의 순서(주소 범위)는 바뀌어도 되지만, 중앙 office는 무조건 먼저 분할해야 한다.
  - 큰 네트워크부터 분할해야 하므로.

# 아무튼 결국은 IPv4는 동이 날 것..

- 2011년 2월, 인터넷 관리 기구인 IANA는 더 이상의 IPv4 할당이 없을 것이라고 선언.
- 너무 빠른 인터넷의 수요 증가로 인한 한정된 주소(약 42억개)를 더 이상 할당할 수 없게 되었기 때문..
- 그런데 14년 전에 이미 동나버린 IPv4임에도 여태껏 이걸 잘 사용하고 있다. 왜일까?
- 많지 않은 수의 IPv4를 가지고 현재까지 잘 인터넷을 사용할 수 있는 이유는 바로 사설 네트워크(Private Network) 덕분이라고 해도 과언이 아니다.



# 사설 네트워크 (Private Network)

# 사설 네트워크? 사설망? 그게 뭔데?

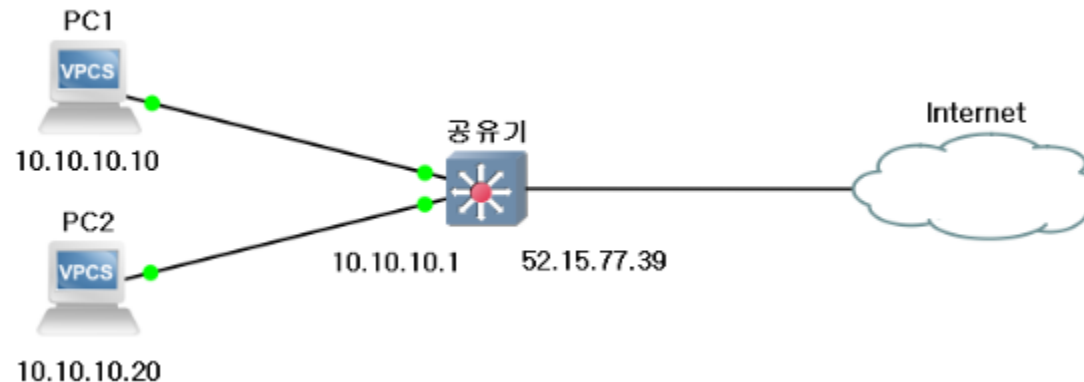
- 사설 네트워크(= 사설망, Private Network)는 IPv4 중 특정 대역을 공인 네트워크(인터넷 등)를 의미한다.
- 사설 네트워크에 소속된 IP(사설 IP 대역)은 다음과 같다.

RFC1918 이름	IP 주소 범위	주소 개수	클래스 내용	최대 사이더 블록 (서브넷 마스크)	호스트 ID 크기
24비트 블록	10.0.0.0 – 10.255.255.255	16,777,216	클래스 A 하나	10.0.0.0/8 (255.0.0.0)	24 비트
20비트 블록	172.16.0.0 – 172.31.255.255	1,048,576	16개의 인접 클래스 B	172.16.0.0/12 (255.240.0.0)	20 비트
16비트 블록	192.168.0.0 – 192.168.255.255	65,536	256개의 인접 클래스 C	192.168.0.0/16 (255.255.0.0)	16 비트

- 위 IP는 오로지 사설 네트워크에서만 사용이 가능하기 때문에 공인 네트워크(인터넷 등 외부 네트워크)에서는 사용할 수 없다.

# 사설 네트워크? 사설망? 그게 뭔데?

- 집에서 사용하는 PC, 휴대폰, ipTV, PS와 같은 게임기 등은 공유기가 할당해주는 사설 IP를 사용한다.
- 이는 기업 등도 마찬가지로, 스위치나 라우터, 방화벽 같은 네트워크 장비 등에 사설 IP와 서브넷 마스크를 지정하고 이를 게이트웨이(사설 IP 할당)로 사용하며 여기 연결된 컴퓨터에 사설 IP를 할당한다.





여기서 잠깐,  
게이트웨이<sup>란?</sup>  
(Gateway)



# 게이트웨이?



사전적 의미로는,  
당연히 프로토스 게이트웨이 역시 게이트웨이이다.

## gate·way

1. (문이 달려 있는) 입구 2. (...로 가는) 관문 3. ~에 이르는 길

발음 미국·영국 [ 'geɪtweɪ ]

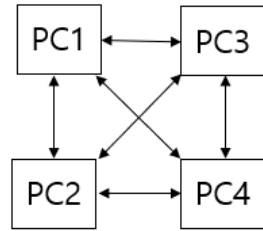


- 사전적 의미로 ‘입구’, ‘관문’
- 즉, a 장소에서 b 장소 출입할 때 이용하는 출입문이라는 것.

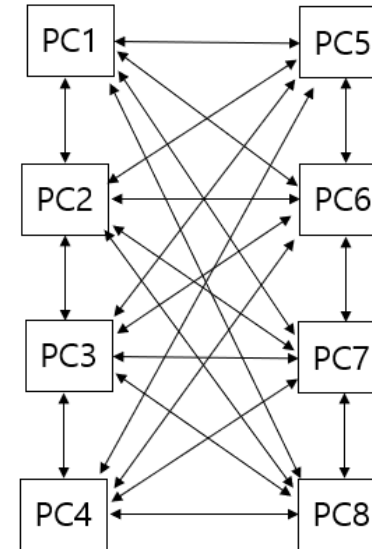
# 게이트웨이?

- 네트워크 입장에서 보면, 게이트웨이는 해당 네트워크에서 다른 네트워크로 나가는 출구이며, 다른 네트워크로 들어가는 입구이다.
- 데이터 입장에서 보면, 게이트웨이는 데이터가 외부 네트워크로 나거나 외부 네트워크로 들어갈 때 반드시 지나가야 하는 통로이다.
- 즉 게이트웨이란, 다른 네트워크로 나가거나 다른 네트워크에서 들어오도록 하기 위해 꼭 필요한 통신 기기(스위치나 라우터, 방화벽 같은 네트워크 장비 등)의 일부분이다.

# 게이트웨이 : 왜 필요한데?



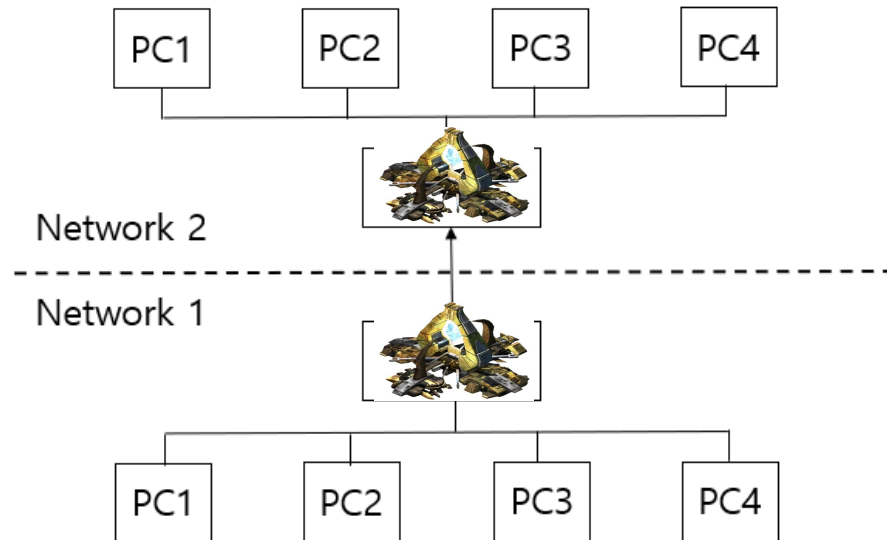
(간편하다)



(비효율적이고, 복잡하다)

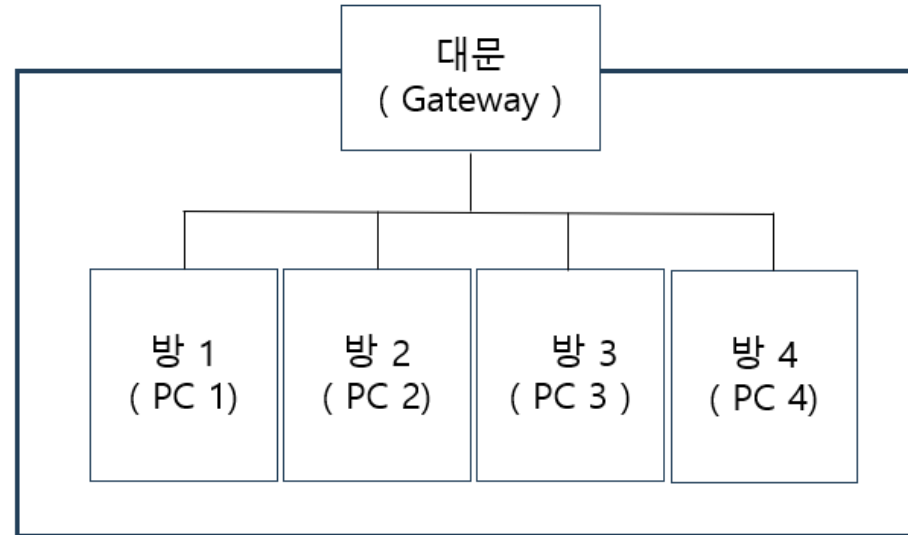
- 같은 네트워크에 있는 소수의 호스트끼리 통신 시에는 직접 서로 통신해도 괜찮지만, 서로 다른 네트워크에 있는 다수의 호스트들이 서로 통신을 할 경우, 경로가 방대해지고 복잡해지기 때문에 매우 비효율적이다.

# 게이트웨이 : 필요하겠는데?



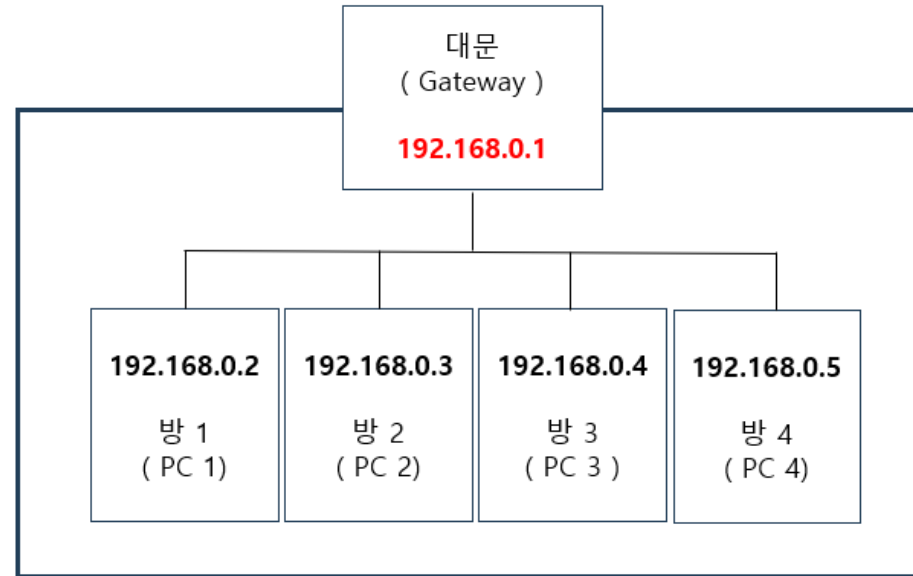
- 그렇기 때문에 통신 경로를 단순화하고, 효율적으로 관리하기 위해 호스트들을 여러 그룹으로 나눈다.
- 이 때 각 그룹의 접속 지점 즉, 연결 관문이 바로 게이트웨이이다.

# 게이트웨이 : 주택에 비유하자면..



- 각 방 : 호스트 PC
- 대문 : 게이트웨이
- 각 방 끼리는 대문을 거치지 않고도 통신 가능하다.

# 게이트웨이 주소 : 대문 번호도 필요하다.



- 대문을 표시하기 위한 대문 번호도 필요한데, 이 대문에 할당된 번호를 게이트웨이 주소라고 할 수 있다.
- 게이트웨이 주소란? 주택 내부에 있는 여러 문들 중 대문을 식별하기 위해 지정된 번호

# 다시 사설 네트워크 설명으로 돌아와서..

- 이렇듯, 사설 네트워크와 공인 네트워크는 각각 사설 IP 주소와 공인 IP 주소를 사용하여 서로 분리된다.
- 그러나 사설 IP 주소는 사설 네트워크에서만 유효하며, 공인 네트워크(ex. 인터넷)에서 직접 사용할 수 없다.
  - 이는 동일한 사설 IP 주소가 여러 네트워크에서 사용될 수 있기 때문에, 공인 네트워크로 나갈 때 IP 충돌을 피하기 위함이다.
- 이 때문에 사설 네트워크에서 공인 네트워크로 나가고자 할 때 사설 IP를 공인 IP로 변환 할 필요가 생겼다.
- 이를 위해 필요한 것이 바로 NAT(네트워크 주소 변환) 과정이다.



# NAT (Network Address Translation)

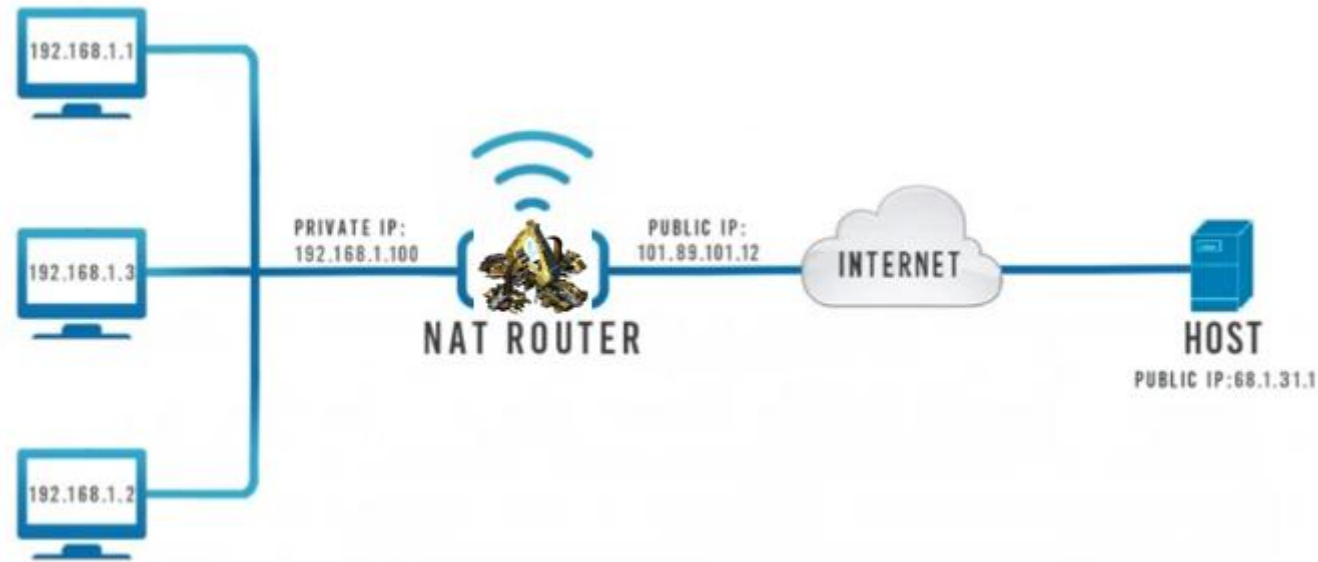


# NAT(네트워크 주소 변환) 이란?

네트워크 주소 변환(영어: network address translation, 줄여서 NAT)은 컴퓨터 네트워킹에서 쓰이는 용어로서, IP 패킷의 TCP/UDP 포트 숫자와 소스 및 목적지의 IP 주소 등을 재기록하면서 라우터를 통해 네트워크 트래픽을 주고 받는 기술을 말한다.

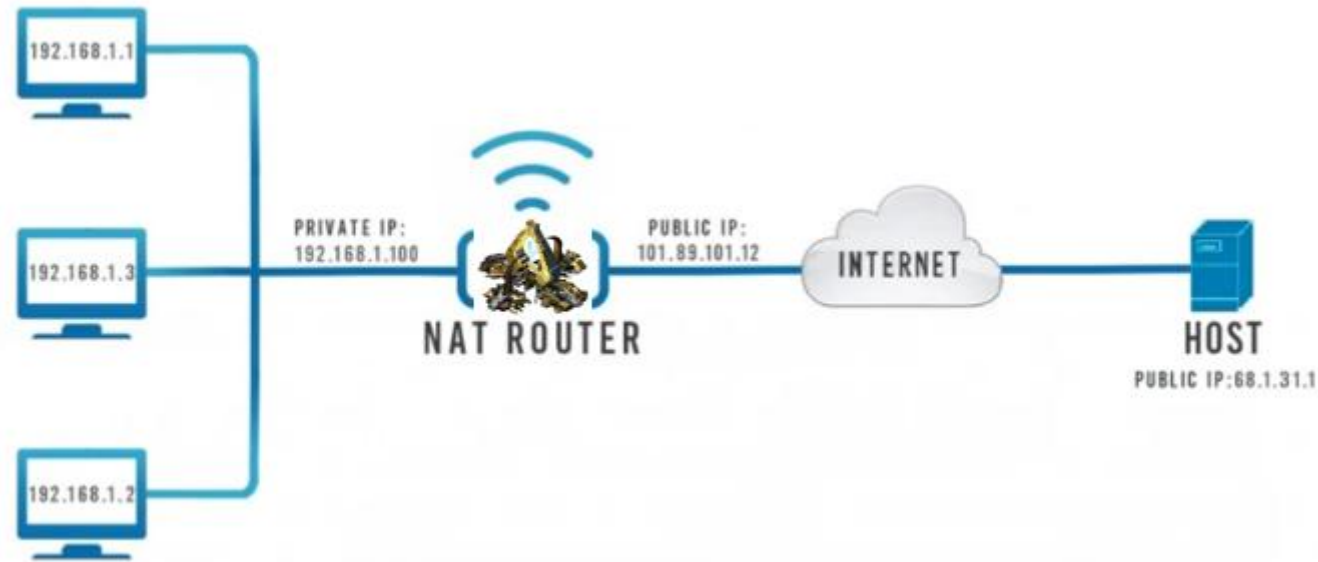
출처 : 위키백과

# NAT(네트워크 주소 변환) 이란?



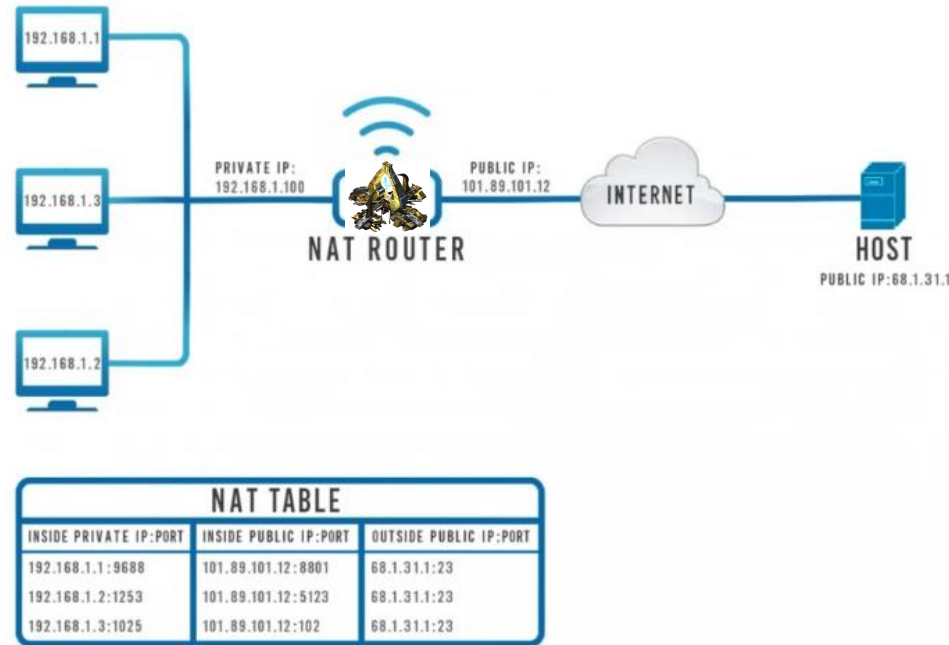
- 쉽게 말해, 사설 네트워크(LAN), 공인 네트워크(인터넷) 간 IP 주소를 변환해주는 기술
- 이를 통해 사설 네트워크 내부의 여러 장치가 한 개의 공인 IP 주소를 사용할 수 있게 된다.
- 여기서의 **NAT 라우터가 곧 게이트웨이의 역할**을 한다.
- 인터넷에서는 해당 전역 주소를 가지는 라우터를 제외한 LAN의 나머지 부분은 볼 수 없다.

# NAT 라우터와 주소 변환



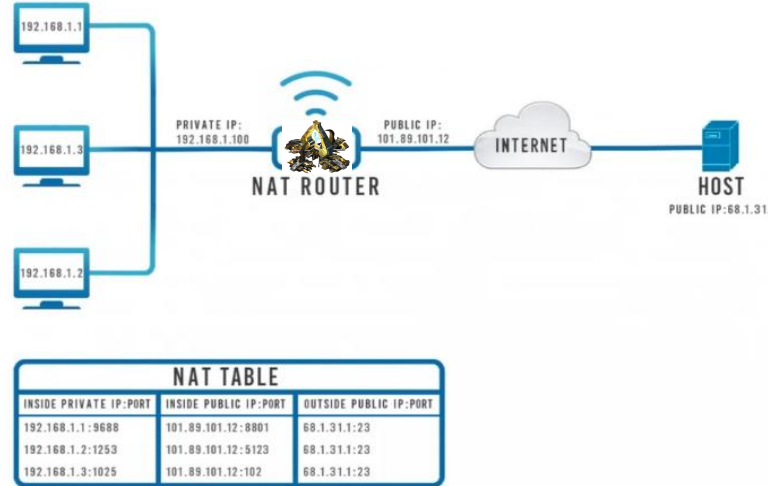
- 외부로 전송되고자 하는 패킷들은 패킷의 source address를 전역 주소로 바꾸는 NAT 라우터를 통해 전송된다.
- 외부에서 들어오는 패킷들도 NAT 라우터를 통해 들어오고, 라우터는 패킷의 destination address를 그에 맞게 적절한 사설 주소로 변경한다.

# 변환 테이블



- 변환 테이블은 사설 네트워크(LAN)을 위해 할당된 블록으로부터 선택된 사설 주소를 전역 주소로 변환하는데 사용된다.
- 변환 테이블은 사설 IP 주소(내부, 외부)와 인터넷 IP 주소, 2가지의 정보를 담은 열을 가지고 있다.

# 변환 테이블



- LAN 안의 클라이언트가 특정 패킷을 외부로 전송하고 싶다면, 라우터가 이를 확인하고 라우터 테이블 안에 LAN 안의 클라이언트의 발신지(Source) 주소를 기록해둔다.
- 그 후 외부 목적지(Destination)로부터 응답이 도착하면, 라우터는 테이블을 확인하며 어느 발신지가 보낸 목적지로부터 온 패킷인지 확인하며 짝을 맞춘다.
- NAT 메커니즘에서는 항상 사설 네트워크(LAN)에서의 통신부터 시작되어야 한다.



감사합니다.