6/26

- ex1

172.16.0.0/16

255.255.0.0

172.16.0.0 /22

172.16.4.0/22

172.16.8.0/22

172.16.12.0/22 ⇒ /24비트로 쪼갠다

172.16.12.0/24

172.16.13.0/24

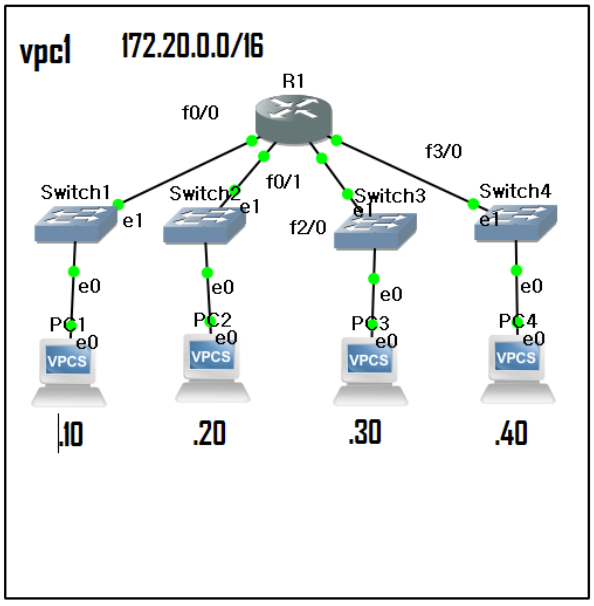
172.16.14.0/24

172.16.15.0/24

네트워크 비트 6개 늘린다 서브넷팅 64개한다.

255.255.252.0

```
ip 172.20.0.10 255.255.252.0 172.20.0.1
ip 172.20.4.20 255.255.252.0 172.20.4.1
ip 172.20.8.30 255.255.252.0 172.20.8.1
ip 172.20.12.40 255.255.252.0 172.20.12.1
각 pc
```



필요한 서브넷 4개 /22

172.20.0.0 /22
172.20.4.0 /22
172.20.8.0 /22
172.20.12.0 /22

LAN1) 172.20.0.0 /22 GW : 172.20.0.1
LAN2) 172.20.4.0 /22 GW : 172.20.4.1
LAN3) 172.20.8.0 /22 GW: 172.20.8.1
LAN4) 172.20.12.0 /22 GW : 172.20.12.1

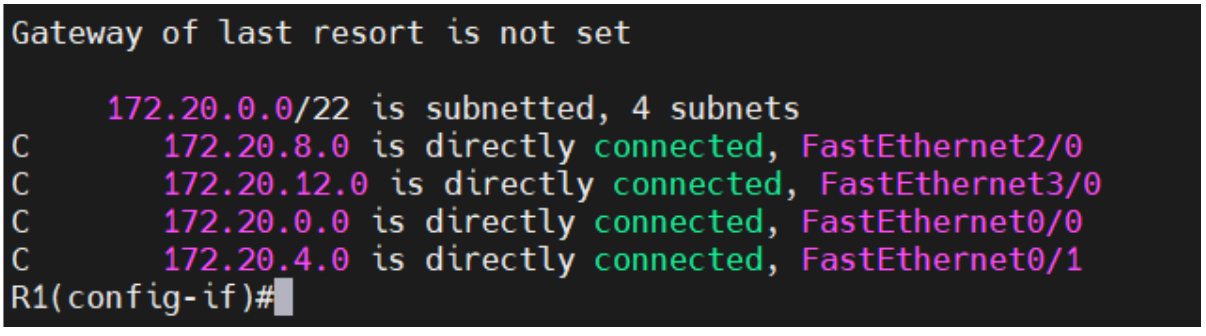
가운데에 있는 라우터가 각 pc들의 통신을 도와준다.

```
PC1> ping 172.20.4.20
172.20.4.20 icmp_seq=1 timeout
84 bytes from 172.20.4.20 icmp_seq=2 ttl=63 time=31.774 ms
84 bytes from 172.20.4.20 icmp_seq=3 ttl=63 time=31.590 ms
84 bytes from 172.20.4.20 icmp_seq=4 ttl=63 time=31.660 ms
84 bytes from 172.20.4.20 icmp_seq=5 ttl=63 time=31.493 ms

PC1> ping 172.20.8.30
172.20.8.30 icmp_seq=1 timeout
84 bytes from 172.20.8.30 icmp_seq=2 ttl=63 time=31.802 ms
84 bytes from 172.20.8.30 icmp_seq=3 ttl=63 time=31.566 ms
84 bytes from 172.20.8.30 icmp_seq=4 ttl=63 time=31.649 ms
84 bytes from 172.20.8.30 icmp_seq=5 ttl=63 time=31.563 ms

PC1> ping 172.20.12.40
172.20.12.40 icmp_seq=1 timeout
84 bytes from 172.20.12.40 icmp_seq=2 ttl=63 time=32.099 ms
84 bytes from 172.20.12.40 icmp_seq=3 ttl=63 time=31.663 ms
84 bytes from 172.20.12.40 icmp_seq=4 ttl=63 time=31.624 ms
84 bytes from 172.20.12.40 icmp_seq=5 ttl=63 time=31.619 ms
```

첫번째 pc에서 보냈을 때 잘 가는 것을 확인 가능

처음에 빠지는건 arp 때문이다.

```
R1)
conf t
int f0/0
ip add 172.20.0.1 255.255.252.0
no sh

int f0/1
ip add 172.20.4.1 255.255.252.0
no sh

int f2/0
ip add 172.20.8.1 255.255.252.0
no sh

int f3/0
ip add 172.20.12.1 255.255.252.0
no sh
```



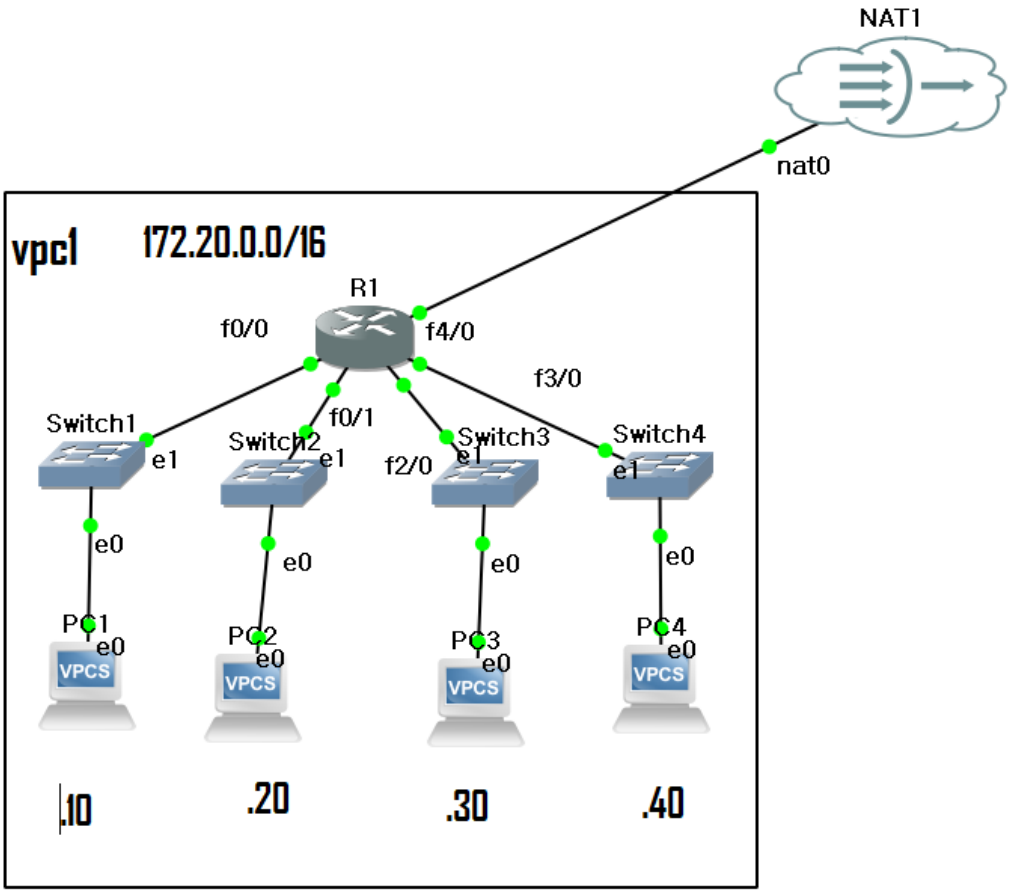
연결이 잘 된 것을 확인 가능하다.

10.0.0.0 / 28

호스트 비트가 4개 남아있기 때문

10.0.0.0 ~ 10.0.0.15

nat는 vmware가 깔려 있어야 사용 가능



외부로 통신하기 위해 nat gateway를 이용한다.

```
PC1> ping 8.8.8.8
8.8.8.8 icmp_seq=1 timeout
84 bytes from 8.8.8.8 icmp_seq=2 ttl=127 time=94.844 ms
84 bytes from 8.8.8.8 icmp_seq=3 ttl=127 time=63.638 ms
84 bytes from 8.8.8.8 icmp_seq=4 ttl=127 time=63.615 ms
84 bytes from 8.8.8.8 icmp_seq=5 ttl=127 time=78.849 ms
```

pc1에서 구글 dns로 보냈을 때 되는 걸 확인 가능

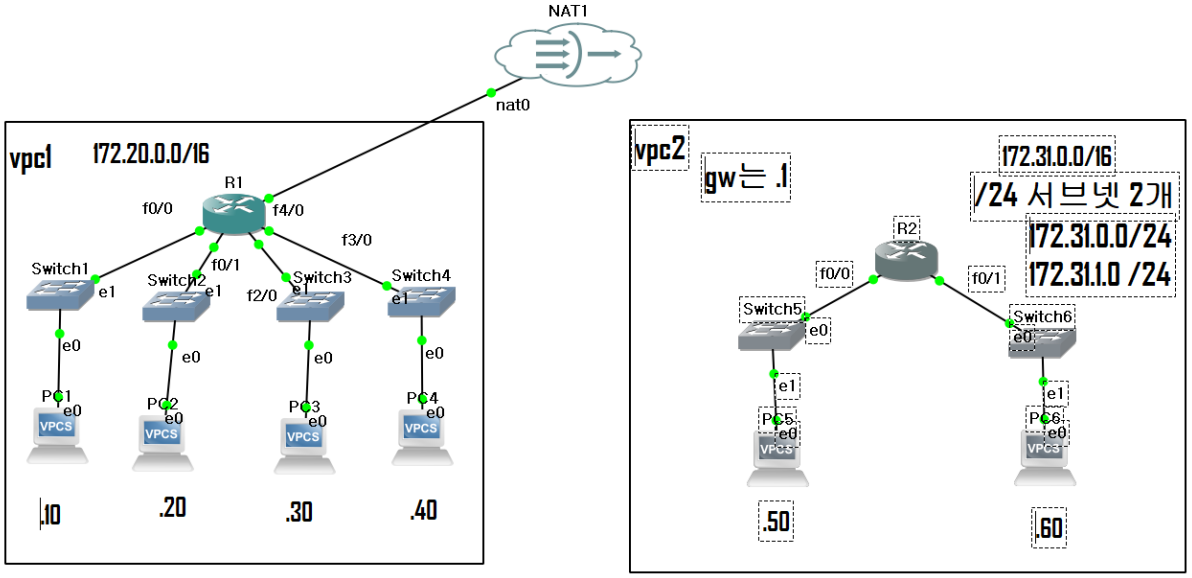
nat gateway는 비용이 들어간 유료 서비스

nat는 private subnet이 외부와 연결하기 위함 바깥에선 해당 subnet에 접근 불가

- public subnet

  외 내부 통신 가능
- private subnet

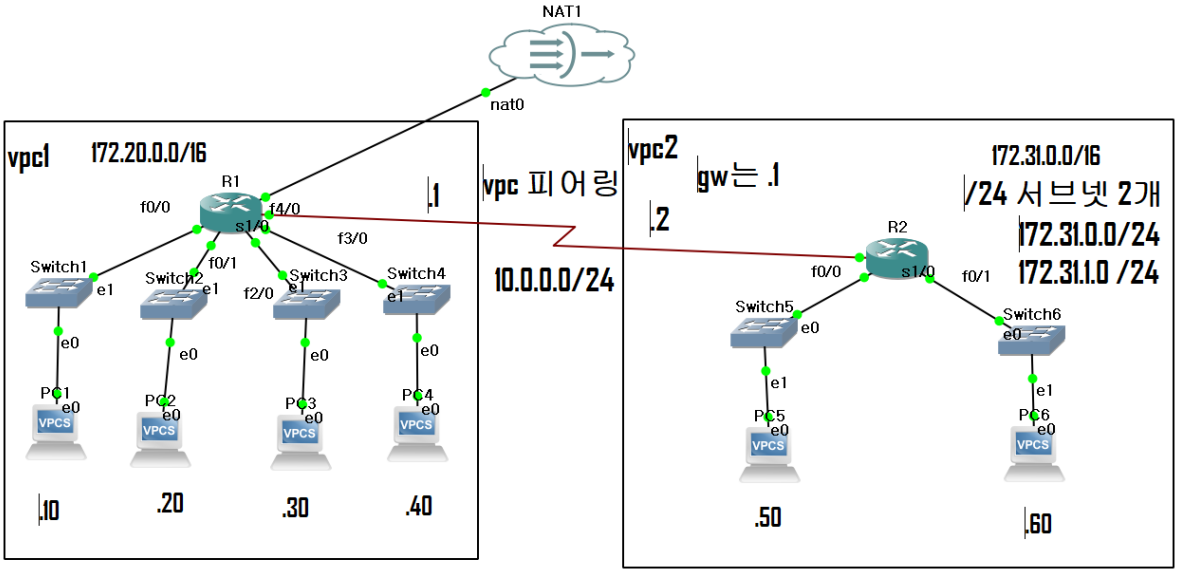
  nat 달시 외부통신 가능 외부에선 접근 불가능



vpc2 코드는 위와 비슷

vpc간의 통신을 해보자

vpc peering이라는 service가 있다.



s1/0 연결이 된 모습

```
R1)
conf t
int s1/0
ip add 10.0.0.1 255.255.255.0
no sh

ip route 172.31.0.0 255.255.0.0 s1/0 10.0.0.2

R2)
conf t
int s1/0
ip add 10.0.0.2 255.255.255.0
no sh

ip route 172.20.0.0 255.255.0.0 s1/0 10.0.0.1
```

```
PC1> ping 172.31.0.50
172.31.0.50 icmp_seq=1 timeout
172.31.0.50 icmp_seq=2 timeout
84 bytes from 172.31.0.50 icmp_seq=3 ttl=62 time=63.663 ms
84 bytes from 172.31.0.50 icmp_seq=4 ttl=62 time=63.520 ms
84 bytes from 172.31.0.50 icmp_seq=5 ttl=62 time=62.693 ms
```

pc1 에서 pc5으로 핑이 가는 것을 확인 할 수 있다.

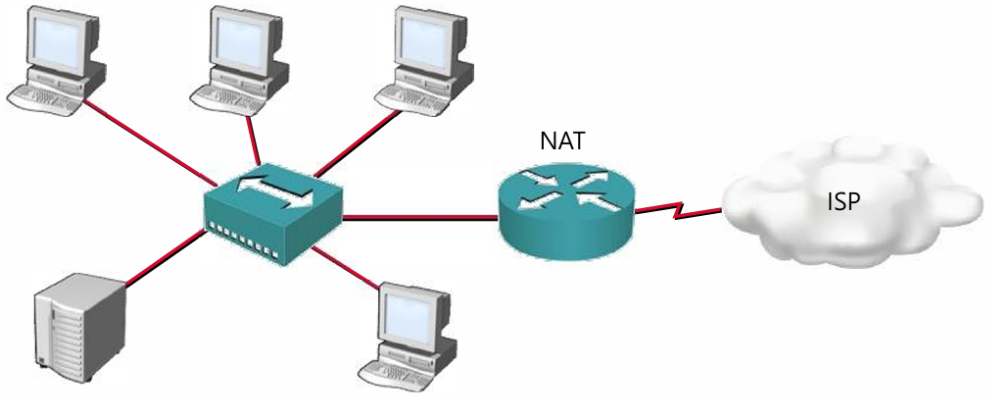
transit gateway를 통해 여러개의 vpc간의 연결을 통제해 줄 수 있다. 편하게 사용 가능 transit gateway도 일종의 라우터이다.

```
R2(config)#ip route 0.0.0.0 0.0.0.0 s1/0 10.0.0.1
R2(config)#do sh ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 10.0.0.1 to network 0.0.0.0

     172.20.0.0/24 is subnetted, 1 subnets
S       172.20.0.0 [1/0] via 10.0.0.1, Serial1/0
     172.31.0.0/24 is subnetted, 2 subnets
C       172.31.1.0 is directly connected, FastEthernet0/1
C       172.31.0.0 is directly connected, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Serial1/0
S*   0.0.0.0/0 [1/0] via 10.0.0.1, Serial1/0
```

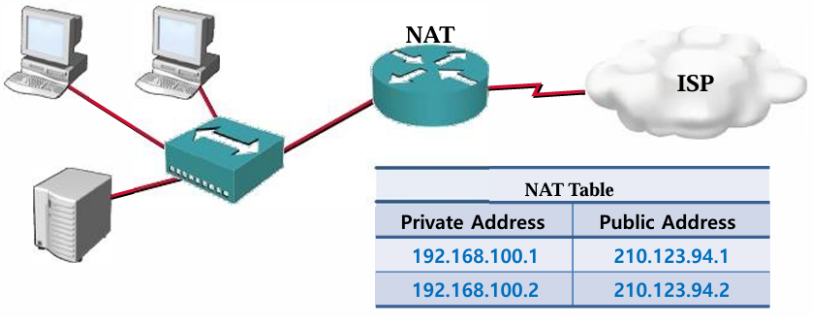
vpc2에서도 다음과 같이 vpc1를 통해서 nat를 이용해서 외부 접근 가능

NAT(Network Address Translation)

- NAT는 RFC 1631에 정의된 것으로 IP Header의 주소를 다른 주소로 바꾸는 기술이다.
  NAT는 사설주소를 사용하는 호스트들이 인터넷에 서비스를 이용할 수 있도록 하기 위해 사용한다.



Static NAT와 Dynamic NAT



NAT Table	
Private Address	Public Address
192.168.100.1	210.123.94.1
192.168.100.2	210.123.94.2

- 정적 NAT는 외부주소로 들어온 요청을 내부 서버로 전달 될 수 있도록 목적지 주소를 변환하는 기능이다.
  이 방법으로 사설망 서버를 구현하고 외부 주소로 들어오는 연결을 내부 서버로 전달할 수 있다.

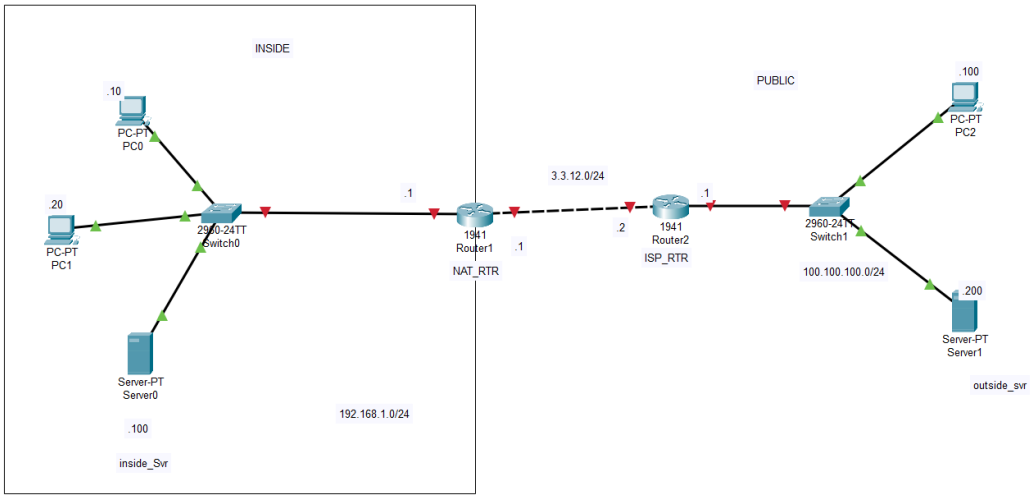
- 동적 NAT는 호스트가 요구하는 Traffic을 받으면 IP 주소내에서 사설 IP를 라우터에 설정된 주소 Pool에 있는 공인 IP로
  변환한 후 외부로 전달한다. 외부에서 응답 신호가 라우터로 돌아오면 NAT라우터는 NAT Table에 있는 이전 정보로
  목적지로 들어온 주소를 사설 IP로 변환해서 내부망으로 전달한다.

외부 → 내부 가 static nat이다.

내부 → 외부가 dynamic nat이다.

목적이 다르다.

nat 기능을 할 수 있는 장비는 많이 있다.



static nat는 local ip 주소와 global ip 주소가 1 대 1 매핑되어있기 때문에

내부에서 외부로 외부에서 내부로 모든 트래픽이 허용됨.

local ip 주소만큼 global ip 주소가 필요하므로 내부 시스템의 인터넷을 위한 용도로는 사용하지 않음.

또한 외부에서 내부로도 모든 트래픽이 허용됨으로 보안상 취약하다.

-Portforwarding : static nat의 특별한 형태 tcp와 udp로 서비스를 구분해 외부에서 내부의 특정 서비스에 접속할 수 있도록 하는 nat

ip nat inside source static tcp 192.168.1.100 21 3.3.12.1 21

static은 1:1이지만 dynamic은 그렇지 않다.

dynamic nat: 내부 클라이언트가 인터넷이 가능하도록 공인주소 1개 또는 몇 개 중에 바꿔주는 기능 클라이언트 주소를 식별하기 위해서 acl을 사용함.

- 설정순서
  1. acl 작성 : 변환할 네트워크 대역을 acl로 작성

     access-list 1 permit 192.168.1.0 0.0.0.255
  2. nat문 작성

     1) 인터페이스 ip 주소로 변환하는 경우

     ip nat inside source list 1 interface g0/0 [overload]

     overload 옵션은 기본옵션으로 NAT → PAT로 변환

     주소 변환 시 포트번호까지 사용 PORT ADDRESS TRANSLATION

     한 개 일때는 overload 가 자동으로 붙는다

     pool을 사용하면 overload를 반드시 붙여야

     2) 공인주소풀에서 변환하는 경우

     - nat pool 생성

       ip nat pool TestPool 200.200.200.1 200.200.200.2 netmask 255.255.255.0
     - nat문 작성

       ip nat inside source list 1 pool TestPool overload

     pool이 핵심
  3. nat 적용

     int g0/1

     ip nat inside

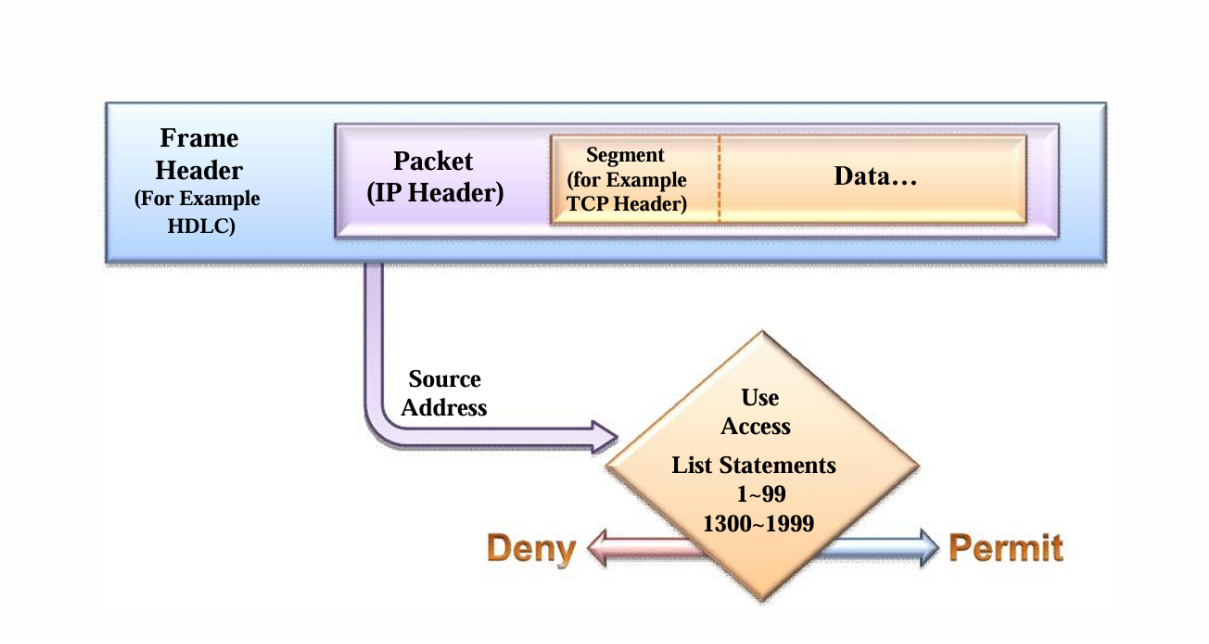
     int g0/0

     ip nat outside

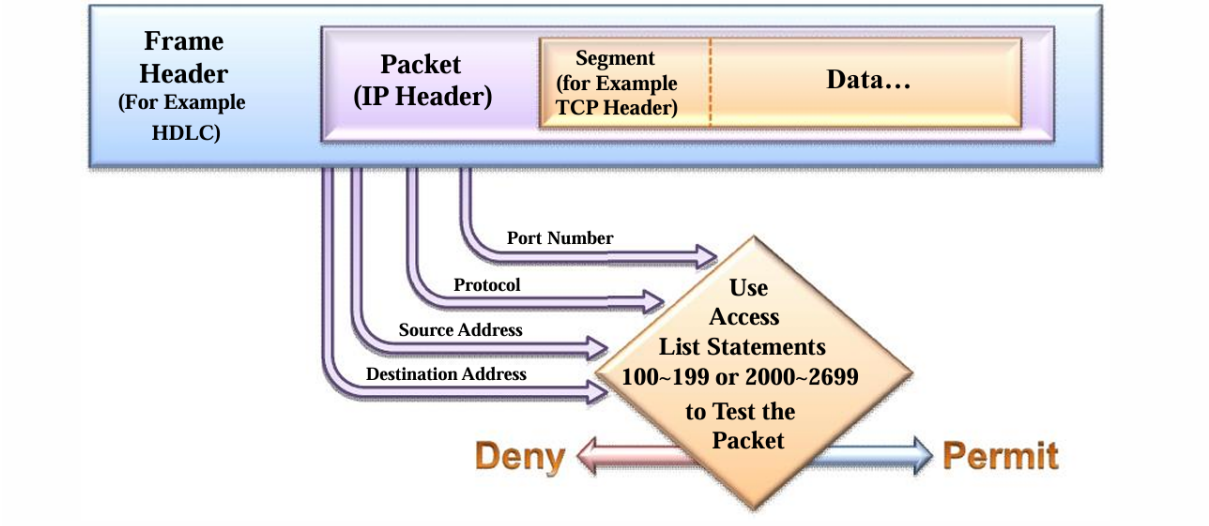
     192.168.1.0 / 24 (254개 ip) → 3.3.12.1 10000,10001,10002

- acl

표준 ACL의 패킷필터링



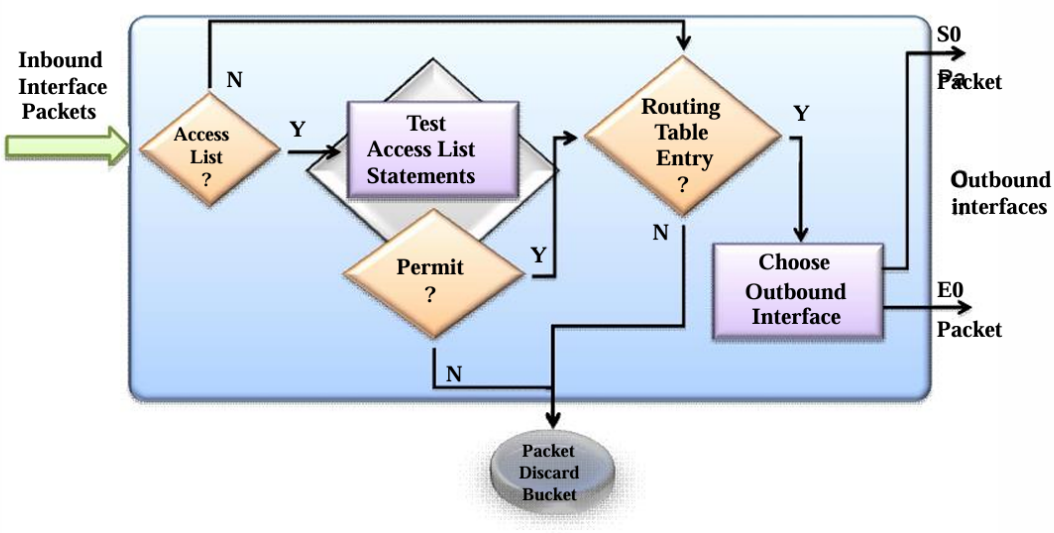
확장 ACL의 패킷필터링



확장은 detail하게 필터링

standard는 거의 안쓰인다. 주로 extand를 사용한다.

인바운드 ACL의 동작과정



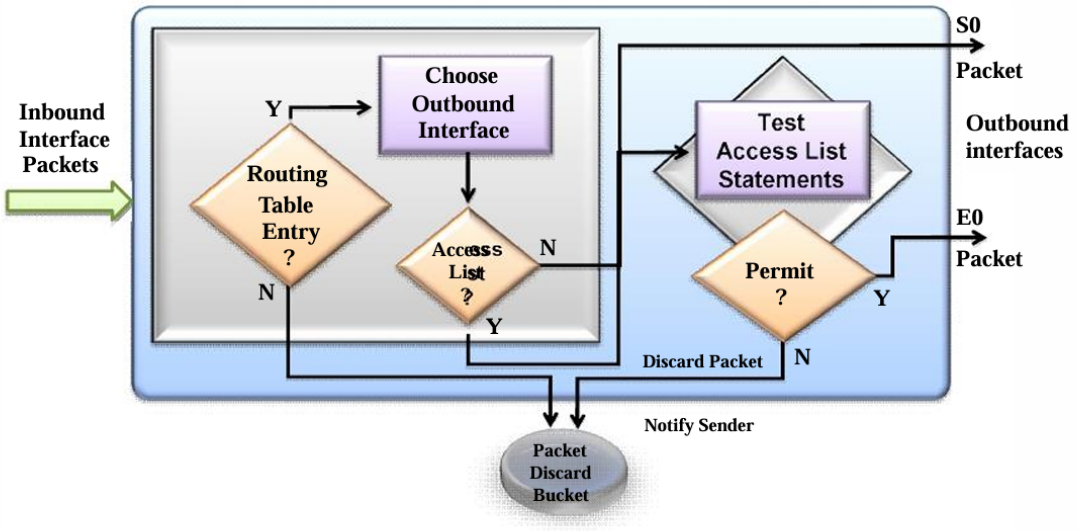
• Access List에 매치되지 않는 모든 Packet은 암묵적으로 거부된다

acl에 의해 허락이 되는 만 라우팅으로 넘어간다.

acl에 의해서 차단될거면 라우팅 테이블을 확인할 필요가 없다.

따라서 라우터 입장에선 인바운드가 더 나은 선택 불필요한 절차가 안생긴다.

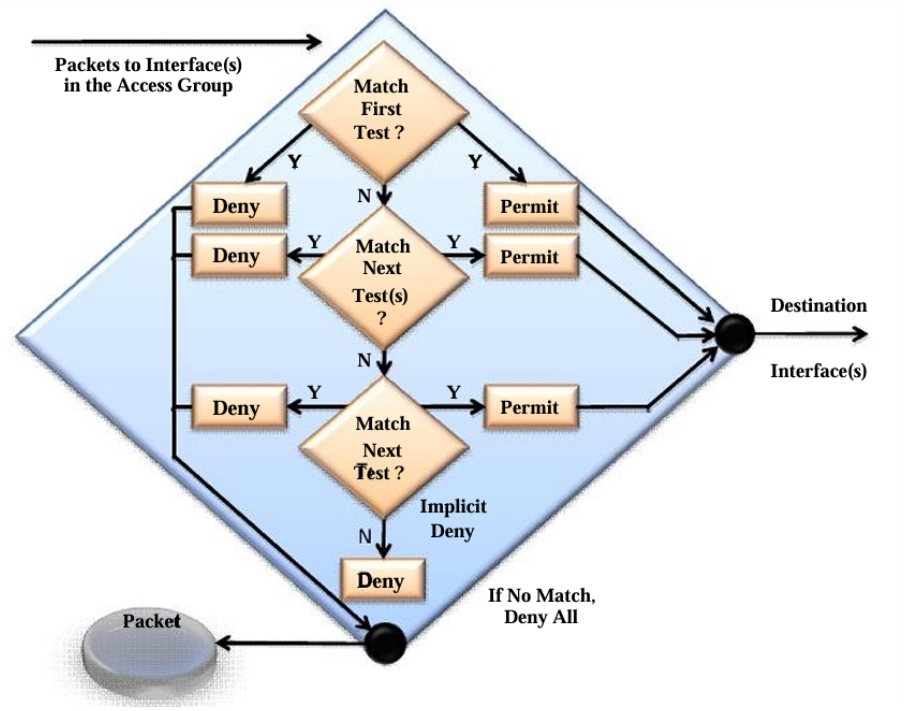
아웃바운드 ACL 동작과정



• Access List에 매치되지 않는 모든 Packet은 암묵적으로 거부된다

라우팅을 먼저 하고 나갈 때 acl 검증

ACL엔트리의 패킷검사 : Permit, Deny



넓은 범위의 acl이 위로 올라가면 통과 되는 경우가 생긴다.

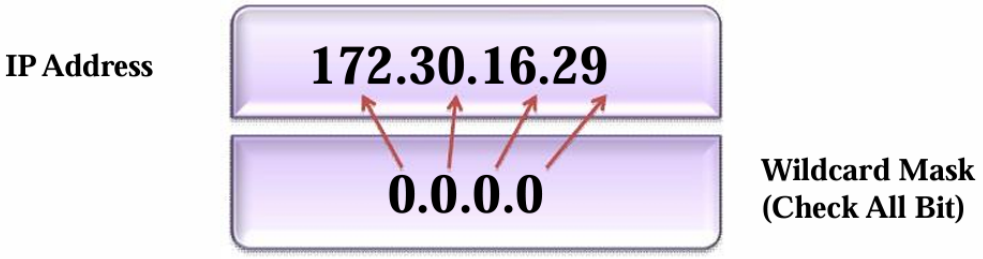
deny의 순서가 중요하다.

- wild card mask

c클래스 마스크를 뒤집어서 쓴다 생각하면 편하다.

특정 IP호스트를 나타내는 wildcard mask

• Test 조건 : 모든 Address Bit 검사 (모두 일치)
(1개의 IP Host Address만 검사)

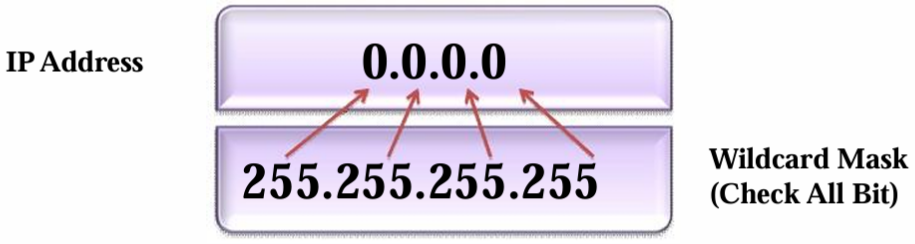


• 172.30.16.29 0.0.0.0은 모든 Address를 검사해서 매치되는 주소 즉, 172.30.16.29를 갖는 호스트를 지정한다

• 하나의 IP를 알리기 위해 IP Address 앞에 약어 host를 사용할 수 있다. 예를 들어 "172.30.16.29 0.0.0.0" 대신 "host 172.20.16.29"를 사용할 수도 있다

모든 주소를 나타내는 wildcard mask

- Test 조건 : 모든 Address Bit 무시 (Match Any)
  (모든 IP Address)

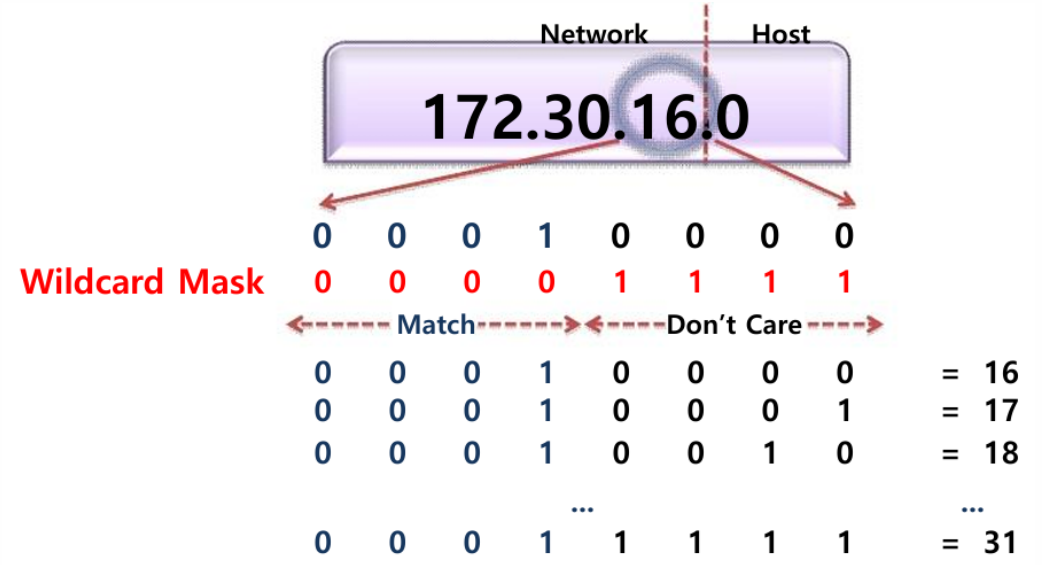


- 모든 Address를 받아들이려면 IP Address는 0.0.0.0을 입력하고 Wildcard mask는 모든 값을 무시(검사 없이 허용)하려면 255.255.255.255를 지정한다
- 관리자는 모든 주소를 지정할 목적으로 0.0.0.0 255.255.255.255를 명시하는 대신 any라는 문자를 사용할 수 있다

특정 IP 서브넷을 나타내는 wildcard mas

• 172.30.16.0/24에서 172.30.31.0/24까지의 IP Subnet 검사하기

- Address와 wildcard mask : 172.30.16.0 0.0.15.255



172.30. 000 1 0000
172.30. 000 1 1111

172 30.0.0 /20 255.255.240.0

ex 1) 다음 ip 주소의 네트워크 주소와 브로드 캐스트 주소는?

219.94.56.100/21

219.94.0011 1000.100

네트워크 주소: 네트워크 비트까지는 일치해야 하고 나머지 비트는 모두 0인 주소

브로드캐스트주소 : 네트워크 비트까지는 일치해야 하고 나머지 비트는 모두 1인 주소

219.94. 0011 1000.0 = 219.94.56.0

219.94. 0011 1111.255 = 219.94.63.255

이 범위 내에 ip는 같은 네트워크

- standard ACL

  출발 주소가 사설 ip 주소 대역을 차단하는 acl

  access-list 10 ~ deny 10.0.0.0 0.255.255.255

  10은 a클래스니까 그것을 뒤집은 것

  access-list 10 deny 172.16.0.0 255.240.0.0 ⇒ 0.15.255.255

  access-list 10 deny 192.168.0.0 0.0.255.255

  access-list 10 permit any

  access-list 10 deny any

  위의 순서대로 acl을 진행 permit any로 받는다.

  > 💡 항상 마지막에 암묵적인 deny any를 넣는다.

  int g0/0

  ip access-group 10 out <= in이 아닌 out으로 조건에 따라 설정한다.

  > 💡 do sh run | i ip nat

  > 💡 do sh run | i ip route

  > 💡 do sh ip ro

- extended acl

  inside_rtr에서 172.16.1.0/24 네트워크에 있는 단말에서 inside_server 서비스 중에 웹 접속 가능하도록 acl 적용

ip access-list extended allow-http

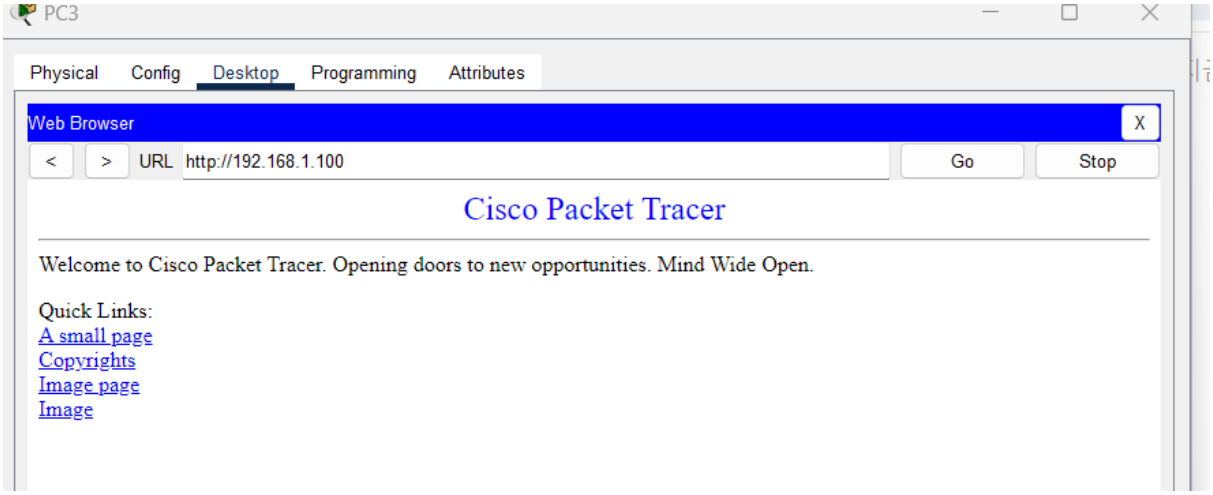
permit tcp 172.16.2.0 0.0.0.255 host 192.168.1.100 eq 80

port 80만 허용

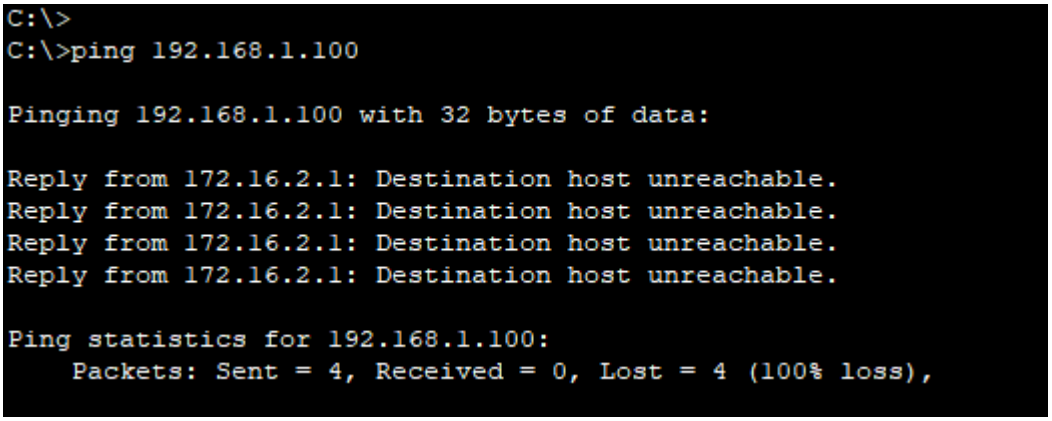
deny ip 172.16.2.0 0.0.0.255 host 192.168.1.100

permit ip any any

deny ip any any



http는 허용



ping은 안되는 모습