CSE 4412 [Data Communication and Networking]

Lab # 8

1. Objectives:

- Describe the concept of Access Control List (ACL)
- Implement standard numbered ACL
- Describe the concept of Network Address Translation (NAT)
- Explain different types of NAT configuration
- Implement NAT in a given topology

2. Theory:

Access Control Lists (ACL):

Defining ***who*** *can/can't access* ***what*** is basically the gist of ACL. In our day-to-day life, we are applying the concept of ACL in many areas. A simple example could be like you need to show your ID card to enter an office. There's a list of employees and your ID is checked against that list to grant access. Similar access controls are in effect in virtually everywhere, especially in places where security is critical. In digital world, this access control is more needed so that only allowed ones can access a certain digital resource. For example, only admins would be allowed access in the backend of a web server or only database admins would be allowed to access database server etc.

In networked devices, ACLs play a crucial role to allow only authorized person/devices to a certain resource. For example, you can define that only a certain host device would be able to access your webserver. You can also define ACLs so that hosts belonging to a particular network can't communicate with hosts of certain other network. There are more scenarios that can be defined depending on the needs of an administrator.

In this lab, we'll learn about Cisco IP ACL i.e., filtering network traffic based on IP address. There are several ACL types that can be configured on a Cisco device. But for the purpose of this lab, we'll only focus on *Numbered Standard IPv4 ACL*. There are two steps to implement an ACL. First, **define the rule**. Second, **apply the rule to an interface**.

The command format for defining a numbered standard IP ACL is:

```
Router(config)# access-list access-list-number
                  {permit|deny}
       {source_address source_wildcard|any}
```

You can either permit or deny a packet based on the source IP of the packet in numbered standard IP ACL. As like the OSPF configuration, you need to specify a wildcard mask to permit/deny a range of source IP addresses based on the given pattern. One important thing you should keep in mind that whenever you apply an ACL to an interface, **all the traffic that doesn't match any ACL rule will be discarded by default**. So, for example, you have defined an ACL to deny a certain source IP. Whenever you apply that rule to an interface, all other packets other than the denied source will also be discarded because there's no matching rule for those packets. So, you must allow other traffic explicitly by defining another ACL. The **`any`** keyword is handy in this case. To permit (or deny) any packet other than the previously specified rules, you can just add the keyword **`any`** in place of the source_address and source_wildcard like the following:

```
Router(config)# access-list 1 permit any
```

Another thing is you can only use numbers in the range *1 to 99* for specifying the *`access-list-number`*. Other numbers are used for **extended numbered ACL**. After defining the ACL rule, now we need to apply it to an interface. Remember that the ACL has no effect until you apply it. The command format for applying an ACL to an interface is like below:

```
Router(config-if)# ip access-group access-list-number
                         {in|out}
```

The ACL is applied either for inbound traffic or outbound traffic of an interface and you need to specify the corresponding keyword i.e., **`in`** or **`out`** for that. One best practice before applying an ACL to an interface is to *verify* the rule by using the following command:

```
Router# show access-lists
```

[For the commands, the **`bold`** ones are *constant keywords* which should be exactly same. `Non-bold` ones are the configurable options.]

Network Address Translation (NAT):

You already know from your theory lectures that IPv6 was born partly due to the address space exhaustion of IPv4. One great technique that was the key for survival of IPv4 is this **NAT**. If not for NAT, IPv4 would be long gone by now. And that gave the world some time to adopt IPv6 in mass scale. In this lab, you'll learn about this special technique called NAT.

Basically, the idea of NAT is there'll be a set of IP addresses for the hosts in the internal network and to the outside world those internal hosts will be exposed using a different set of IP addresses. You know that each host is recognized through its IP address on the internet. To conform with this, each host connected to the internet would have to have a unique IP which readily becomes close to impossible considering there are billions of connected devices.
NAT allows you to assign arbitrary IP addresses to your internal hosts where these addresses are only locally significant i.e., these are locally unique. Then in the edge or gateway router of the network you'll have one or a set of IP addresses which are globally unique. That edge or gateway router will do the

conversion/translation from the globally unique address to the locally unique ones or vice-versa. Outside world won't know the actual IP addresses of internal hosts which provides an extra layer of security. Moreover, your organization can buy only a handful of global IP addresses from the ISP but can use

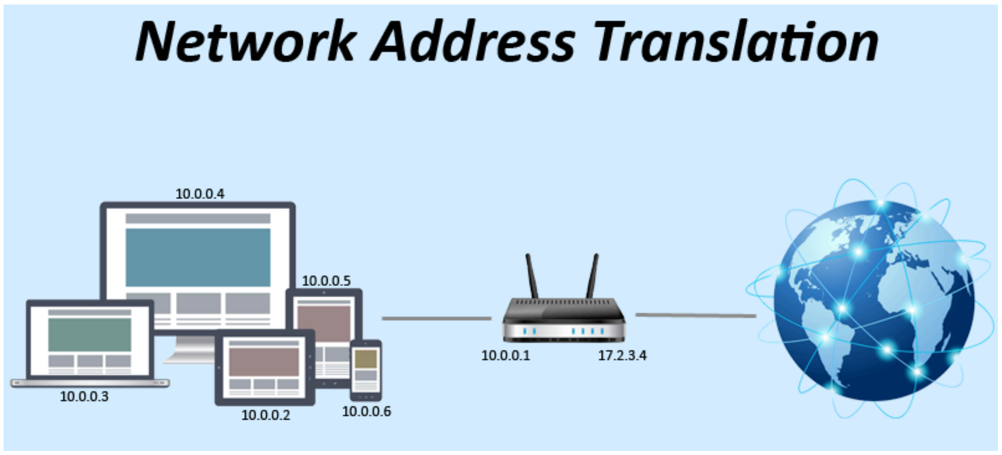


*Figure SEQ Figure * ARABIC 1: NAT*

those with much greater number of hosts through NAT. The following figure summarizes what we just talked about.

Now that we know the basics of NAT, lets get technical. There are three types of NAT that you can define in Cisco devices: Static, Dynamic and Overloaded or Port Address Translation (PAT).

Static NAT

It allows one-to-one mapping between local and global addresses. You will have to configure one global IP address for each of internal hosts that you want NAT to translate. The command to enable the static translation is as follows:

```
Router(config)# ip nat inside source static local_ip global_ip
```

After you've specified the translation, you'll need to do two things. First, you need to specify the *inside* interface. The inside interface denotes that the hosts connected to this interface will have their IPs translated to the global one. Second, you need to specify the *outside* interface. Outside interface denotes that through this interface the translated packets will go out to the world. There can be more than one inside or outside interfaces. After you specify these interfaces, NAT will start the specified translations. You need to specify these *inside* and *outside* interfaces for the other two NAT types also. Following are the commands to specify the inside and outside interfaces:

```
Router(config-if)# ip nat inside
Router(config-if)# ip nat outside
```

Dynamic NAT

This type of NAT establishes a mapping between a local address and a pool of global addresses. For a single local address, a global IP address will be selected from the pool dynamically. When not in use, the assigned global IP will be released after a certain time-out period so that other hosts can re-use it. This is more convenient than the static one as you don't need to manually configure every mapping. For configuring dynamic NAT, first you need to *create an access list permitting the **local addresses*** to get translated. Then you have to specify the pool of global IP addresses from where the IPs will be allocated. The pool is a range of IP addresses in a given network where the subnet mask will specify the corresponding network portion. The command to specify the pool is as follows:

```
Router(config)# ip nat pool POOL_NAME start_ip end_ip netmask
                         subnet_mask
```

Then you need to establish the relation between the earlier defined *access list* and *nat pool* through the following command:

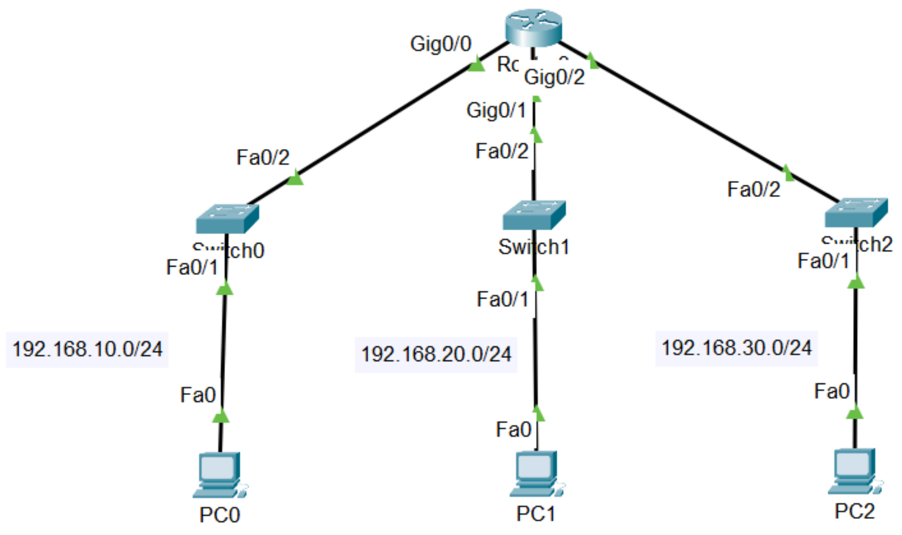
```
Router(config)# ip nat inside source list access_list_number pool
                         POOL_NAME
```

After that, like the static NAT, you have to specify the *inside* and *outside* interfaces.

Port Address Translation (PAT)
For dynamic NAT, in the worst case, you'd need as many global IP addresses as the internal hosts. This is not plausible in most circumstances where you've limited global IP and hundreds of local hosts. Its where PAT comes in. PAT establishes many-to-one mapping between multiple local hosts and a single global IP address. It uses the Port (TCP/UDP port) information to distinguish between different internal hosts and assign a single global IP to all those addresses thus greatly conserving the global address pool. The configuration of PAT is almost similar to dynamic NAT except you just have to add the **`overload`** keyword at the very end while specifying the relation between access list and nat pool. The command format is below:

```
3. Router(config)# ip nat inside source list access_list_number pool
                    POOL_NAME overload
```

4. Configure ACL:



I. **Configure Router Interfaces**
Router(config)# int g0/0

```
Router(config-if)# ip address 192.168.10.1 255.255.255.0
Router(config-if)# no shutdown
Router(config)# int g0/1
Router(config-if)# ip address 192.168.20.1 255.255.255.0
Router(config-if)# no shutdown
Router(config)# int g0/2
Router(config-if)# ip address 192.168.20.1 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)# exit
Router# copy running-config startup-config
```

II. **Configure PC0**

```
IP: 192.168.10.5
Mask: 255.255.255.0
Gateway: 192.168.10.1
```

III. **Configure PC1**

```
IP: 192.168.20.5
Mask: 255.255.255.0
Gateway: 192.168.20.1
```

IV. **Configure PC2**

```
IP: 192.168.30.5
Mask: 255.255.255.0
Gateway: 192.168.30.1
```

V. **Define ACL**

```
Router(config)# access-list 1 deny 192.168.10.0 0.0.0.255
Router(config)# access-list 1 permit any
```

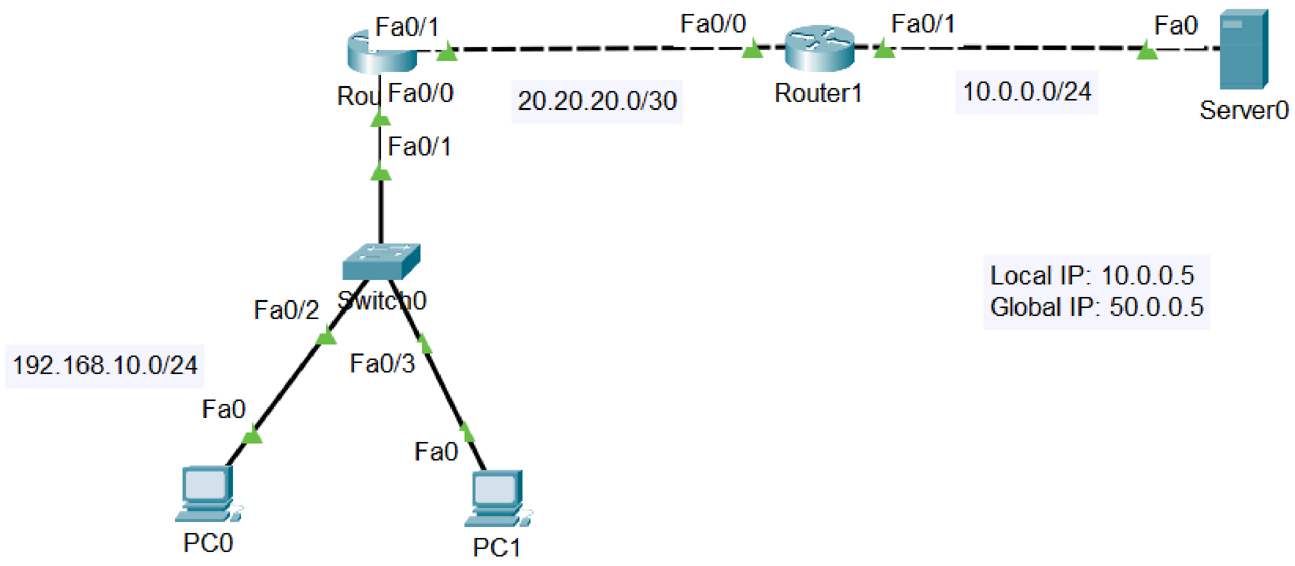
VI. **Verify ACL**

```
Router# show access-lists
```

VII. **Apply ACL**

```
Router(config)# interface gigabitEthernet 0/2
Router(config-if)# ip access-group 1 out
```

5. Configure NAT:



VIII. **Configure Router0 Interfaces**

```
Router(config)# int fa0/0
Router(config-if)# ip address 192.168.10.1 255.255.255.0
Router(config-if)# no shutdown
Router(config)# int fa0/1
Router(config-if)# ip address 20.20.20.1 255.255.255.252
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# ip route 50.0.0.0 255.255.255.0 20.20.20.2
Router# copy running-config startup-config
```

IX. **Configure Router1 Interfaces**

```
Router(config)# int fa0/1
Router(config-if)# ip address 10.0.0.1 255.255.255.0
Router(config-if)# no shutdown
Router(config)# int fa0/0
Router(config-if)# ip address 20.20.20.2 255.255.255.252
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# ip route 192.168.10.0 255.255.255.0 20.20.20.1
Router# copy running-config startup-config
```

X. **Configure PC0**

```
IP: 192.168.10.5
Mask: 255.255.255.0
Gateway: 192.168.10.1
```

XI. **Configure PC1**

```
IP: 192.168.10.10
Mask: 255.255.255.0
Gateway: 192.168.10.1
```

XII. **Configure Server0**

```
IP: 10.0.0.05
Mask: 255.255.255.0
Gateway: 10.0.0.1
```

XIII. **Enable static NAT insider Router1**

```
Router(config)# ip nat inside source static 10.0.0.5 50.0.0.5
Router(config)# int fa0/1
Router(config-if)# ip nat inside
Router(config)# int fa0/0
Router(config-if)# ip nat outside
Router# copy running-config startup-config
```

XIV. **Verify NAT**

```
Router# show ip nat translations
Router# show ip nat statistics
```

6. Tasks:

I. You will configure *numbered standard ACL* following the instructions given in the task. The task description for this task is provided in the pdf ***Task-1_configure-standard-ipv4-acls***. You're provided a .pka file for this task.

II. You will configure *dynamic NAT and PAT* following the instructions given in the task. The task description for this task is provided in the pdf ***Task-2_configure-NAT***. You're provided a .pka file for this task.