Spring 2014
Computer Networks
CMPE323

Laboratory Experiment No. 8: Introduction to NAPT

Aims and Objectives:

- Introducing Network Address and Port Translation (NAPT).

Materials Required:

- IP routers,
- PCs with Ethernet adapters,
- and straight-through/crossover/rollover cables.

Change Log:

- 15-4-2014: original document – mkhonji.
- 23-4-2014: added PC3, changed lab no. from 7 to 8 – mkhonji.

1 Introduction

IPv4 addresses are 32 bit numbers which gives us 2^{32} unique addresses (some of which are not usable for Internet communication as they are reserved for other purposes).

Although the 2^{32} addresses might seem to be large at first, the amount of growth that the Internet has experienced demands even a larger address space. This became apparent in the 1980s when the fears from depleted IPv4 address surfaced.

In order to handle the shortage in the IPv4 address space, a number of solutions were proposed at the same time, such as:

- Larger IP address space (e.g. IPv6).
- Classless Inter-Domain Routing instead of classfull routing.
- Network Address Translation (NAT).

It is highly likely that if you used the Internet, you have already used Network Address and Port Translation (NAPT), which is the most wide-spread NAT type.

There are other types of NAT, such as: static NAT, dynamic NAT, source NAT, destination NAT, ... However, in this lab, we will only focus on the most common NAT type which is NAPT.

Note: it is important to note that although the original motives for NAT were to postpone the depletion of IPv4 addresses, NAT is used for many other scenarios as well, such as solving some IP address conflict scenarios and IPv4 to IPv6 migration.

1.1 NAPT

In order to save public IP addresses, two perquisites are needed:

- Private IP addresses.
- , and a NAPT-ready device (often integrated in routers).

1.1.1 Private IPv4 addresses

RFC1918 defines the following ranges (or blocks) of private IP addresses:

- 10.0.0.0 – 10.255.255.255 (named as the 24-bit private address block, historically known as the class A private IP address block).
- 172.16.0.0 – 172.31.255.255 (named as the 20-bit private address block, historically known as the class B private IP address block)
- 192.168.0.0 – 192.168.255.255 (named as the 16-bit private address block, historically known as the class C address block).

The interesting property of private IP addresses is that they are not used in the Internet but rather reserved for local networks only.

This way, different local networks can re-use the same IP address blocks as long as they are not inter-connected directly without address translation.

For example, if 100 computers in company X and 100 other computers in company Y demand access to the Internet, they can all use the same address block (e.g. 24-bit block) and till being able to access the Internet via NAPT (as covered in the next section). This way, we can greatly save IPv4 addresses.

1.1.2 NAPT-ready devices

This subsection will explain how a NAPT device can allow many computers with private IP addresses access the Internet using only a single public IP address.

As depicted in Figure 1, a number of computers in a private network (one that uses the 10.0.0.0/8 24-bit block) require access to the Internet via some Internet Service Provider (ISP).

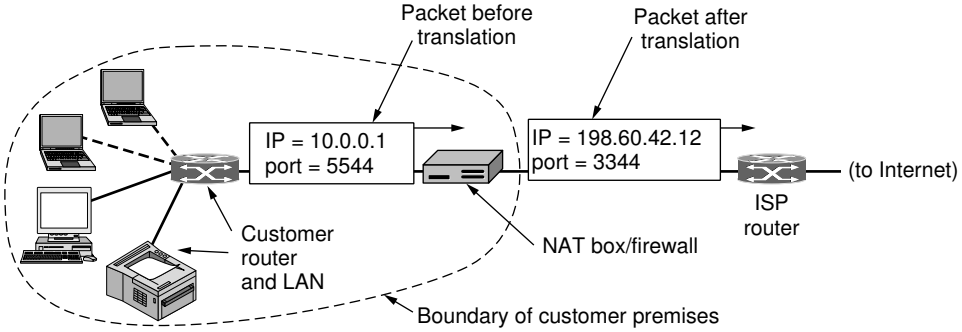

Figure 1: A Network Address and Port Translation (NAPT) scenario — Source: Computer Networks 5^{th} Edition, 2010, by Andrew S. Tanenbaum and David J. Wetherall, page 453.

In order to perform NAPT, the following steps are performed:

1. A computer with the IP 10.0.0.1 in the private network 10.0.0.0/8 sends a packet to some resource in the Internet (e.g. TCP request to port 80 on 173.194.39.39 which is a Google web server). Therefore, the source IP address is 10.0.0.1, the source TCP port is some dynamically chosen number by the computer's TCP implementation (we will assume the source port is 50,000), the destination IP address is 173.194.39.39 (the IP of some Google web server), and the destination port number is 80 (often used for HTTP web servers).

2. Some Internet Service Provider (ISP) has assigned a single public IP address to the local network, and this IP address is 198.60.42.12.

3. As the computer is not connected to any network that has the IP address 173.194.39.39, it will forward the packet to its default gateway (the local router).

4. Before the packet is forwarded to the ISP router (to deliver the packet to its destination), NAPT translation occurs as follows:

(a) The packet is modified such that its source IP address 10.0.0.1 is changed to 198.60.42.12.

(b) An entry is added to the *NAT translation table* of the NAPT device which maps the source IP address 10.0.0.1 and the source TCP port 50,000 to the public IP address 198.60.42.12 with the same source TCP port 50,000. **Note:** if the combination 198.60.42.12:50000 exists in the table, the port is translated to some other port number such as 50,001. The goal is ensuring that different connections have unique `publicIP:port` mappings.

5. The modified packet is sent to the ISP which in turn forwards it to other routers that will ultimately reach the target destination 173.194.39.39 on the TCP 80. The receiver of this packet will notice that the source IP address is 198.60.42.12 (instead of 10.0.0.1). In other words, we can think of the NAPT translation entry as `TCP: 10.0.0.1:50000 --> 198.60.42.12:50000`.

6. If the Google web server 198.60.42.12 responded back (which most likely it will given how HTTP works), it will send a packet with a destination IPv4 address of 198.60.42.12 and a destination port number of 50,000.

7. When the NAPT device gets the response from the Google web server, the NAPT device will lookup its NAT translation table for a match for `198.60.42.12:5000`.

8. The NAT table lookup will succeed and will identify that the combination `198.60.42.12:50000` was originally `10.0.0.1:5000`.

9. the NAPT device will reverse the translation back by modifying the destination IP address 198.60.42.12 and port 50,000 back to the original one which are the IP 0.0.0.1 and the port 50,000.

This way, theoretically, each public IP address can be used by 2^{16} TCP or UDP connections. The number 2^{16} is due to the fact that TCP and UDP port numbers are 16 bit numbers.

2 Lab Preparation

1. Erase the configuration of `R1`[1] and reboot `PC1`, `PC2` and `PC3`.

2. Physically connect and configure IP addresses of the router[2] and the PCs[3] as depicted in Figure 2.

3. Add default routes[4] to `PC1`, `PC2` and `PC3`.

[1] `enable`, `erase startup-config`, `reload`, and make sure to answer `no` to all yes/no questions while hitting *enter* for all `confirm` prompts.

[2] `enable`, `configure terminal`, `interface Gi0/0`, `ip address <ip_address> <subnet_mask>`, `no shutdown`.

[3] `ifconfig eth0 <ip_address>/<netmask_bits>`

[4] `route add -net 0.0.0.0/0 gw <nexthop_ip>`

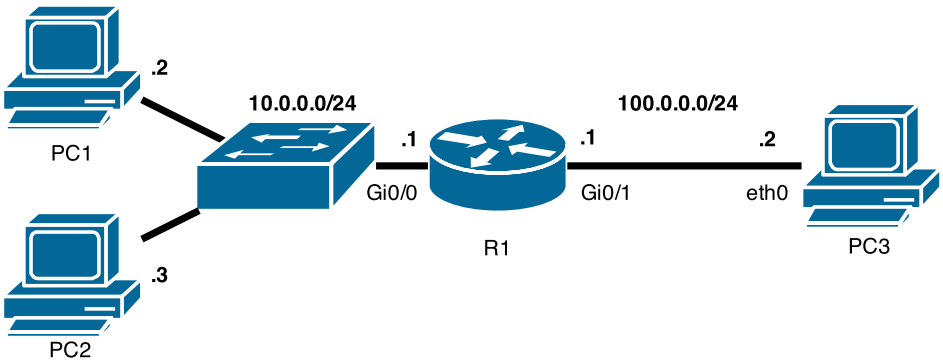

Figure 2: Lab topology.

4. Run Wireshark on all the PCs.
5. Using the `nc`[5] command run three applications on `PC3` that listen on TCP ports as follows:
   - Application 1: listen on IP 100.0.0.2 and TCP port 100.
6. Using the `nc`[6] command run three applications on `PC3` that listen on UDP ports as follows:
   - Application 2: listen on IP 100.0.0.2 and UDP port 100.

3 Lab experiments

3.1 Configuring NAPT on `R1`

Connect to `R1` using the console port and perform the following configurations:

- Configure an Access-Control List (ACL) that specifies which IP addresses to be matched. Note that `0.0.0.255` is called a *wild-card mask* which is semantically the same as the subnet mask `255.0.0.0`. Note that a wild-card mask is the opposite of a subnet mask. The reason for such difference in the Cisco IOS is historical and beyond the lab's scope.
  1. `enable`.
  2. `config terminal`.
  3. `ip access-list standard <list_name>`.
  4. `permit 10.0.0.0 0.0.0.255`.
  5. `end`.
- Configure `R1` so that it knows which interface faces the inside (possibly private) network and which interface faces the outside (possibly public) network.

[5] `nc -l -p <port> -s <ip_address>`
[6] `nc -l -u -p <port> -s <ip_address>`

1. `enable`.
2. `config terminal`.
3. `int gi0/0`.
4. `ip nat inside`.
5. `exit`.
6. `int gi0/1`.
7. `ip nat outside`.
8. `ip nat inside`.
9. `end`.

- Enable NAPT such that, for any IP packet that enters an inside interface that is also going to be routed to the outside interface, will have the NAPT translation applied against (as described earlier).

1. `enable`.
2. `config terminal`.
3. `ip nat inside source list <list_name> interface fa0/1 overload`.
4. `end`.

3.2 Analysing packets

[100 points]

Note: when done, show your work to the lab engineer for grading purposes.

While monitoring Wireshark instances on relevant PCs, use `PC1` to perform the following tasks:

1. View the *NAT translation table* of `R1` by `show ip nat translations`. It should be empty.
2. Using PC1, connect to `PC3`'s `nc` server that is listening on the TCP port 100 such that the source port is `50000`. Use the command `nc -p 5000 100.0.0.2 100`. By observing Wireshark, note down the following information:
   - Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC1`'s Wireshark instance.
   - Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC3`'s Wireshark instance.
3. View the *NAT translation table* of `R1` by `show ip nat translations` and compare its content against the information that you have gathered above. You should see new entries. How do the entries relate to the data that you have observed in the previous point?
4. Using PC1, connect to `PC3`'s `nc` server that is listening on the UDP port 100 such that the source port is `50000`. Use the command `nc -u -p 5000 100.0.0.2 100`. By observing Wireshark, note down the following information:

- Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC1`'s Wireshark instance.
- Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC3`'s Wireshark instance.

5. View the *NAT translation table* of `R1` by `show ip nat translations` and compare its content against the information that you have gathered above. You should see new entries. How do the entries relate to the data that you have observed in the previous point?

6. Using PC2, connect to `PC3`'s `nc` server that is listening on the TCP port 100 such that the source port is `50000`. Use the command `nc -p 5000 100.0.0.2 100`. By observing Wireshark, note down the following information:

   - Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC2`'s Wireshark instance.
   - Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC3`'s Wireshark instance.

7. View the *NAT translation table* of `R1` by `show ip nat translations` and compare its content against the information that you have gathered above. You should see new entries. How do the entries relate to the data that you have observed in the previous point?

8. Using PC2, connect to `PC3`'s `nc` server that is listening on the UDP port 100 such that the source port is `50000`. Use the command `nc -u -p 5000 100.0.0.2 100`. By observing Wireshark, note down the following information:

   - Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC2`'s Wireshark instance.
   - Source IP, layer 4 protocol, source port, destination IP and destination port numbers as it appears on `PC3`'s Wireshark instance.

9. View the *NAT translation table* of `R1` by `show ip nat translations` and compare its content against the information that you have gathered above. You should see new entries. How do the entries relate to the data that you have observed in the previous point?

10. Then answer the following questions:

    - Which IP addresses are being translated (source or destination)?
    - Which port numbers are being translated (source or destination)?
    - What are the cases when the translated port numbers are identical to the pre-translation port numbers?
    - What are the cases when the translated port numbers are different to the pre-translation port numbers?
    - How can NAPT save IPv4 addresses?

3.3 Extra questions (not graded)

- For protocols that do not use TCP or UDP port numbers, how can NAPT function? For example, ICMP does not rely on TCP nor UDP yet it functions correctly. In other words, how can the NAPT device identify which ICMP Echo-Reply is for which ICMP Echo (request) from which PC or process?

- What is the behaviour of a Cisco router if its NAT translation table is full?