

## 1. 网络地址转换实验报告

1. 一、实验题目：网络地址转换实验
2. 二、实验内容：
3. 三、实验过程：
4. 四、实验结果：
5. 五、思考题

# 网络地址转换实验报告

---

热伊莱·图尔贡 2018K8009929030

## 一、实验题目：网络地址转换实验

---

## 二、实验内容：

---

### 1. 实验内容一：SNAT实验

- 运行给定网络拓扑(nat\_topo.py)
- 在n1, h1, h2, h3上运行相应脚本 n1: disable\_arp.sh, disable\_icmp.sh, disable\_ip\_forward.sh, disable\_ipv6.sh h1-h3: disable\_offloading.sh, disable\_ipv6.sh
- 在n1上运行nat程序： n1# ./nat exp1.conf
- 在h3上运行HTTP服务： h3# python ./http\_server.py
- 在h1, h2上分别访问h3的HTTP服务 h1# wget http://159.226.39.123:8000 h2# wget http://159.226.39.123:8000

### 2. 实验内容二：DNAT实验

- 运行给定网络拓扑(nat\_topo.py)
- 在n1, h1, h2, h3上运行相应脚本 n1: disable\_arp.sh, disable\_icmp.sh, disable\_ip\_forward.sh, disable\_ipv6.sh h1-h3: disable\_offloading.sh, disable\_ipv6.sh
- 在n1上运行nat程序： n1# ./nat exp2.conf
- 在h1, h2上分别运行HTTP Server： h1/h2# python ./http\_server.py
- 在h3上分别请求h1, h2页面 h3# wget http://159.226.39.43:8000 h3# wget http://159.226.39.43:8001

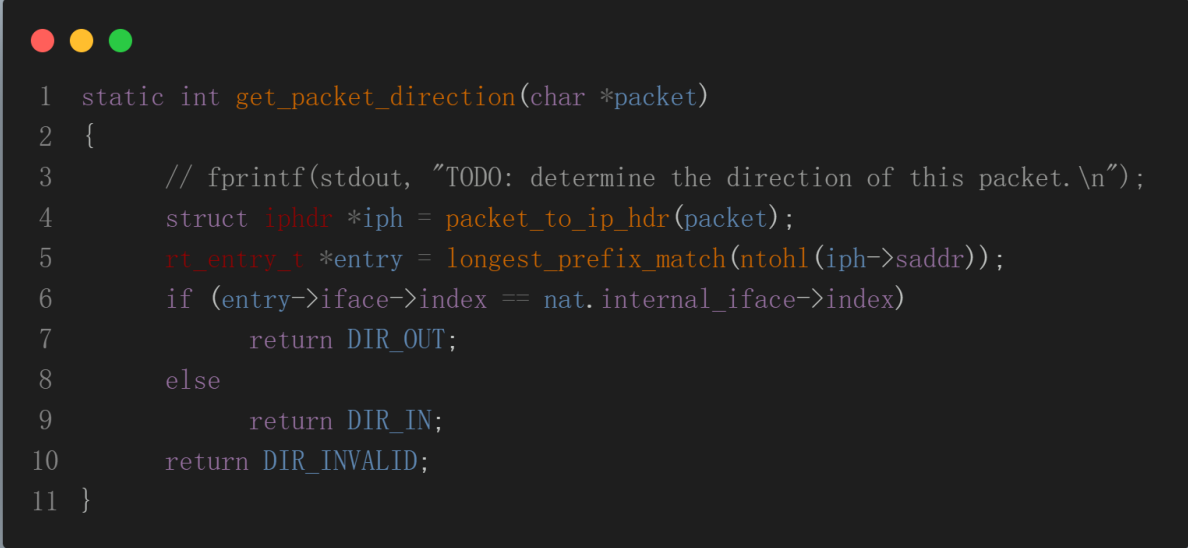
### 3. 实验内容三：手动构造一个包含两个nat的拓扑

- $h1 \leftrightarrow n1 \leftrightarrow n2 \leftrightarrow h2$
- 节点n1作为SNAT， n2作为DNAT， 主机h2提供HTTP服务， 主机h1穿过两个nat连接到h2并获取相应页面

## 三、实验过程：

---

### 1. 区分数据包方向



```
1 static int get_packet_direction(char *packet)
2 {
3     // fprintf(stdout, "TODO: determine the direction of this packet.\n");
4     struct iphdr *iph = packet_to_ip_hdr(packet);
5     rt_entry_t *entry = longest_prefix_match(ntohl(iph->saddr));
6     if (entry->iface->index == nat.internal_iface->index)
7         return DIR_OUT;
8     else
9         return DIR_IN;
10    return DIR_INVALID;
11 }
```

### 2. 合法数据包

- 数据包的方向为DIR\_IN

```

1  if (dir == DIR_IN)
2      {
3          list_for_each_entry(entry, &nat.nat_mapping_list[index], list)
4          {
5              if (entry->external_ip == ntohl(iph->daddr) && entry->external_port == ntohs(tcp->dport))
6              {
7                  found = 1;
8                  break;
9              }
10         }
11     if (!found)
12     {
13         struct dnat_rule *rule_entry = NULL;
14         list_for_each_entry(rule_entry, &nat.rules, list)
15         {
16             if (ntohl(iph->daddr) == rule_entry->external_ip && ntohs(tcp->dport) == rule_entry->external_port)
17             {
18                 find = 1;
19                 break;
20             }
21         }
22         if (find)
23         {
24             entry = (struct nat_mapping *)malloc(sizeof(struct nat_mapping));
25             memset(entry, 0, sizeof(struct nat_mapping));
26             entry->internal_ip = rule_entry->internal_ip;
27             entry->internal_port = rule_entry->internal_port;
28             entry->external_ip = rule_entry->external_ip;
29             entry->external_port = rule_entry->external_port;
30             entry->remote_ip = rm_ip;
31             entry->remote_port = rm_port;
32             entry->update_time = time(NULL);
33             list_add_tail(&entry->list, &nat.nat_mapping_list[index]);
34         }
35         else
36         {
37             pthread_mutex_unlock(&nat.lock);
38             return NULL;
39         }
40     }
41     tcp->dport = htons(entry->internal_port);
42     iph->daddr = htonl(entry->internal_ip);
43     entry->conn.external_fin = (tcp->flags == TCP_FIN) ? TCP_FIN : 0;
44     if (ntohl(tcp->seq) > entry->conn.external_seq_end)
45         entry->conn.external_seq_end = ntohl(tcp->seq);
46     if (ntohl(tcp->ack) > entry->conn.external_ack && tcp->flags == TCP_ACK)
47         entry->conn.external_ack = ntohl(tcp->ack);
48 }

```

- 数据包的方向为DIR\_OUT

```

1  else if (dir == DIR_OUT)
2      {
3          list_for_each_entry(entry, &nat.nat_mapping_list[index], list)
4          {
5              if (entry->internal_ip == ntohl(iph->saddr) && entry->internal_port == ntohs(tcph->sport))
6              {
7                  found = 1;
8                  break;
9              }
10         }
11     if (!found)
12     {
13         entry = (struct nat_mapping *)malloc(sizeof(struct nat_mapping));
14         memset(entry, 0, sizeof(struct nat_mapping));
15         entry->internal_ip = ntohl(iph->saddr);
16         entry->internal_port = ntohs(tcph->sport);
17         entry->external_ip = nat.external_iface->ip;
18         entry->external_port = alloc_port();
19         entry->remote_ip = rm_ip;
20         entry->remote_port = rm_port;
21         entry->update_time = time(NULL);
22         list_add_tail(&entry->list, &nat.nat_mapping_list[index]);
23     }
24     tcph->sport = htons(entry->external_port);
25     iph->saddr = htonl(entry->external_ip);
26     if (tcph->flags == TCP_FIN)
27         entry->conn.internal_fin = TCP_FIN;
28     if (ntohl(tcph->seq) > entry->conn.internal_seq_end)
29         entry->conn.internal_seq_end = ntohl(tcph->seq);
30     if (ntohl(tcph->ack) > entry->conn.internal_ack && tcph->flags == TCP_ACK)
31         entry->conn.internal_ack = ntohl(tcph->ack);
32 }

```

### 3. NAT老化（Timeout）操作

```
1 void *nat_timeout()
2 {
3     while (1)
4     {
5         // fprintf(stdout, "TODO: sweep finished flows periodically.\n");
6         pthread_mutex_lock(&nat.lock);
7         for (int i = 0; i < HASH_8BITS; i++)
8         {
9             struct nat_mapping *entry, *q;
10            list_for_each_entry_safe(entry, q, &nat.nat_mapping_list[i], list)
11            {
12                if (is_flow_finished(&entry->conn) ||
13                    time(NULL) - entry->update_time > TCP_ESTABLISHED_TIMEOUT)
14                {
15                    nat.assigned_ports[entry->external_port] = 0;
16                    list_delete_entry(&entry->list);
17                    free(entry);
18                }
19            }
20        }
21        pthread_mutex_unlock(&nat.lock);
22        sleep(1);
23    }
24
25    return NULL;
26 }
```

## 四、实验结果：

---

### 1. 实验内容一：SNAT实验

"Node: h1"

```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# wget http://159.226.39.123:8000
--2022-05-17 15:18:17-- http://159.226.39.123:8000/
Connecting to 159.226.39.123:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 212 [text/html]
Saving to: 'index.html.3'

index.html.3      100%[=====>]      212  --.-KB/s    in 0s
2022-05-17 15:18:17 (64.9 MB/s) - 'index.html.3' saved [212/212]
```

"Node: h2"

```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# wget http://159.226.39.123:8000
--2022-05-17 15:18:14-- http://159.226.39.123:8000/
Connecting to 159.226.39.123:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 212 [text/html]
Saving to: 'index.html.2'

index.html.2      100%[=====>]      212  --.-KB/s    in 0s
2022-05-17 15:18:14 (46.1 MB/s) - 'index.html.2' saved [212/212]
```

```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# python2 ./http_server.py
Serving HTTP on 0.0.0.0 port 8000 ...
159.226.39.43 - - [17/May/2022 15:18:14] "GET / HTTP/1.1" 200 -
159.226.39.43 - - [17/May/2022 15:18:17] "GET / HTTP/1.1" 200 -
```

```
rayilam@rayilam-VirtualBox:~/CN/08-nat$ cat index.html.2
```

```
<!doctype html>
<html>
    <head> <meta charset="utf-8">
        <title>Network IP Address</title>
    </head>
    <body>
        My IP is: 159.226.39.123
        Remote IP is: 159.226.39.43
    </body>
</html>
```

```
rayilam@rayilam-VirtualBox:~/CN/08-nat$ cat index.html.3
```

```
<!doctype html>
<html>
    <head> <meta charset="utf-8">
        <title>Network IP Address</title>
    </head>
    <body>
        My IP is: 159.226.39.123
        Remote IP is: 159.226.39.43
    </body>
</html>
```

## 2. 实验内容二：DNAT实验

```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# wget http://159.226.39.43:8000
--2022-05-17 15:25:06-- http://159.226.39.43:8000/
Connecting to 159.226.39.43:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 208 [text/html]
Saving to: 'index.html'
```

```
index.html          100%[=====>]          208  --.-KB/s    in 0s
2022-05-17 15:25:06 (59.2 MB/s) - 'index.html' saved [208/208]
```

```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# wget http://159.226.39.43:8001
--2022-05-17 15:25:18-- http://159.226.39.43:8001/
Connecting to 159.226.39.43:8001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 208 [text/html]
Saving to: 'index.html.1'
```

```
index.html.1        100%[=====>]          208  --.-KB/s    in 0s
2022-05-17 15:25:18 (45.5 MB/s) - 'index.html.1' saved [208/208]
```

### "Node: h1"

```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# python2 ./http_server.py
Serving HTTP on 0.0.0.0 port 8000 ...
159.226.39.123 - - [17/May/2022 15:25:06] "GET / HTTP/1.1" 200 -
```



```
root@rayilam-VirtualBox:/home/rayilam/CN/08-nat# python2 ./http_server.py
Serving HTTP on 0.0.0.0 port 8000 ...
159.226.39.123 - - [17/May/2022 15:25:18] "GET / HTTP/1.1" 200 -
```

```
rayilam@rayilam-VirtualBox:~/CN/08-nat$ cat index.html
```

```
<!doctype html>
<html>
    <head> <meta charset="utf-8">
        <title>Network IP Address</title>
    </head>
    <body>
        My IP is: 10.21.0.1
        Remote IP is: 159.226.39.123
    </body>
</html>
```

```
rayilam@rayilam-VirtualBox:~/CN/08-nat$ cat index.html.1
```

```
<!doctype html>
<html>
    <head> <meta charset="utf-8">
        <title>Network IP Address</title>
    </head>
    <body>
        My IP is: 10.21.0.2
        Remote IP is: 159.226.39.123
    </body>
</html>
```

### 3. 实验内容三：手动构造一个包含两个nat的拓扑

```
--2022-05-17 16:23:23-- http://159.226.39.123:8000/
Connecting to 159.226.39.123:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 207 [text/html]
Saving to: 'index.html'

index.html          100%[=====]          207  --.-KB/s   in 0s

2022-05-17 16:23:23 (44.2 MB/s) - 'index.html' saved [207/207]

Serving HTTP on 0.0.0.0 port 8000 ...
159.226.39.43 - - [17/May/2022 16:23:23] "GET / HTTP/1.1" 200 -
```

```
<!doctype html>
<html>
    <head> <meta charset="utf-8">
        <title>Network IP Address</title>
    </head>
    <body>
        My IP is: 10.11.0.1
        Remote IP is: 159.226.39.43
    </body>
</html>
```

## 五、思考题

1. 实验中的NAT系统可以很容易实现支持UDP协议，现实网络中NAT还需要对ICMP进行地址翻译，请调研说明NAT系统如何支持ICMP协议。

ICMP协议没有端口号，因此无法像UDP协议一样，通过<IP, 端口号>建立唯一映射。ICMP协议报文格式如下：

类型 (Type)	代码 (Code)	校验和 (Checksum)
标识符 (Identifier)		序列号 (Sequence number)
选项 (Option)		

假设主机结点h1要ping结点h2，h1发送ICMP报文，h1会根据报文头部中包含的类型信息 **Type** 和代码信息 **Code**生成源端口号，根据报文头部中包含的标识符信息 **Identifier** 生成目的端口号，nat路由器收到包后，**Type+Code**作为内网端口，**IDENTIFIER**作为外网端口，生成响应nat表项，然后将ICMP报头的源端口和目的端口进行相应修改，在结点 h2 收到 ICMP 报文后，生成 ICMP 响应报文，**Type+Code**作为源端口，**IDENTIFIER**作为目的端口。nat路由器结点收到响应报文后，查找 nat表项并且确定应该转发给私网中的结点 h1 。两个结点完成一次来回通信。

2. 给定一个有公网地址的服务器和两个处于不同内网的主机，如何让两个内网主机建立TCP连接并进行数据传输。（提示：不需要DNAT机制）让两个处于不同nat网络下的结点直接建立连接，称为nat穿透。此时每个结点需要知道自己的内网地址和对方的公网地址端口，从而发送数据包、建立连接。可以通过发送一个数据包给

**server**，**server**通过解析和查表，获得结点的公网地址，再将地址回传给节点的方法发现自己的公网地址端口，然后通过第三方**server**来交换双方的公网地址端口，从而建立连接。