

# 第十一周实验报告

## 网络地址转换实验

2015K8009922021

李一苇

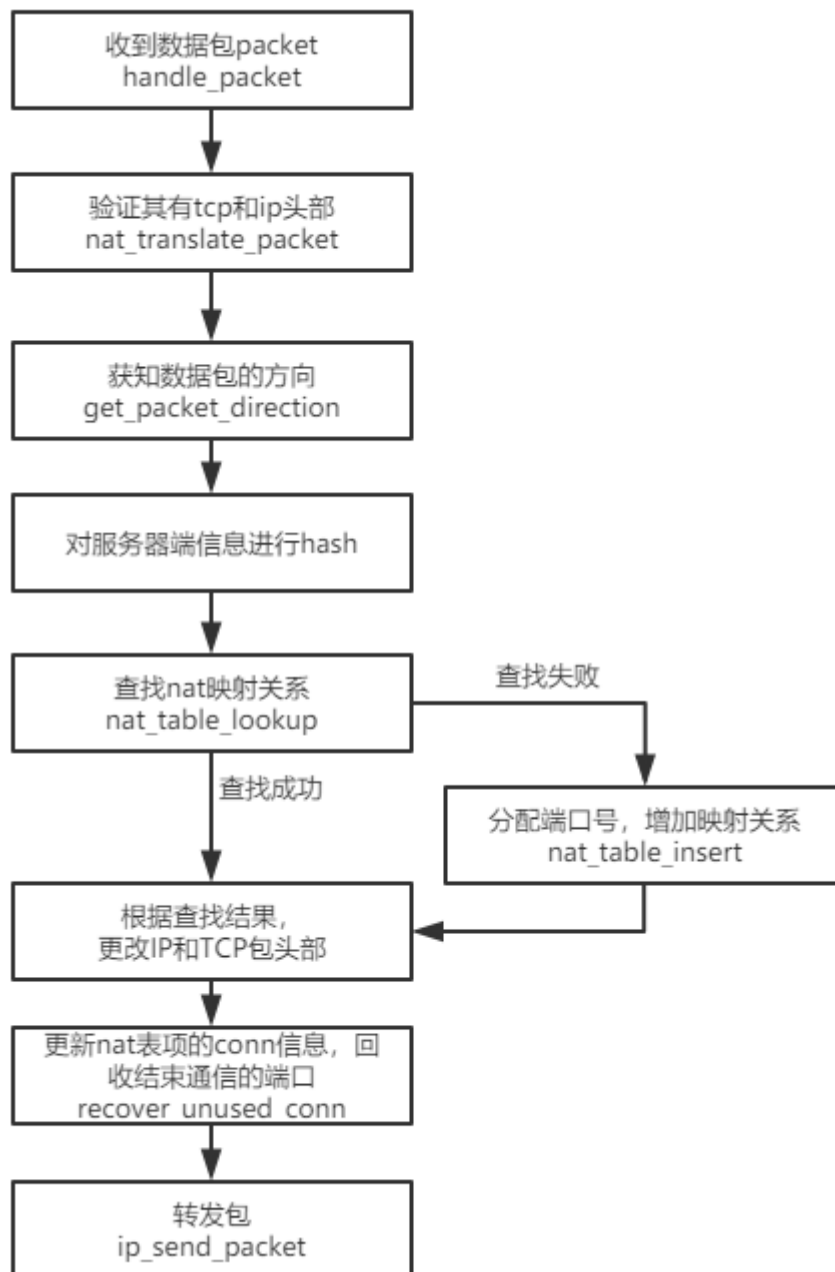
---

### 一、实验内容

- NAT映射表管理
  - 维护NAT连接映射表，支持映射的添加、查找、更新和老化操作
- 数据包的翻译操作
  - 对到达的合法数据包，进行IP和Port转换操作，更新头部字段，并转发数据包
  - 对于到达的非法数据包，回复ICMP Destination Host Unreachable

### 二、实验流程

网络地址转换的流程图如下，本实验按照流程图依次实现各函数：



1. 收到数据包，验证其头部结构已经由老师实现：

2. 获知数据包的方向：

注意判断DIR\_IN的条件之一是，目的IP和external\_iface的IP直接相等，不用查找最长前缀了。

```

static int get_packet_direction(char *packet)
{
    struct iphdr *ip = packet_to_ip_hdr(packet);
    u32 dst = ntohl(ip->daddr);
    u32 src = ntohl(ip->saddr);
    rt_entry_t *dst_entry = longest_prefix_match(dst);
    rt_entry_t *src_entry = longest_prefix_match(src);
    if ((src_entry->iface->ip == nat.internal_iface->ip) && (dst_entry->iface->ip ==
nat.external_iface->ip)) {
        return DIR_OUT;
    }
}
  
```

```

    if ((src_entry->iface->ip == nat.external_iface->ip) && (dst == nat.external_iface->ip)) {
        return DIR_IN;
    }
    return DIR_INVALID;
}

```

### 3. hash操作

```

u32 srv_ip = dir == DIR_OUT? ntohs(ip->daddr) : ntohs(ip->saddr);
u16 srv_port = dir == DIR_OUT? ntohs(tcp->dport) : ntohs(tcp->sport);
memcpy(srv_info, &srv_ip, 4);
memcpy(srv_info + 4, &srv_port, 2);
u8 hash_srvinfo = hash8(srv_info, 6);

```

当方向为DIR\_IN时，服务器的信息为源地址；当方向为DIR\_OUT时，服务器信息为目的地址，拼接后得到hash值

### 4. 查找nat表项:

表头为 `&nat.nat_mapping_list[srv_hash]`

当方向为DIR\_IN时，应匹配外部的IP和端口

当方向为DIR\_OUT时，应匹配内部的IP和端口

```

list_for_each_entry(mapping_entry, head, list) {
    if (dir == DIR_OUT) {
        if (mapping_entry->internal_ip == ip && mapping_entry->internal_port ==
port) {
            return mapping_entry;
        }
    }
    else {
        if (mapping_entry->external_ip == ip && mapping_entry->external_port ==
port) {
            return mapping_entry;
        }
    }
}

```

### 5. 插入nat表项

新建mapping\_entry后插入，注意新表项的外部IP为 `nat.external_ip`，外部端口值为 `assign_external_ports()` 得到的值

### 6. 修改IP和TCP包头部

具体地说，当方向为DIR\_IN时，修改ip的daddr和tcp的dport；否则修改ip的saddr和tcp的sport  
与此同时还应修改tcp和ip的checksum

### 7. 更新nat\_mapping表项

需更新 `entry->update_time = time(NULL);`

更新连接状态:

```

void recover_unused_conn(struct nat_mapping *pos, struct tcphdr *tcp, int dir){
    if(tcp->flags == TCP_FIN + TCP_ACK) {
        if (dir == DIR_IN)
            pos->conn.internal_fin = 1;
        else
            pos->conn.external_fin = 1;
    }

    if(tcp->flags == TCP_RST){
        printf("RST!\n");
        if((pos->conn).internal_fin + (pos->conn).external_fin == 2){
            nat.assigned_ports[pos->external_port] = 0;
            list_delete_entry(&(pos->list));
            free(pos);
        }
    }
}

```

8. 发送包：由 `ip_send_packet()` 直接实现。

### 三、实验结果和分析

验证方法（运行脚本 `nat_topo.py`）：

- 在n1上nat，进行数据包的处理
- 在h2上运行HTTP服务
  - 执行 `python -m SimpleHTTPServer`，启动HTTP服务
- 在h1上访问h2的HTTP服务
  - h1 # `wget http://159.226.39.123:8000`

结果如下：

h1和h2成功建立TCP连接，收到h2的数据包并保存

```

Node: h1
23:8000
--2018-11-17 15:27:55-- http://159.226.39.123:8000/
正在连接 159.226.39.123:8000... 已连接。
已发出 HTTP 请求，正在等待回... 200 OK
长度：1080 (1.1K) [text/html]
正在保存至: "index.html.1"

index.html.1      100%[=====>]  1.05K  --.-KB/s   in 0s
2018-11-17 15:27:55 (153 MB/s) - 已保存 "index.html.1" [1080/1080]

```

抓包截图如下：

h1的分组1表明发出TCP的SYN请求，与此同时h2的分组3收到该请求，发现分组3的源IP地址变为NAT设备的外部interface的IP地址，且源端口变为12345（即最小的可分配端口值）

h2的分组4表明回应该请求（SYN，ACK），h1的分组4表明收到该回应，且该分组的目的地址经由NAT设备转换，变为h1的IP地址

之后的过程与上文类似，可见NAT设备成功进行了地址的转换。

正在捕获 h1-eth0

文件 编辑 视图 窗口 帮助

SSSS

Time

Source

Destination

Protocol

Length

Info

1	0.000000000	10.21.0.1	159.226.39.123	TCP	74	33180 → 8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460
2	0.000076339	ba:a1:72:ac:41:7b	Broadcast	ARP	42	Who has 10.21.0.1? Tell 10.21.0.254
3	0.000078431	2e:fe:df:ac:93:30	ba:a1:72:ac:41:7b	ARP	42	10.21.0.1 is at 2e:fe:df:ac:93:30
4	0.000083429	159.226.39.123	10.21.0.1	TCP	74	8000 → 33180 [SYN, ACK] Seq=9 Ack=1 Win=28696 Len=0
5	0.000091686	10.21.0.1	159.226.39.123	TCP	66	33180 → 8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0
6	0.000478762	10.21.0.1	159.226.39.123	HTTP	212	GET / HTTP/1.1
7	0.000748460	159.226.39.123	10.21.0.1	TCP	66	8000 → 33180 [ACK] Seq=1 Ack=147 Win=30208 Len=0
8	0.000402121	159.226.39.123	10.21.0.1	TCP	83	8000 → 33180 [PSH, ACK] Seq=1 Ack=147 Win=30208 Len=0
9	0.000415624	10.21.0.1	159.226.39.123	TCP	66	33180 → 8000 [ACK] Seq=147 Ack=18 Win=29696 Len=0
10	0.000442134	159.226.39.123	10.21.0.1	TCP	166	8000 → 33180 [PSH, ACK] Seq=18 Ack=147 Win=30208 Len=0
11	0.000445343	10.21.0.1	159.226.39.123	TCP	66	33180 → 8000 [ACK] Seq=147 Ack=118 Win=29696 Len=0
12	0.000691732	159.226.39.123	10.21.0.1	TCP	88	8000 → 33180 [PSH, ACK] Seq=118 Ack=147 Win=30208 Len=0
13	0.000907467	10.21.0.1	159.226.39.123	TCP	66	33180 → 8000 [ACK] Seq=147 Ack=140 Win=29696 Len=0
14	0.000716938	159.226.39.123	10.21.0.1	TCP	114	8000 → 33180 [PSH, ACK] Seq=140 Ack=147 Win=30208 Len=0

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
Ethernet II, Src: 2e:fe:df:ac:93:30 (2e:fe:df:ac:93:30), Dst: ba:a1:72:ac:41:7b (ba:a1:72:ac:41:7b)  
Internet Protocol Version 4, Src: 10.21.0.1, Dst: 159.226.39.123  
Transmission Control Protocol, Src Port: 33180, Dst Port: 8000, Seq: 0, Len: 0

正在捕获 h2-eth0

文件 编辑 视图 窗口 帮助

Apply a display filter

<Ctrl>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	16:87:4d:4c:3a:e4	Broadcast	ARP	42	Who has 159.226.39.123? Tell 159.226.39.43
2	0.000055536	ee:93:c3:db:2a:c0	16:87:4d:4c:3a:e4	ARP	42	159.226.39.123 is at ee:93:c3:db:2a:c0
3	0.000091050	159.226.39.123	159.226.39.123	TCP	74	8000 → 12345 [SYN, ACK] Seq=9 Ack=1 Win=28696 Len=0
4	0.000029770	159.226.39.123	159.226.39.43	TCP	74	8000 → 12345 [SYN, ACK] Seq=9 Ack=1 Win=28696 Len=0
5	0.000059536	159.226.39.43	159.226.39.123	TCP	66	12345 → 8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0
6	0.000462370	159.226.39.43	159.226.39.123	HTTP	212	GET / HTTP/1.1
7	0.000471600	159.226.39.123	159.226.39.43	TCP	66	8000 → 12345 [ACK] Seq=1 Ack=147 Win=30208 Len=0
8	0.000255201	159.226.39.123	159.226.39.43	TCP	83	8000 → 12345 [PSH, ACK] Seq=1 Ack=147 Win=30208 Len=0
9	0.000389109	159.226.39.43	159.226.39.123	TCP	66	12345 → 8000 [ACK] Seq=147 Ack=18 Win=29696 Len=0
10	0.000396940	159.226.39.123	159.226.39.43	TCP	166	8000 → 12345 [PSH, ACK] Seq=18 Ack=147 Win=30208 Len=0
11	0.000412871	159.226.39.43	159.226.39.123	TCP	66	12345 → 8000 [ACK] Seq=147 Ack=118 Win=29696 Len=0
12	0.000562265	159.226.39.123	159.226.39.43	TCP	88	8000 → 12345 [PSH, ACK] Seq=118 Ack=147 Win=30208 Len=0
13	0.000607042	159.226.39.43	159.226.39.123	TCP	66	12345 → 8000 [ACK] Seq=147 Ack=140 Win=29696 Len=0
14	0.000671375	159.226.39.123	159.226.39.43	TCP	114	8000 → 12345 [PSH, ACK] Seq=140 Ack=147 Win=30208 Len=0

Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
Ethernet II, Src: 16:87:4d:4c:3a:e4 (16:87:4d:4c:3a:e4), Dst: ee:93:c3:db:2a:c0 (ee:93:c3:db:2a:c0)  
Internet Protocol Version 4, Src: 159.226.39.43, Dst: 159.226.39.123  
Transmission Control Protocol, Src Port: 12345, Dst Port: 8000, Seq: 0, Len: 0