网络地址转换(NAT)实验

武庆华

wuqinghua@ict.ac.cn

中科院计算所 INSTITUTE OF COMPUTING TECHNOLOGY

中国科学院大学 University of Chinese Academy of Sciences

提纲

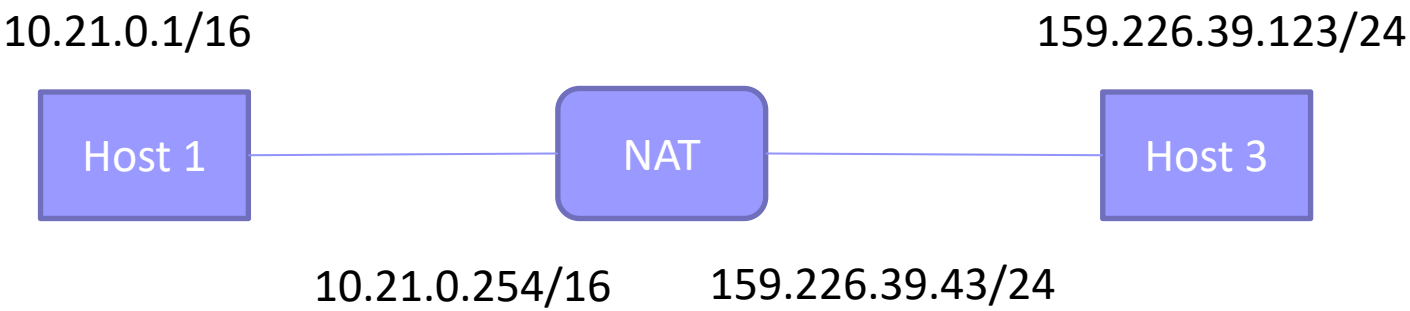
- NAT地址转换
- NAT实现
- NAT实验内容
- 附件文件列表

私网（Private Network） IP地址

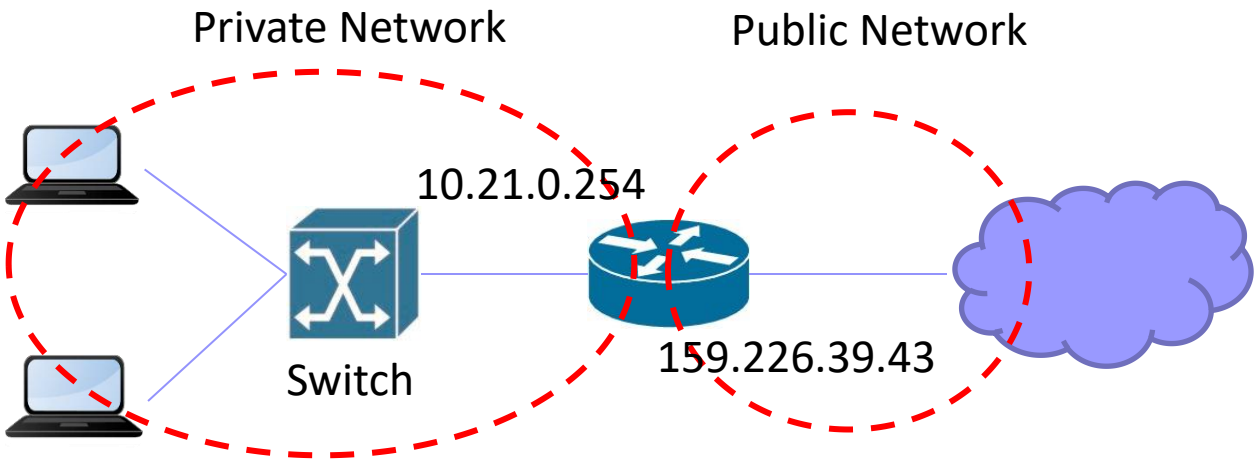
- 私网IP地址空间
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- 为什么要有私网IP地址空间？
  - IP地址数量限制
  - 网络管理需要

网络地址转换

- 如何连接私网IP地址和公网IP地址？
  - 如果使用原有路由机制，则私网地址没有存在的意义
  - 网络地址转换：类似代理的机制
- 网络地址转换：
  - 给定网络拓扑以及节点的网络地址配置，NAT地址转换使得私网节点与公网节点（甚至另一个私网的节点）能够互联并传输数据



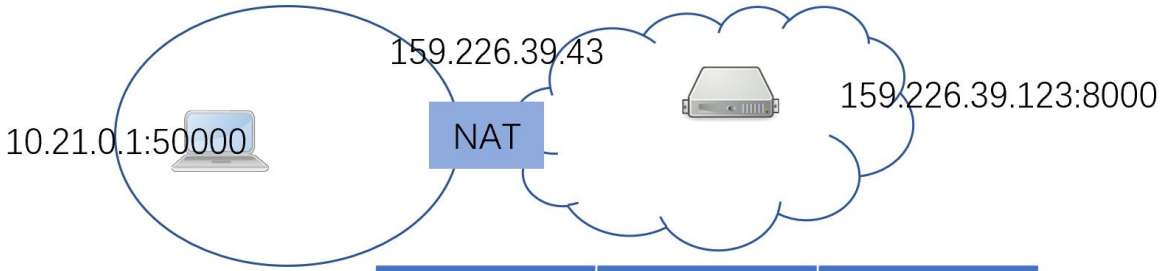
NAT (Network Address Translation)



- NAT设备主要工作：
  1. 维护私网地址/端口 与 公网地址/端口的映射关系
  2. 对数据包内容进行重写（Translation），修改IP地址、端口等字段，使得数据包在相应网络中有意义

NAT工作场景

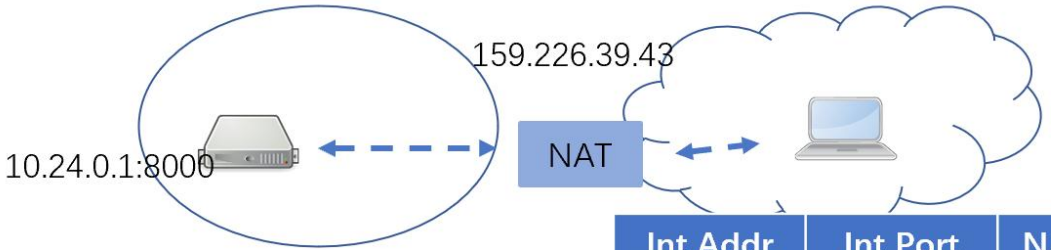
- 私网主机连接到公网服务器



Int Addr	Int Port	NAT Port
10.21.0.1	50000	12345

Source NAT (SNAT)

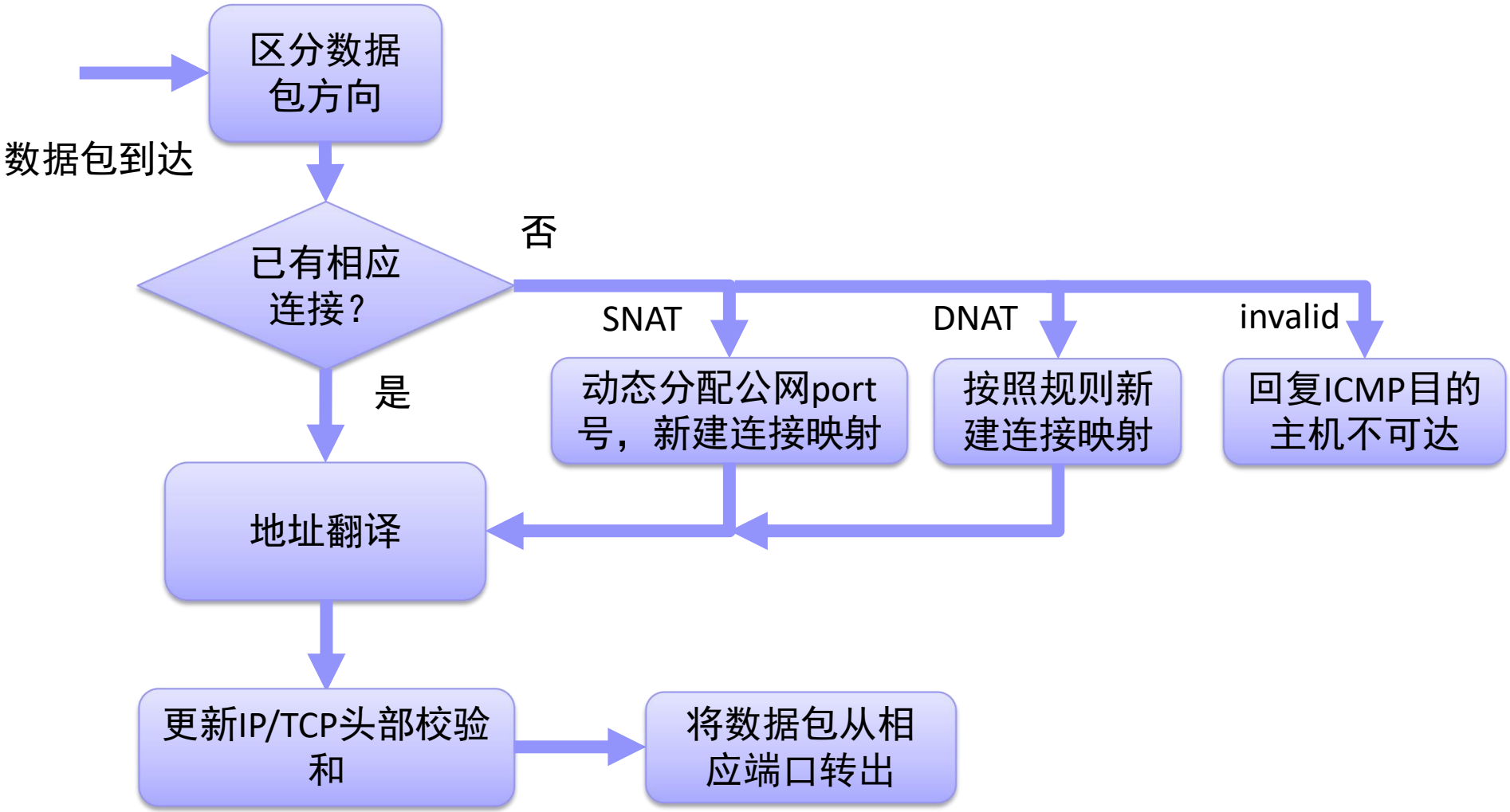
- 私网主机作为服务器



Int Addr	Int Port	NAT Port
10.24.0.1	8000	8000

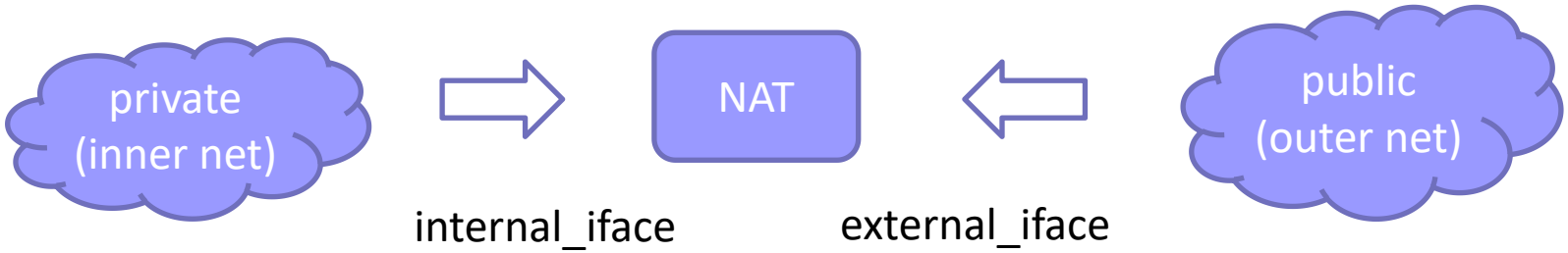
Destination NAT (DNAT)

NAT工作机制


区分数据包方向
数据包到达
已有相应连接？
否
是
SNAT
DNAT
invalid
动态分配公网port号，新建连接映射
按照规则新建连接映射
回复ICMP目的主机不可达
地址翻译
更新IP/TCP头部校验和
将数据包从相应端口转出

区分数据包方向

- 只处理两个方向的数据包



1. 当源地址为内部地址，且目的地址为外部地址时，方向为DIR_OUT
2. 当源地址为外部地址，且目的地址为external_iface地址时，方向为DIR_IN

- 如何判断是内部地址还是外部地址？
  - 查询路由表，根据目的地址相应转发条目对应的iface判断地址类别

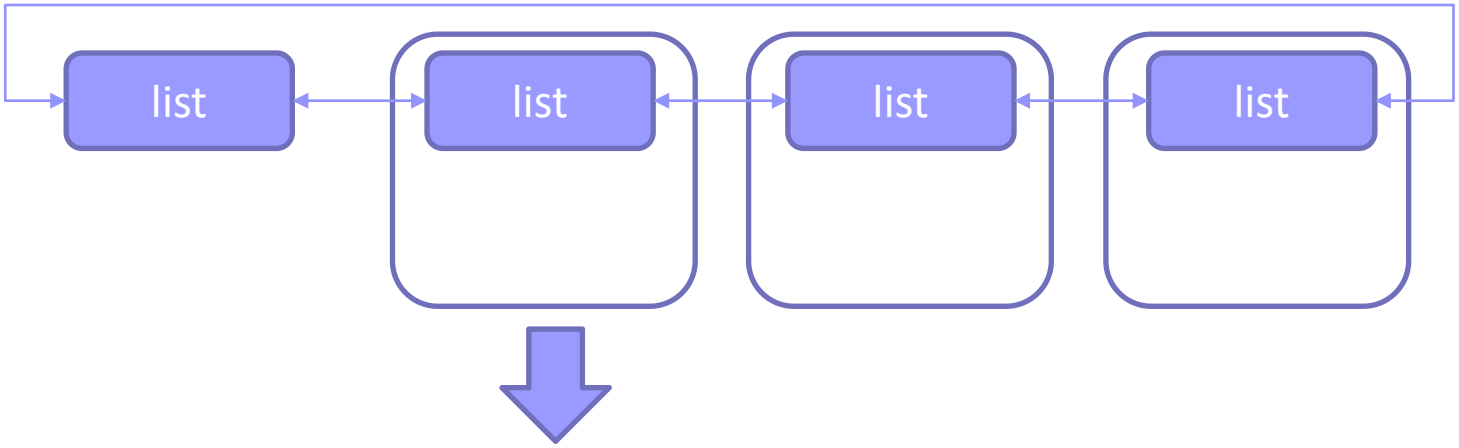
合法数据包

- 该数据包在NAT中有对应连接映射（existing)
- 该数据包的方向为DIR_OUT，为该TCP连接的第一个数据包（请求连接数据包），NAT中没有对应连接映射（SNAT)
- 该数据包的方向为DIR_IN，为该TCP连接的第一个数据包，NAT中没有对应连接映射，但有对应处理规则 (DNAT)

NAT地址翻译

- Existing
  - 查找映射关系，进行(internal_ip, internal_port) <-> (external_ip, external_port)之间的转换
- SNAT
  - saddr = external_iface->ip; sport = assign_external_port();
    - 不能使用端口0建立连接
  - 建立连接映射关系
    - (internal_ip, internal_port) <-> (external_ip, external_port)
- DNAT
  - daddr = rule->daddr; dport = rule->dport;
  - 建立连接映射关系
    - (internal_ip, internal_port) <-> (external_ip, external_port)
- 更新IP/TCP数据包头部字段(包括校验和)

连接（映射关系）的维护



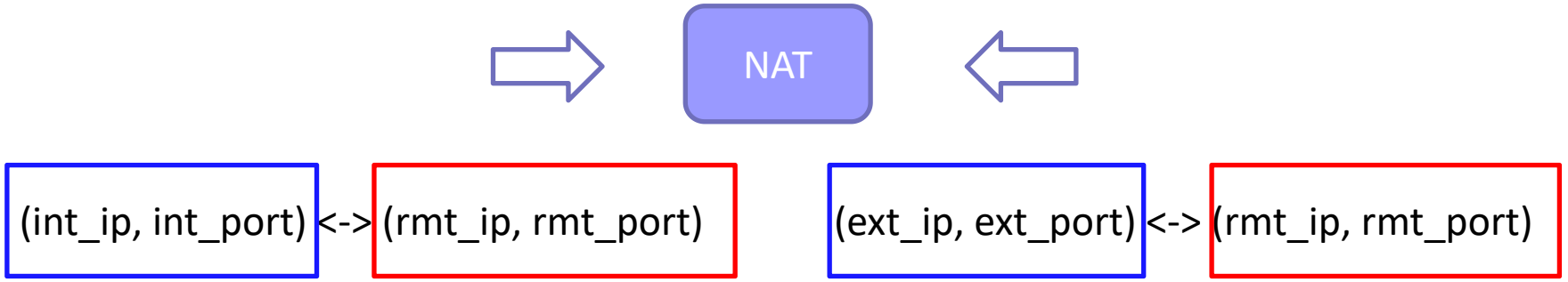

```
struct nat_mapping {
    struct list_head list;

    u32 internal_ip;
    u16 internal_port;
    u32 external_ip;
    u16 external_port;
    ...;
};
```

- 一个NAT设备需要同时维护数万条映射关系
- 链表查找方式效率非常低
- 可以考虑使用Hash方式

使用Hash查找映射关系

- Hash表存储映射关系 key??? -> nat_mapping



Observation: (rmt_ip, rmt_port) 是地址翻译中的不变量

Problem: 可能有多个主机同时请求该服务，这些连接有相同的rmt_ip+rmt_port

Solution: 可以先用(rmt_ip, rmt_port) 定位到一组映射结构（链表），再根据数据包方向，决定用(rmt_ip, rmt_port) + (int_ip, int_port) 还是(rmt_ip, rmt_port) + (ext_ip, ext_port) 来确定唯一的映射结构

NAT老化（Timeout）操作

- 端口号是NAT设备中的宝贵资源
  - 实验中的SNAT设备，一个公网地址最多支持65535个并发连接
  - 对于已经结束的连接，可以收回已分配的端口号，释放连接映射资源

- 对认为已经结束的连接进行老化操作
  - 双方都已发送FIN且回复相应ACK的连接，一方发送RST包的连接，可以直接回收
  - 双方已经超过60秒未传输数据的连接，认为其已经传输结束，可以回收

NAT数据结构

```
struct nat_table {
    struct list_head nat_mapping_entries[256]; // 映射Hash表
    iface_info_t *internal_iface;          // 私网端口
    iface_info_t *external_iface;          // 公网端口
    u8 assigned_ports[65536];              // port号池
    struct list_head rules;                // DNAT规则
    pthread_mutex_t lock;                  // 互斥操作锁
    pthread_t thread;                      // 老化线程ID
};
static struct nat_table nat;               // NAT主数据结构
```

TCP连接映射数据结构

- 为了快速访问对应的映射结构，用Hash表来存储映射关系 (ip, port) -> nat_mapping

```
struct nat_mapping {
    struct list_head list;

    u32 internal_ip;
    u16 internal_port;
    u32 remote_ip;
    u16 remote_port;
    u32 external_ip;
    u16 external_port;

    time_t update_time;
    struct nat_conn_state state;
};
```

DIR_OUT

DIR_IN

```
struct nat_conn_state {
    u8 internal_fin;
    u8 external_fin;

    u32 internal_seq_end;
    u32 external_seq_end;

    u32 internal_ack;
    u32 external_ack;
};
```

NAT实现

- 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

NAT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

结果示意

获取网页

```
root@alvin-ubuntu: ~/networking
root@alvin-ubuntu:~/networking # wget http://159.226.39.12
--2019-11-12 17:16:56--  http://159.226.39.123:8000/
Connecting to 159.226.39.123:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 213 [text/html]
Saving to: 'index.html'

index.html            100%[======================>]       2

2019-11-12 17:16:56 (25.4 MB/s) - 'index.html' saved [213/

root@alvin-ubuntu:~/networking # cat index.html

<!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>
root@alvin-ubuntu:~/networking #
```

从h3角度看

抓包结果

h1-eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools

tcp

No.	Time	Source	Destination	Protocol
1	0.000000000	10.21.0.1	159.226.39.123	TCP
4	0.001093008	159.226.39.123	10.21.0.1	TCP
5	0.001140565	10.21.0.1	159.226.39.123	TCP
6	0.001252383	10.21.0.1	159.226.39.123	HTTP
7	0.001331878	159.226.39.123	10.21.0.1	TCP
8	0.007256671	159.226.39.123	10.21.0.1	TCP
9	0.007276519	10.21.0.1	159.226.39.123	TCP
10	0.007374477	159.226.39.123	10.21.0.1	TCP
11	0.007385092	10.21.0.1	159.226.39.123	TCP
12	0.007451011	159.226.39.123	10.21.0.1	HTTP
13	0.007457471	10.21.0.1	159.226.39.123	TCP
14	0.007608256	159.226.39.123	10.21.0.1	TCP
15	0.008304576	10.21.0.1	159.226.39.123	TCP

h3-eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools

tcp

No.	Time	Source	Destination	Protocol
3	0.000048970	159.226.39.43	159.226.39.123	TCP
4	0.000072544	159.226.39.123	159.226.39.43	TCP
5	0.000898660	159.226.39.43	159.226.39.123	TCP
6	0.000954769	159.226.39.43	159.226.39.123	HTTP
7	0.000968682	159.226.39.123	159.226.39.43	TCP
8	0.006850955	159.226.39.123	159.226.39.43	TCP
9	0.006988695	159.226.39.43	159.226.39.123	TCP
10	0.007006659	159.226.39.123	159.226.39.43	TCP
11	0.007081104	159.226.39.43	159.226.39.123	TCP
12	0.007093141	159.226.39.123	159.226.39.43	HTTP
13	0.007144689	159.226.39.43	159.226.39.123	TCP
14	0.007176728	159.226.39.123	159.226.39.43	TCP
15	0.008061976	159.226.39.43	159.226.39.123	TCP

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

实验内容三

- 手动构造一个包含两个nat的拓扑
  - h1 <-> n1 <-> n2 <-> 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

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。

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）

附件文件列表

- exp[1-2].conf # NAT配置文件
- libipstack(32).a #如果是32位机器，需要将文件名中的32去掉；也可将``路由器转发实验''中自己编译的版本拷贝过来
- http_server.py # 简单HTTP Server实现
- include
- ip.c
- Makefile
- main.c
- nat.c # NAT相关函数，待实现
- nat_topo.py
- scripts