

## Lab6

171830635 俞星凯

实验目的	<ol style="list-style-type: none"><li>1. 设计和实现一个简单的虚拟专用网络的机制。</li><li>2. 与已有的标准实现（如 PPTP）进行比较，进一步理解 VPN 的工作原理和内部实现细节。</li></ol>
数据结构说明	<ol style="list-style-type: none"><li>1. 路由配置相关数据结构<pre>struct RouteItem {     uint32_t dstNet;     uint32_t netmask;     uint32_t ifIndex;     uint32_t gateway; };  struct ArpTableItem{     uint32_t ipAddr;     uint8_t macAddr[6]; };  struct DeviceItem{     uint32_t ifIndex;     //uint8_t  ifName[10];     uint32_t ipAddr;     uint8_t macAddr[6]; };</pre>上述数据结构分别定义了静态路由表，ARP 缓存和设备信息。</li><li>2. 数据包相关数据结构<pre>struct IcmpPack{     uint8_t  type;     uint8_t  code;     uint16_t checksum;     uint16_t id;     uint16_t sequence;     uint8_t  data[56]; };  struct IPPack{     uint8_t header_length:4,            version:4;     uint8_t dscp:6,            ecn:2;</pre></li></ol>

```
uint16_t total_length;
uint16_t id;
uint16_t fragment_off:13,
        flags:3;
uint8_t ttl;
uint8_t protocol;
uint16_t checksum;
uint32_t srcIP;
uint32_t dstIP;
uint8_t payload[500];
```

```
};
```

```
struct EthPack
{
    uint8_t dstMacAddr[6];
    uint8_t srcMacAddr[6];
    uint16_t ethType;
    struct IPPack ipPack;
};
```

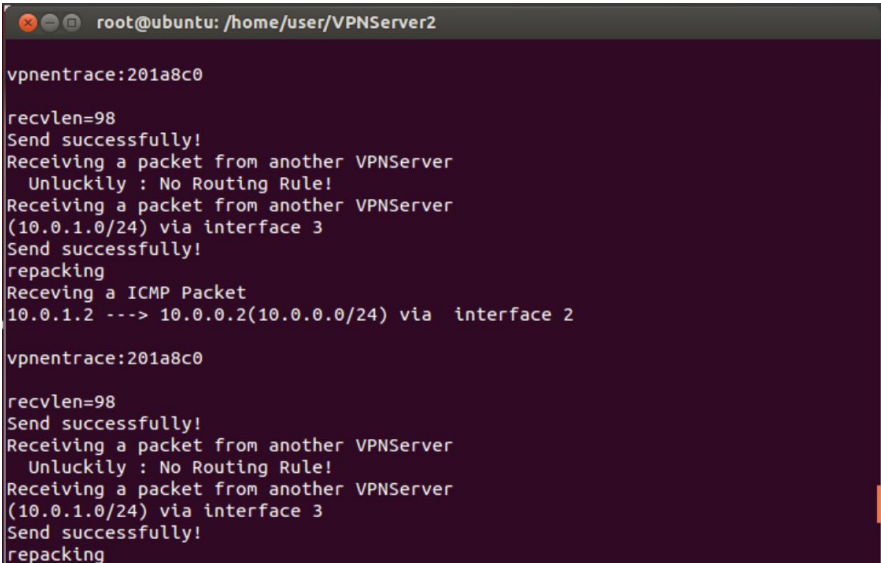
```
struct EthArpPack{
    //ethernet header
    uint8_t dstMacAddr[6];
    uint8_t srcMacAddr[6];
    uint16_t ethType;

    uint16_t hrdType;
    uint16_t proType;
    uint8_t hrdLen;
    uint8_t proLen;
    uint16_t opcode;
    uint8_t srcHrdAddr[6];
    uint32_t srcProAddr;
    uint8_t dstHrdAddr[6];
    uint32_t dstProAddr;
};
```

上述数据结构分别定义了 ICMP，IP，以太网帧和 ARP。

配置文件说明	<div>192.168.0.2 //连接公有网的 IP</div> <div>eth0 //接入口的端口号</div> <div>10.0.1.0/24 //目的 IP</div> <div>192.168.1.2 //对方 VPN</div> <div>eth1 //端口号</div> <div>10.0.0.0/24 //目的 IP</div> <div>10.0.0.2 //己方主机</div> <div>eth0 //端口号</div>
程序设计的思路 以及运行流程	<div>1. 运行流程</div> <div><pre>graph TD     A["一个标准的 IP 包， 包的内容未知，包头 地址为 VPN 地址"] -- "发送到 VPN 接入口上" --&gt; B["包被解析，被发现可以路由， 将整个包作为内容添加必要头部信息， 添加到一个标准的可以在网络上 传输的 IP 包中"]     B --&gt; C["作为一个标准的 IP 包在网络上进行 传输，过程同一般的 IP 包没有区别。 目的地址为对应 VPN 目的地址所在 网络的接入口"]     C --&gt; D["VPN 接入口收到该包，将其解包， 并查看内容头部信息，发现确是本接入 点所在 VPN 网络上的主机能接收的。 将其重新解包，发送到目的主机。"]     D --&gt; E["包被接收，传输完毕"]     F["读取配置文件，设置基于 VPN 的 基本路由，ARP 缓存信息"] --&gt; G["对机器各个接口进行监听， 如有包进入，将对包的类型进行检测"]     G -- "来自内部" --&gt; H["查询 VPN 虚拟路由， 对收到的包进行重新打包，并发送到 互联网上"]     G -- "来自外部" --&gt; I["查询 VPN 虚拟路由以及其他信息， 对该报进行解包，重新安插以太网头部 信息，并发送到相应节点上"]     H --&gt; J(( ))     I --&gt; J     J --&gt; G</pre></div>

	<p>2. VPN 的设计思路</p> <p>先手动设置 VPN 服务器的 IP 地址,然后通过 <code>getIfIndex()</code>,<code>getIfIP()</code>,<code>getIfMac()</code> 三个函数完成对于设备信息的加载,需要注意的是本次没有使用字符串来描述信息,而是直接使用 32 位或 8 位无符号整数,便于运算。接下来创建套接字并持续接收数据包,如果是从内向外发送,则调用 <code>repack()</code>;如果是从外向内发送,则调用 <code>unpack()</code>。在 <code>repack()</code>中,为整个数据包添加头部信息,源 IP 为自己的 IP,目的 IP 为对方 VPN 的 IP,源 MAC 为自己的 MAC,目的 MAC 为 Network 的 MAC,这样就可以在 Network 中传输了;在 <code>unpack()</code>中,去除原先的头部信息,包括源、目的 IP 和 MAC,数据包就变回了原先的数据包,再根据路由表将它传输给对应接收方。</p>
运行结果截图	<p>1. PC1</p>  <pre>user@ubuntu: ~ 64 bytes from 10.0.1.2: icmp_req=16 ttl=64 time=2.52 ms 64 bytes from 10.0.1.2: icmp_req=17 ttl=64 time=1.42 ms 64 bytes from 10.0.1.2: icmp_req=18 ttl=64 time=4.50 ms 64 bytes from 10.0.1.2: icmp_req=19 ttl=64 time=1.25 ms 64 bytes from 10.0.1.2: icmp_req=20 ttl=64 time=1.14 ms 64 bytes from 10.0.1.2: icmp_req=21 ttl=64 time=2.26 ms 64 bytes from 10.0.1.2: icmp_req=22 ttl=64 time=3.66 ms 64 bytes from 10.0.1.2: icmp_req=23 ttl=64 time=3.50 ms 64 bytes from 10.0.1.2: icmp_req=24 ttl=64 time=1.38 ms 64 bytes from 10.0.1.2: icmp_req=25 ttl=64 time=4.66 ms 64 bytes from 10.0.1.2: icmp_req=26 ttl=64 time=1.16 ms 64 bytes from 10.0.1.2: icmp_req=27 ttl=64 time=1.41 ms 64 bytes from 10.0.1.2: icmp_req=28 ttl=64 time=1.22 ms 64 bytes from 10.0.1.2: icmp_req=29 ttl=64 time=1.30 ms 64 bytes from 10.0.1.2: icmp_req=30 ttl=64 time=1.27 ms 64 bytes from 10.0.1.2: icmp_req=31 ttl=64 time=1.19 ms 64 bytes from 10.0.1.2: icmp_req=32 ttl=64 time=1.40 ms 64 bytes from 10.0.1.2: icmp_req=33 ttl=64 time=2.07 ms 64 bytes from 10.0.1.2: icmp_req=34 ttl=64 time=1.14 ms ^C --- 10.0.1.2 ping statistics --- 39 packets transmitted, 34 received, 12% packet loss, time 38098ms rtt min/avg/max/mdev = 1.071/1.932/6.762/1.264 ms user@ubuntu:~\$</pre> <p>2. VPNServer1</p>  <pre>user@ubuntu: ~/VPNServer1 Receiving a packet from another VPNServer (10.0.0.0/24) via interface 2 Send successfully! repacking  vpnentrace110:200a8c0 Receiving a ICMP Packet 10.0.0.2 ---&gt; 10.0.1.2(10.0.1.0/24) via interface 3  vpnentrace:200a8c0  recvlan=98 Send successfully! Receiving a packet from another VPNServer Unluckily : No Routing Rule! Receiving a packet from another VPNServer (10.0.0.0/24) via interface 2 Send successfully! repacking  vpnentrace110:200a8c0 Receiving a ICMP Packet 10.0.0.2 ---&gt; 10.0.1.2(10.0.1.0/24) via interface 3</pre>

	<div>3. VPNServer2</div> <div>A terminal window titled 'root@ubuntu: /home/user/VPNServer2' displays the following log output: vpnentrace:201a8c0 recvlen=98 Send successfully! Receiving a packet from another VPNServer Unluckily : No Routing Rule! Receiving a packet from another VPNServer (10.0.1.0/24) via interface 3 Send successfully! repacking Receiving a ICMP Packet 10.0.1.2 ---&gt; 10.0.0.2(10.0.0.0/24) via interface 2 vpnentrace:201a8c0 recvlen=98 Send successfully! Receiving a packet from another VPNServer Unluckily : No Routing Rule! Receiving a packet from another VPNServer (10.0.1.0/24) via interface 3 Send successfully! repacking</div>
相关资料	<div>1. 实验教材</div> <div>2. <a href="#">百度百科 VPN 词条</a></div> <div>3. <a href="#">VPN 原理及报文格式</a></div>