

实验报告评分表

评分项目	分值	评分标准	得分
实验原理	10	10-8：原理理解准确，说明清晰完整 7-5：原理理解基本准确，说明较为清楚 4-0：说明过于简单	
VPN 系统设计	25	25-19：系统架构和模块划分合理，详细设计说明详实准确 18-11：系统架构和模块划分基本合理，详细设计说明较为详实准确 10-0：系统架构和模块划分不恰当，详细设计说明过于简单	
VPN 实现细节	25	25-19：文字表达清晰流畅，实现方法技术优良，与设计实现及代码一致 18-11：文字表达较清晰流畅，实现方法一般，与设计实现及代码有偏差 10-0：文字表达混乱，实现方法过于简单	
测试结果与分析	20	20-15：功能测试覆盖完备，测试结果理想，分析说明合理可信 14-9：功能测试覆盖基本完备，测试结果基本达标，分析说明较少 8-0：功能测试覆盖不够，测试未达到任务要求，缺乏分析说明	
体会与建议	10	10-8：心得体会真情实感，意见中肯，建议合理可行，体现了个人的思考 7-5：心得体会较为真实，意见建议较为具体 4-0：过于简单敷衍	
格式规范	10	图、表的说明，行间距、缩进、目录等不规范相应扣分	
总 分			

---

# 目 录

<b>1</b>	<b>实验原理.....</b>	<b>1</b>
<b>2</b>	<b>VPN 系统设计.....</b>	<b>4</b>
2.1	概要设计.....	4
2.2	详细设计.....	5
<b>3</b>	<b>VPN 实现细节.....</b>	<b>10</b>
3.1	问题 1 .....	10
3.2	问题 2 .....	11
3.3	问题 3 .....	12
<b>4</b>	<b>测试结果与分析.....</b>	<b>13</b>
4.1	认证 VPN 服务器.....	13
4.2	认证 VPN 客户端.....	14
4.3	加密隧道通信.....	15
4.4	支持多客户端.....	16
4.5	易用性和稳定性.....	18
<b>5</b>	<b>体会与建议.....</b>	<b>19</b>
5.1	心得体会.....	19
5.2	意见建议.....	19

# 1 实验原理

## (1) 网络拓扑

本次实验的网络结构如图 1 所示，在两个外网主机 HostU 和 HostU2 上运行 VPNclient，其 IP 地址为 10.0.2.7 和 10.0.2.6；VPN 网关上运行 VPNserver，其在外网的 IP 为 10.0.2.8，在内网的 IP 为 192.168.60.1，在内网中有一个主机为 HostV，其 IP 为 192.168.60.101。

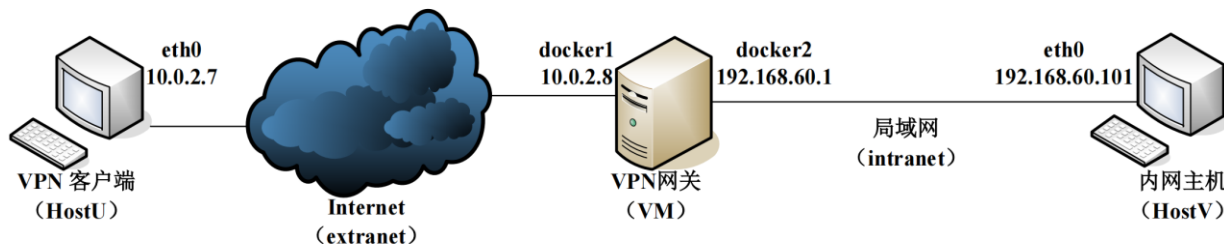


图 1 实验网络结构拓扑图

在实际环境中，VPN 客户端（HostU）和 VPN 服务器的外网口通过 Internet 连接。简单起见，我们在本实验中将这两台机器直接连接到同一 docker 网络“extranet”中模拟 Internet。第三台机器 HostV 是内部局域网的计算机，我们使用 docker 网络“intranet”将 HostV 与 VPN 服务器的内网口连接，模拟内部局域网。HostV 不能直接从 Internet 访问，即 HostU 不能直接访问 HostV。Internet 主机 HostU 上的用户希望通过与 VPN 服务器建立 TLS/SSL VPN 隧道，实现与内部局域网的主机 HostV 通信。

## (2) 通信机制

实验目的是实现主机 HostU 和主机 HostV 之间的加密 VPN 通信，其中 HostU 是外网 extranet 的主机，通过 VPN 网关 VM 连接到内网 intranet。内网的地址段为 192.168.60.0/24。本次实验采用的 vpn 结构是 TLS/SSL VPN，通信过程示例如图 2 所示：（以客户端发往服务器为例）

How packets flow from client to server when running "telnet 10.0.20.100" using a VPN

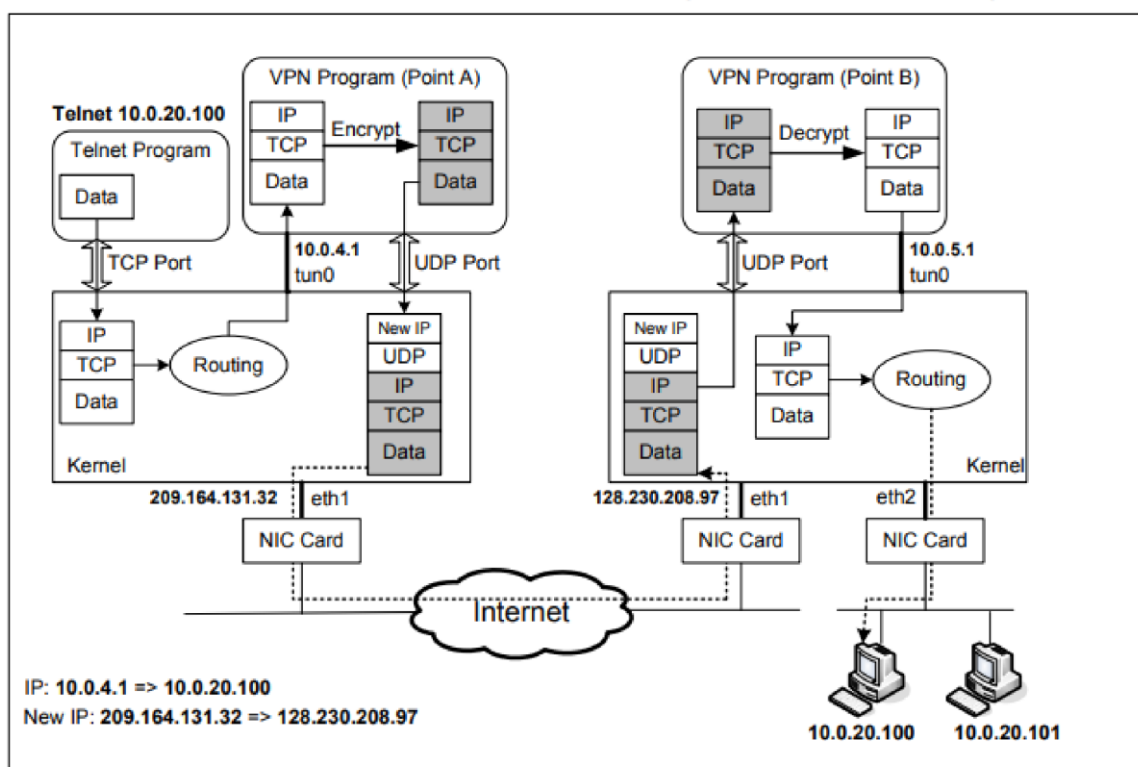


图 2 通信过程

首先，在客户主机上创建虚拟网口 tun0，将发往目的主机的 IP 的数据包路由到 tun0，VPNclient 程序在 tun0 进行监听，将收到的数据包进行加密，隧道封装，再通过主机的物理网口发送到目的主机的 vpn 服务器（通过 SSL 通信实现）。目的主机服务器也在自己主机上创建一个虚拟网口 tun0，将 tun0 收到的数据路由到子网主机网段。在隧道的另一边收到加密数据包后，进行提取解密，得到客户机发出的原始数据包并发送到 tun0，此时改数据包的 dst IP 就是指向子网中的目的主机。这样就能成功发送到目的主机了。

### (3) 加密原理

TLS（传输层安全）协议：

- **TLS Initialization:** 在服务器和客户端中，TLS 库和算法被初始化。使用 `SSL_library_init()`、`SSL_load_error_strings()` 和 `SSL_load_error_strings()` 来加载必要的 SSL 库和算法。
- **TLS Context Setup:** 服务器端使用 `SSL_CTX_new()` 创建一个新的 SSL 上下文，客户端使用 `SSL_CTX_new()` 配置 TLS 方法。服务器加载其证书和私钥文件来验证其身份。
- **TLS Handshake:** 通过 `SSL_accept()` 在服务器端和 `SSL_connect()` 在客户端进行 TLS 握手，确保双方之间建立了安全的通信通道。握手成功后，双方可以进行加密通信。

- **Data Encryption:** 握手成功后，所有在 TLS 通道上传输的数据都是加密的。服务器和客户端使用 `SSL_read()` 和 `SSL_write()` 进行数据的加密传输。

#### (4) 认证机制

对于客户机和服务器的单向认证机制，采用的是 TLS 认证协议，具体步骤如图 3:

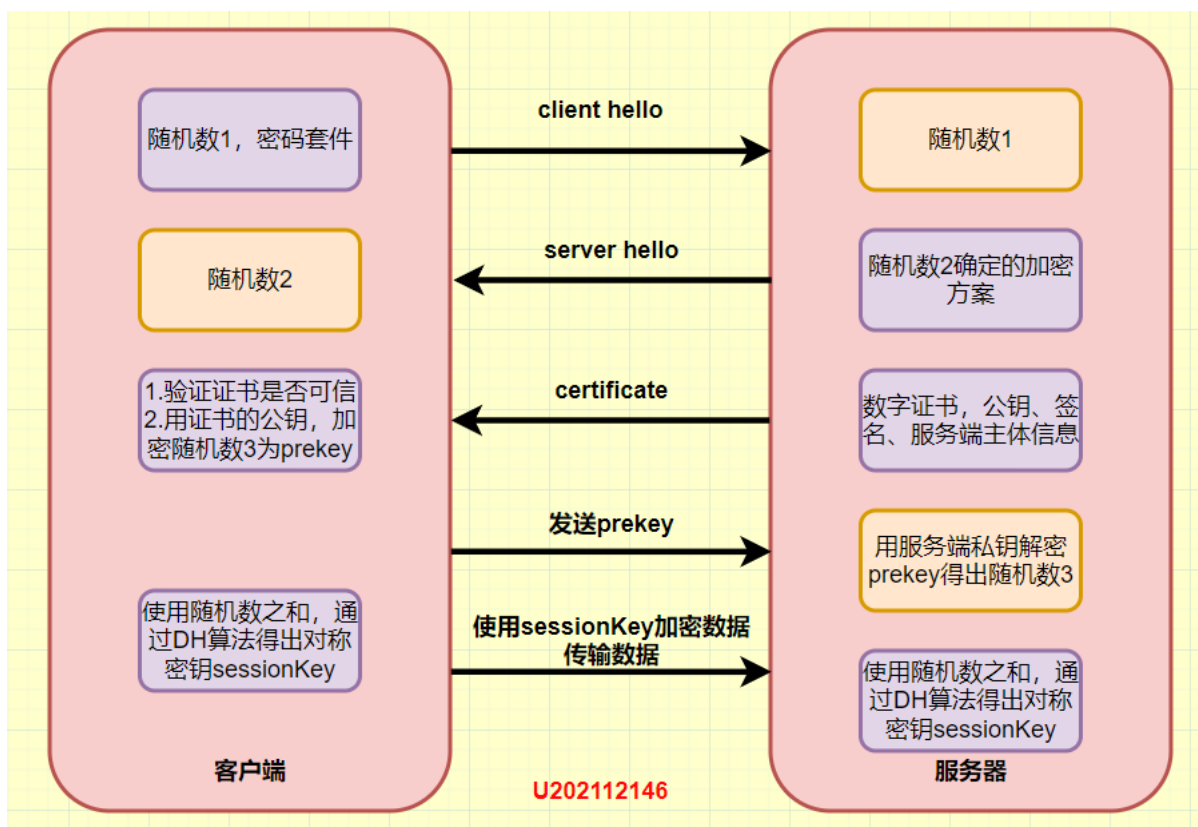


图 3 TLS 协议工作流程

其中，签名算法和对称加密算法采用的是 AES256-GCM，摘要算法采用的是 SHA384。对于服务端对客户端的认证，采用的是用户名和口令的验证方法，客户端将用户名、密码和最后一个 IP 字节发送给服务器。服务器使用 `getspnam()` 从阴影密码文件中获取加密的密码条目。然后使用 `crypt()` 函数对接收到的密码进行加密，并与阴影密码文件中的加密密码进行比较。如果匹配，认证通过；否则，认证失败。

## 2 VPN 系统设计

### 2.1 概要设计

本次 VPN 实验一共包括两个部分，即 client 和 server。其中 vpn 服务器部署在内部网关上，vpn 客户端部署在用户主机上。以下分别说明两个部分的具体模块划分和工作机制。

#### 1. VPN 服务器

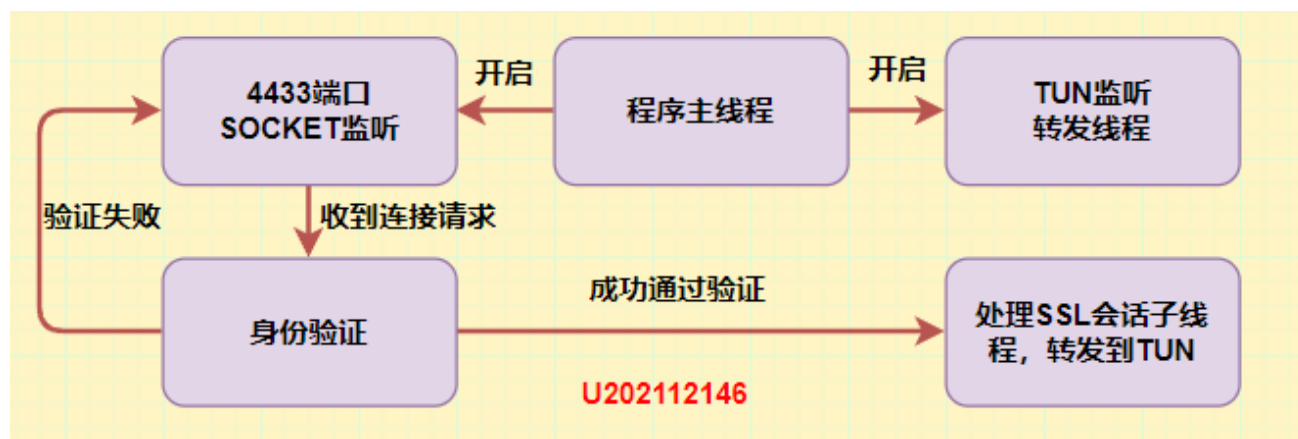


图 4 VPN 服务器工作流程

##### (1) 程序主线程

主线程的职责是启动并初始化网络设备，这涉及到生成一个 TUN 虚拟网络接口，并进行必要的系统配置。随后，它在服务器的预设端口号 4433 上进行网络监听，以识别并响应来自客户端的连接请求。一旦客户端发起连接，主线程便与客户端执行 SSL 协议的握手过程，通过这一过程协商安全参数并确立一个安全的 SSL 连接。在 SSL 连接建立之后，主线程获取客户端提交的用户名和密码，执行身份验证的步骤。如果用户提供的凭证验证成功，主线程将为这个特定的 SSL 连接创建一个新的子线程。该子线程的目的是处理并转发所有通过 SSL 接口接收到的数据。

##### (2) TUN 监听线程

在运行 VPN 系统过程中，网关的 TUN 虚拟网络接口需要被持续监控。为了实现 VPN 客户端的流量传输，必须在内网主机上配置路由规则，确保所有发往 VPN 客户端 IP 地址的网络数据包都能被正确地导向 VPN 网关，也就是虚拟机（VM）。当 VM 接收到这些数据包后，它们会被转发至 TUN0 虚拟网络接口。

此时，一个特定的子线程在此过程中扮演关键角色。它的任务是专门监控 TUN0 虚拟网络接口上的活动。一旦该子线程监听到有数据包到达，它会根据在建立 SSL 连接时创建的映射关系——即 IP 地址与 SSL 隧道之间的对应关系——来确定数据包的传输路径。随后，子线程

利用相应的 SSL 安全隧道，将这些数据包安全地发送至目的地。这个过程确保了数据在传输过程中的加密和完整性，从而保障了 VPN 通信的安全性。

### (3) SSL 监听线程

对每个身份验证成功的客户端，主线程会保留创建的 SSL 会话，并建立子线程处理这些会话，将隧道收到的内容，转发到虚拟网口 tun0。

## 2. VPN 客户端

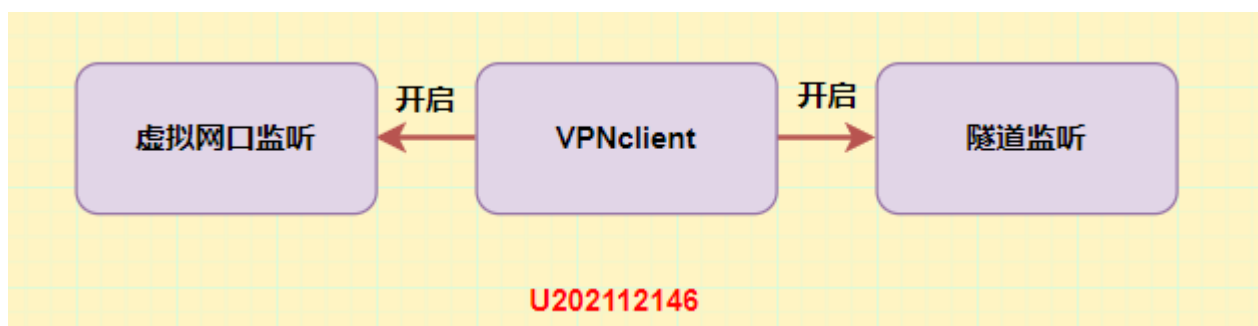


图 5 VPN 客户端的工作模块

客户端在 VPN 系统中的职责相对简单，因为它仅需管理单个 SSL 隧道。客户端的程序执行流程从系统初始化和配置开始，包括与服务器建立连接、创建虚拟网络接口以及启动两个子线程。在与服务器建立联系的过程中，客户端首先与服务器完成 SSL 连接，随后提交用户输入的用户名和密码供服务器进行身份验证。一旦身份验证成功，客户端将 SSL 隧道和虚拟网络接口的文件描述符加入到 select 函数的监控集合中。当任一接口接收到数据包时，客户端程序便将其转发至另一接口，实现数据的传输。这个过程确保了数据在客户端和服务器之间的安全流动。

## 2.2 详细设计

### 1. 服务器

#### (1) 监听 tun 模块

```
void *listen_tun(void *tunfd) {  
    int fd = *(int *)tunfd;  
    char buff[2000];  
    int ret;  
    SSL *ssl;  
    uint32_t ip = 0;
```

```

while (1) {
    int len = read(fd, buff, 2000);
    if (len > 19 && buff[0] == 0x45) {
        printf("Receive TUN: len = %d | ip.des = 192.168.53.%d\n", len,
            (int)buff[19]);
        ip = *(uint32_t *)&buff[16];
        ssl = Get_ssl(ip);
        if (ssl != NULL)
            ret = SSL_write(ssl, buff, len);
        else {
            printf("SSL address is NULL\n");
        }
        if (ret <= 0) {
            printf("SSL_write error\n");
            SSL_get_error(ssl, ret);
        }
        memset(buff, 0, 2000);
    }
}
}

```

这个函数的作用就是监听 tun 网口并转发到对应隧道, 具体来说, 在接收到一个数据包后, 读取数据包的目的 IP 字段, 使用 Get\_ssl() 函数, 在虚拟 IP-SSL 文件描述符映射表中, 查找对应的 SSL 文件描述符, 再写到隧道中去。映射表结构如下:

```

typedef struct IP_MAP {
    uint32_t ip;
    SSL *sslfd;
    pthread_t transfer;
} IP_MAP;

```

其中, ip 为为客户端生成的虚拟 ip, sslfd 为对应的 SSL 隧道的文件描述符, transfer 为对应 SSL 隧道转发的线程句柄。

## (2) Get\_ssl() 函数实现

```

SSL *Get_ssl(uint32_t ip) {
    int i;
    for (i = 0; i < 254; i++) {
        if (ip == ipmap[i].ip) {
            return ipmap[i].sslfd;
        }
    }
}

```



```

    }
}
return 0;
}

```

为了区分不同的客户端，服务器会为每个向监听端口发送连接请求的客户端分配一个虚拟 IP，用作该客户端的唯一标识。生成虚拟 IP 的函数如下：

```

int get_virtual_ip(SSL *ssl) //建表维护
{
    uint32_t vip = inet_addr("192.168.53.2");
    uint32_t step = 0x1 << 24;
    int i = 0;
    int empty;
    for (i = 0; i < 254; i++) {
        if (ipmap[i].ip == 0) {
            empty = i;
            break;
        }
    }
    ipmap[empty].ip = vip + empty * step;
    ipmap[empty].sslfd = ssl;
    printf("assign a virtual ip:0x%08lu\tssl address:0x%p\n", ipmap[empty].ip,
           ipmap[empty].sslfd);
    return empty;
}

```

基本思想是在 192.168.53.0/24 地址段中，顺序分配未被使用的地址，断开 SSL 连接后，将对应占用的 IPMAP 清空。

对于 TLS 握手，认证等操作，使用 openssl 的函数进行操作，其自己会完成需要的过程，我们只需要在设置好证书，私钥等后，使用 SSL\_accept 函数等待连接，连接成功后就可以很方便的使用 SSL\_read 和 SSL\_write 进行操作了。

### (3) 初始化 SSL\_server 的函数

```

SSL *setupTLSServer() {
    SSL_METHOD *meth;
    SSL_CTX *ctx;
    SSL *ssl;
    int err;

```

```

SSL_library_init();
SSL_load_error_strings();
SSL_load_ssl_algorithms();
meth = SSLv23_server_method();
ctx = SSL_CTX_new(meth);
SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE, NULL);
SSL_CTX_load_verify_locations(ctx, CACERT, NULL); //加载 CA 证书
if (SSL_CTX_use_certificate_file(ctx, CERTF, SSL_FILETYPE_PEM) <= 0) {
    ERR_print_errors_fp(stderr);
    exit(3);
}
if (SSL_CTX_use_PrivateKey_file(ctx, KEYF, SSL_FILETYPE_PEM) <= 0) {
    ERR_print_errors_fp(stderr);
    exit(4);
}
if (!SSL_CTX_check_private_key(ctx)) {
    fprintf(stderr, "Private key does not match the certificate public key\n");
    exit(5);
}
ssl = SSL_new(ctx);
return ssl;
}

```

#### (4) 服务器对客户端的口令验证代码

```

int login(char *user, char *passwd) {
    struct spwd *pw;
    char *epasswd;
    pw = getspnam(user);
    if (pw == NULL) {
        return -1;
    }
    printf("Login name: %s\n", pw->sp_namp);
    printf("Passwd: %s\n", pw->sp_pwdp);

    epasswd = crypt(passwd, pw->sp_pwdp);
    if (strcmp(epasswd, pw->sp_pwdp)) {
        return 0;
    }
}

```

```
}  
    return 1;  
}
```

其中 `getspnam()` 函数，用于在 `/etc/shadow` 文件中取出对应用户名的口令的哈希值，`crypt()` 函数计算得到用户输入的口令的哈希值，再与 `shadow` 文件中的哈希值进行比较。

## 2. 客户端

客户端的功能比较简单，对于 `tun` 和 `SSL` 隧道报文的转发，和服务器类似，但是不需要查表，因为两个都是单一的信道，这里就不再赘述。对于客户端的虚拟 IP，是在建立 `SSL` 连接后，再由服务器发送回来的，接收虚拟 IP 以及对虚拟网口 `tun` 进行设置的代码如下：

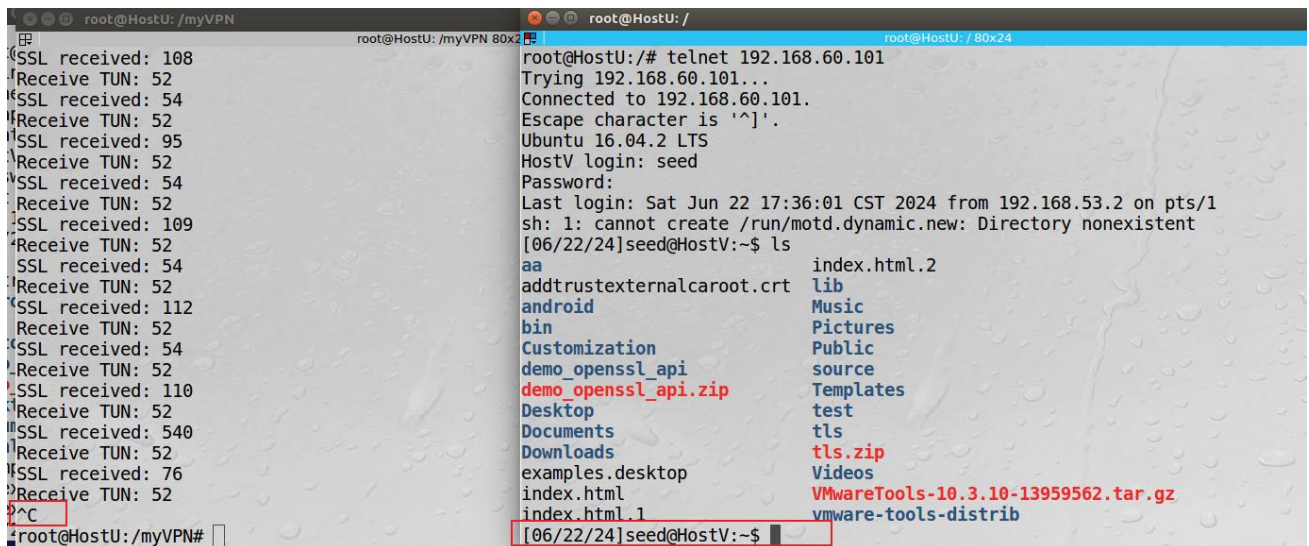
```
ssl_info_vip get_info;  
    SSL_read(ssl, &get_info, sizeof(get_info));  
    printf("%s\n", get_info.msg);  
    char virtual_ip[20];  
    struct in_addr srcvip;  
    srcvip.s_addr = get_info.vip;  
    strcpy(virtual_ip, inet_ntoa(srcvip));  
    printf("Virtual IP:%s\n", virtual_ip);  
  
    int tunfd;  
    tunfd = createTunDevice();  
    char cmd[50];  
    strcpy(cmd, "ifconfig tun0 ");  
    strcat(cmd, virtual_ip);  
    strcat(cmd, "/24 up");  
    system(cmd);
```

## 3 VPN 实现细节

### 3.1 问题 1

**问题:** 回答实验指导手册 4.2 第 6 步提出的问题: 在 HostU 上, telnet 到 HostV。在保持 telnet 连接存活的同时, 断开 VPN 隧道。然后我们在 telnet 窗口中输入内容, 并报告观察到的内容。然后我们重新连接 VPN 隧道。需要注意的是, 重新连接 VPN 隧道, 需要将 vpnserver 和 vpnclient 都退出后再重复操作, 请思考原因是什么。正确重连后, telnet 连接会发生什么? 会被断开还是继续? 请描述并解释你的观察结果。

**答案:**



The image shows two terminal windows. The left window, titled 'root@HostU: /myVPN', displays a series of log messages: 'SSL received: 108', 'Receive TUN: 52', 'SSL received: 54', 'Receive TUN: 52', 'SSL received: 95', 'Receive TUN: 52', 'SSL received: 54', 'Receive TUN: 52', 'SSL received: 109', 'Receive TUN: 52', 'SSL received: 54', 'Receive TUN: 52', 'SSL received: 112', 'Receive TUN: 52', 'SSL received: 54', 'Receive TUN: 52', 'SSL received: 110', 'Receive TUN: 52', 'SSL received: 540', 'Receive TUN: 52', 'SSL received: 76', 'Receive TUN: 52'. The right window, titled 'root@HostU: /', shows a telnet session to 192.168.60.101. It displays the connection process, the escape character '^', the Ubuntu 16.04.2 LTS login prompt, and the password 'seed'. After logging in, it shows the last login time and the directory listing for the user 'seed' in the '/home/seed' directory. The directory listing includes files like 'index.html.2', 'lib', 'Music', 'Pictures', 'Public', 'source', 'Templates', 'test', 'tls', 'tls.zip', 'Videos', 'VMwareTools-10.3.10-13959562.tar.gz', and 'vmware-tools-distrib'. The prompt is '[06/22/24]seed@HostV:~\$'.

图 6 telnet 无响应示意图

在 HostU 上, telnet 到 HostV。如图 6 所示, 在保持 telnet 连接存活的同时, 断开 VPN 隧道。然后我们在 telnet 窗口中输入内容, 发现是没有任何响应的, 无法输入数据并发送, 因为 VPN 隧道断开了, HostU 和 HostV 之间的直接网络路径将丢失, 数据是无法到达 HostV 的。

重新连接 VPN 隧道, 需要先完全退出 vpnserver 和 vpnclient, 然后重新启动它们以建立新的 VPN 连接。是因为 VPN 的重连通常需要重新建立新的加密隧道和相关的网络配置, 例如: VPN 客户端和服务端需要重置它们的状态, 以确保所有连接都是最新的, 并且没有残留的旧连接状态; 重新建立安全连接可能需要重新进行 TLS 握手, 这需要客户端和服务端重新初始化它们的安全参数。

正确重连后, telnet 连接会继续, 并且在断开期间, 我输入的没有响应的字符会立马出现显示在 HostV 的 shell 上。telnet 连接在 VPN 断开时保持在 TCP 层面的活动 (例如, 通过持

---

续发送数据），它会继续存在，但由于 VPN 隧道的断开，数据可能无法成功传输，一旦 VPN 隧道重新建立，如果 telnet 客户端和服务端都配置为使用 VPN 隧道作为路由路径，telnet 会话应该能够恢复并继续传输数据。

## 3.2 问题 2

**问题：**请介绍你的 VPN 的登录协议，即在 SSL 连接建立之后、隧道通信传输之前，SSLVPN 客户端与服务端交互了哪些报文？每个报文的格式是怎样的？每个报文的作用是什么？报文内容取值是如何定义的？报文交互的顺序是怎样的？

**答案：**

### 1. VPN 登录协议概述

#### (1) SSL 连接建立

首先，SSL 连接建立之后，客户端和服务端之间会进行一系列的握手和验证步骤。这些步骤已经在代码中通过 OpenSSL 库的函数实现。

#### (2) 客户端认证报文

在 SSL 连接成功建立后，客户端会发送一系列认证报文到服务端。这些报文包括用户名、密码和最后一个 IP 地址字节。

#### (3) 隧道通信

在认证通过后，客户端和服务端之间开始隧道通信，传输具体的数据包。

### 2. 报文格式与作用

- 用户名报文，格式为 `username`，作用为标识用户身份，由用户输入的用户名，作为字符串发送。
- 密码报文，格式为 `password`，作用为验证用户身份，由用户输入的密码，作为字符串发送。
- 最后一个 IP 地址字节报文，格式为 `last_ip_byte`，作用为配置 TUN 设备和网络路由，本地 TUN 设备的 IP 地址的最后一个字节，作为字符串发送。

### 3. 报文交互的详细过程

#### (1) 客户端与服务端建立 SSL 连接

`SSL_connect` 完成后，SSL 连接建立。服务端通过回调函数 `verify_callback` 验证客

---

户端证书。

### (2) 客户端发送用户名

`SSL_write(ssl, argv[3], strlen(argv[3]))`; 客户端发送包含用户名的报文到服务器, 服务器接收并记录用户名。

### (3) 客户端发送密码

`SSL_write(ssl, argv[4], strlen(argv[4]))`; 客户端发送包含密码的报文到服务器, 服务器接收并验证密码。

### (4) 客户端发送最后一个 IP 地址字节

`SSL_write(ssl, argv[5], strlen(argv[5]))`; 客户端发送包含本地 IP 地址最后一个字节的报文到服务器, 服务器接收并记录该 IP 地址字节, 用于配置网络路由。

### (5) 服务器完成验证并允许隧道通信

验证通过后, 服务器通知客户端, 隧道通信开始。

## 3.3 问题 3

**问题:** 你的 VPN 服务器支持多客户端采用的什么技术? VPN 服务器收到保护子网主机的应答报文时, 如何判断应发送给哪个 VPN 客户端的隧道?

**答案:**

VPN 服务器通常采用多线程或多进程架构来同时处理来自多个 VPN 客户端的连接。每个客户端连接可以在自己的线程或进程中被独立处理, 从而实现并发服务。

VPN 服务器收到应答报文后, 通过查找建立 SSL 连接时记录的虚拟 IP-隧道映射表, 最后, VPN 服务器通过相应的 VPN 隧道将封装后的报文发送给目标 VPN 客户端。

## 4 测试结果与分析

### 4.1 认证 VPN 服务器

#### 1. 用 openssl 检查 VPN 服务器证书信息

```
huyu
Identity: huyu
Verified by: huyu
Expires: 07/21/2024
~ Details

Subject Name
C (Country): CN
ST (State): Hubei
L (Locality): Wuhan
O (Organization): HUSTCSE
OU (Organizational Unit): HUSTCSE
CN (Common Name): huyu
EMAIL (Email Address): U202112146@hust.edu.cn

Issuer Name
C (Country): CN
ST (State): Hubei
L (Locality): Wuhan
O (Organization): HUSTCSE
OU (Organizational Unit): HUSTCSE
CN (Common Name): huyu
EMAIL (Email Address): U202112146@hust.edu.cn

Issued Certificate
Version: 3
Serial Number: 00 9C 09 99 3B 2D A3 05 A0
Not Valid Before: 2024-06-21
Not Valid After: 2024-07-21
```

图 7 服务器证书信息

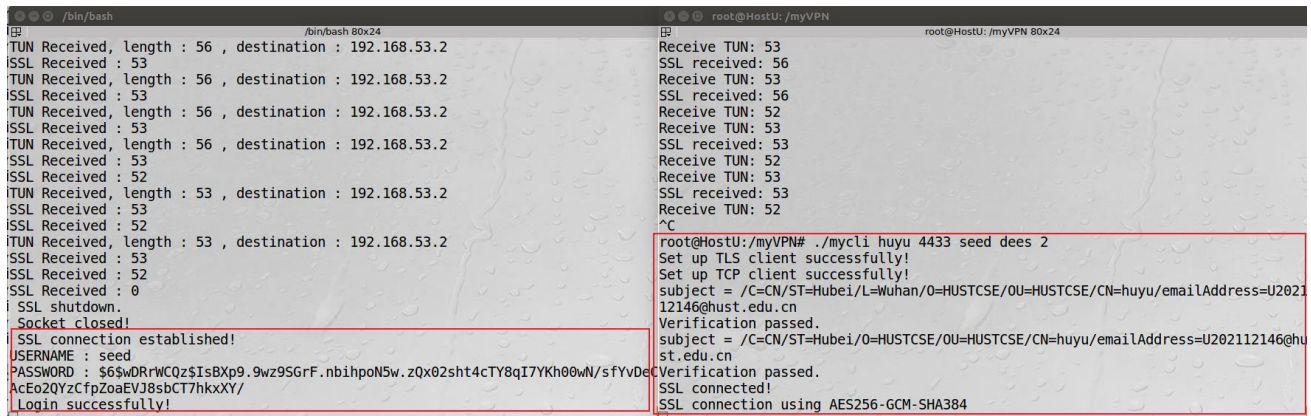
服务器证书如图 7 所示，同时在登录 VPNclient 时，也可以看到证书信息，如图 8 所示。

```
root@HostU:/myVPN# ./mycli huyu 4433 seed dees 2
Set up TLS client successfully!
Set up TCP client successfully!
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
subject = /C=CN/ST=Hubei/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
SSL connected!
SSL connection using AES256-GCM-SHA384
```

图 8 证书相关信息

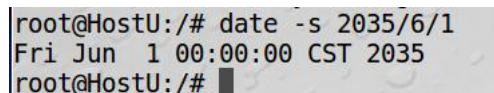
#### 2. 修改 VPN 客户端主机时间到 VPN 服务器证书有效期之后再登录 VPN 服务器





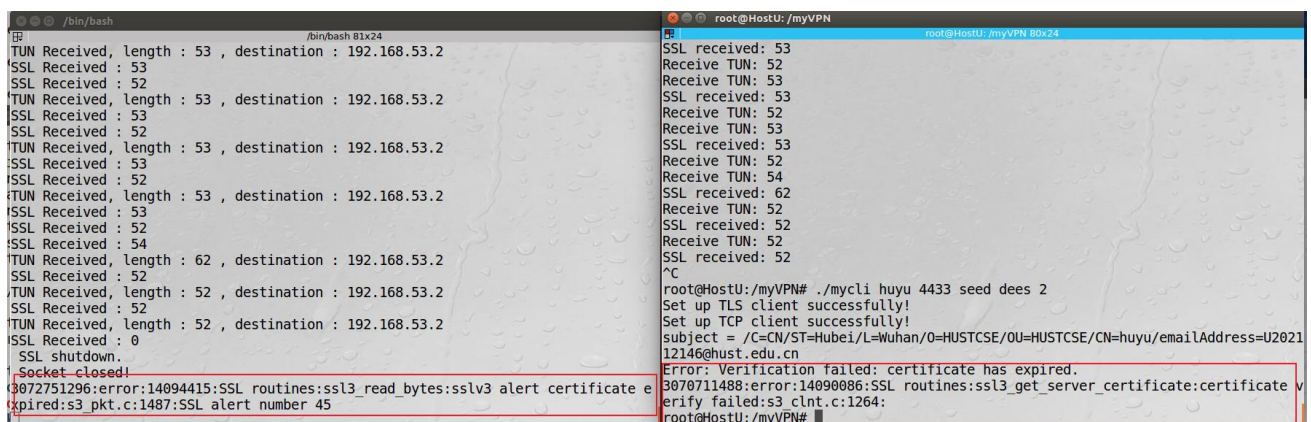
### 图9 正常登录

当证书没有过期时,正常登录情况如图 9 所示,左边 VPNserver 显示 Login successfully, 右边 VPNclient 显示 SSL connected! 。



### 图 10 改变时间

将客户端 HostU 时间调到 2035 年 6 月 1 日（即证书已经过期）。



**图 11 证书过期**

如图 11 所示，可以发现，调完时间后再次登录，VPNserver 和 VPNclient 都会报错，证书已经过期。

## 4.2 认证 VPN 客户端

1. VPN 客户端以错误的用户名或口令登录 VPN 服务器



```

/bin/bash
SSL Received : 52
TUN Received, length : 119 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 54 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 59 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 110 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 54 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 76 , destination : 192.168.53.2
SSL Received : 0
SSL shutdown.
Socket closed!
SSL connection established!
USERNAME : seed
PASSWORD : $6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN/sfYvDeC
AcEo2QYzCfpZoaEVJ8sbCT7hkxXY/
Error: The password is incorrect!
Error: Login failed!
Socket closed!

VPNserver

root@HostU: /myVPN
SSL received: 59
Receive TUN: 52
SSL received: 110
Receive TUN: 52
SSL received: 54
Receive TUN: 52
SSL received: 76
Receive TUN: 52
^C
root@HostU: /myVPN#
root@HostU: /myVPN# ./mycli huyu 4433 seed failure 2
Set up TLS client successfully!
Set up TCP client successfully!
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
SSL connected!
SSL connection using AES256-GCM-SHA384
SSL received: 0
Connection closed!
root@HostU: /myVPN#

VPNclient
```

图 12 错误口令登录

如图 12 所示，当我用错误口令 failure（正确口令为 dees）登录时，服务器会报错，口令错误，登录失败。

## 2. VPN 客户端以正确的用户名口令登录 VPN 服务器

```

/bin/bash
TUN Received, length : 59 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 110 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 54 , destination : 192.168.53.2
SSL Received : 52
TUN Received, length : 76 , destination : 192.168.53.2
SSL Received : 0
SSL shutdown.
Socket closed!
SSL connection established!
USERNAME : seed
PASSWORD : $6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN/sfYvDeC
AcEo2QYzCfpZoaEVJ8sbCT7hkxXY/
Error: The password is incorrect!
Error: Login failed!
Socket closed!

VPNserver

root@HostU: /myVPN
Set up TLS client successfully!
Set up TCP client successfully!
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
SSL connected!
SSL connection using AES256-GCM-SHA384
SSL received: 0
Connection closed!
root@HostU: /myVPN# ./mycli huyu 4433 seed dees 2
Set up TLS client successfully!
Set up TCP client successfully!
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
subject = /C=CN/ST=Hubei/L=Wuhan/O=HUSTCSE/OU=HUSTCSE/CN=huyu/emailAddress=U202112146@hust.edu.cn
Verification passed.
SSL connected!
SSL connection using AES256-GCM-SHA384

VPNclient
```

图 13 正确口令登录

如图 13 所示，当我用正确口令 dees 登录时，服务器显示登录成功。

## 4.3 加密隧道通信

要求：VPN 客户端 ping 内网主机，wireshark 在 VPN 服务器外口截包检查

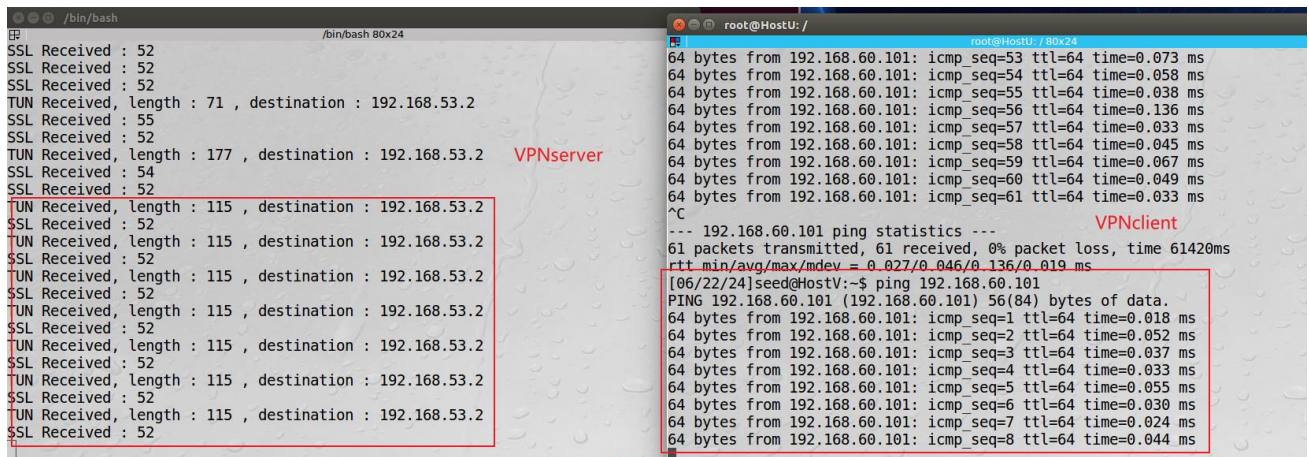


图 14 VPNclient ping 内网主机示意图

如图 14 所示，HostU 主机成功 ping 通内网主机 192.168.60.101。

调用 wireshark 抓包，可以看到加密的 TLS 隧道通信，且报文内容是加密之后的乱码，如图 15 所示。

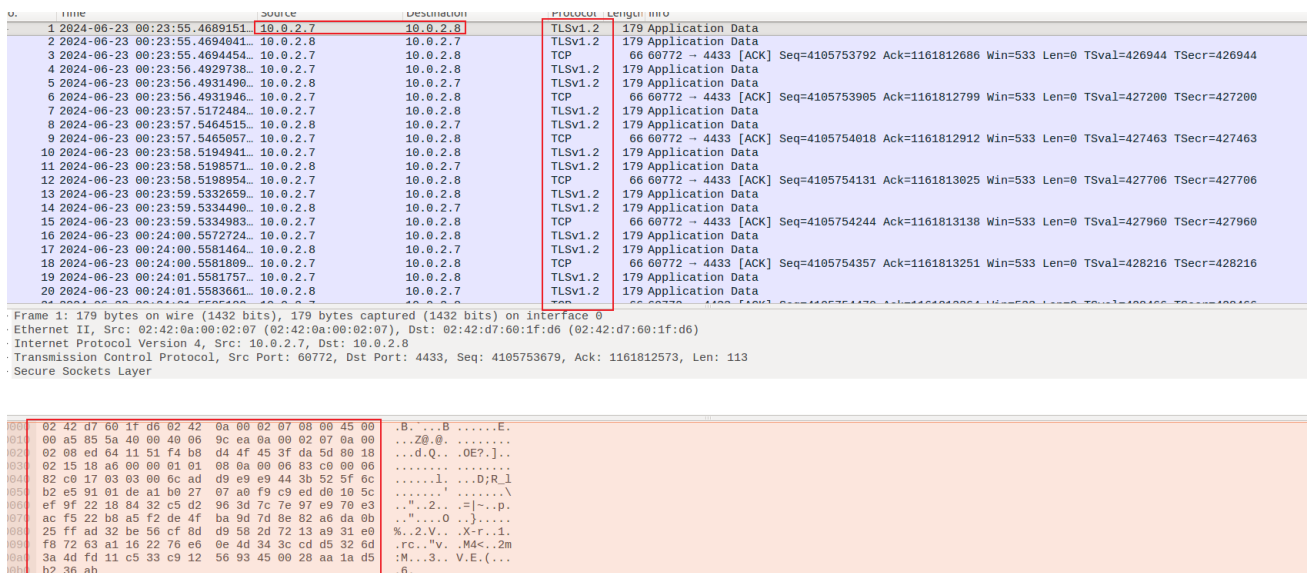


图 15 wireshark 抓包结果

## 4.4 支持多客户端

### 1. 开启 2 个以上 VPN 客户端容器，同时登录 VPN 服务器，分别测试 telnet 通信

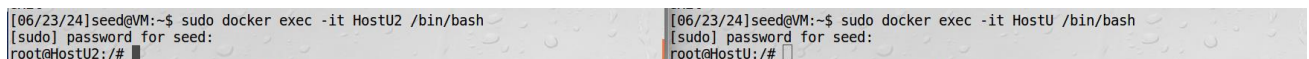


图 16 开启两个 docker

新建 docker 进行测试，同时在 HostU(10.0.2.7)和 HostU2(10.0.2.6)上运行客户端程



序，如图 16、17 所示。

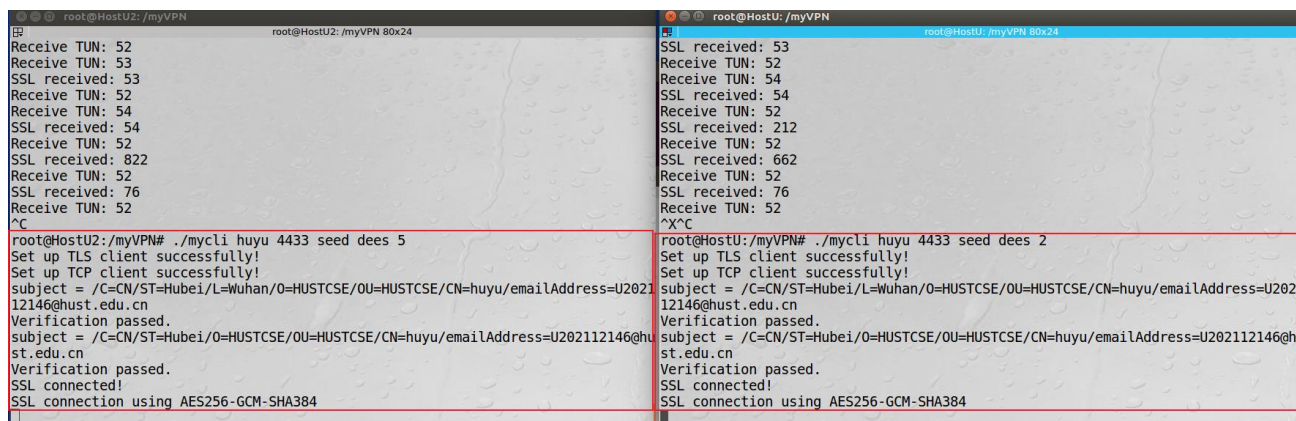


图 17 同时登录 VPNclient

在两个 VPNclient 同时测试 telnet 连接，如图 18 所示，可以看到连接成功，且 ls 测试指令也能正常执行。

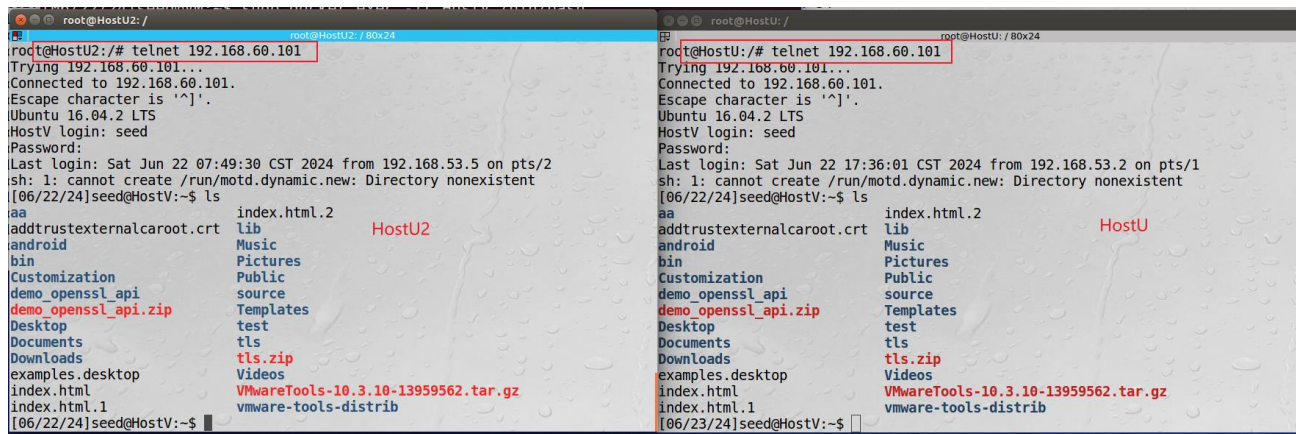


图 18 双开测试 telnet 连接

## 2. 断开其中一个 VPN 客户端，测试另外一个的隧道通信

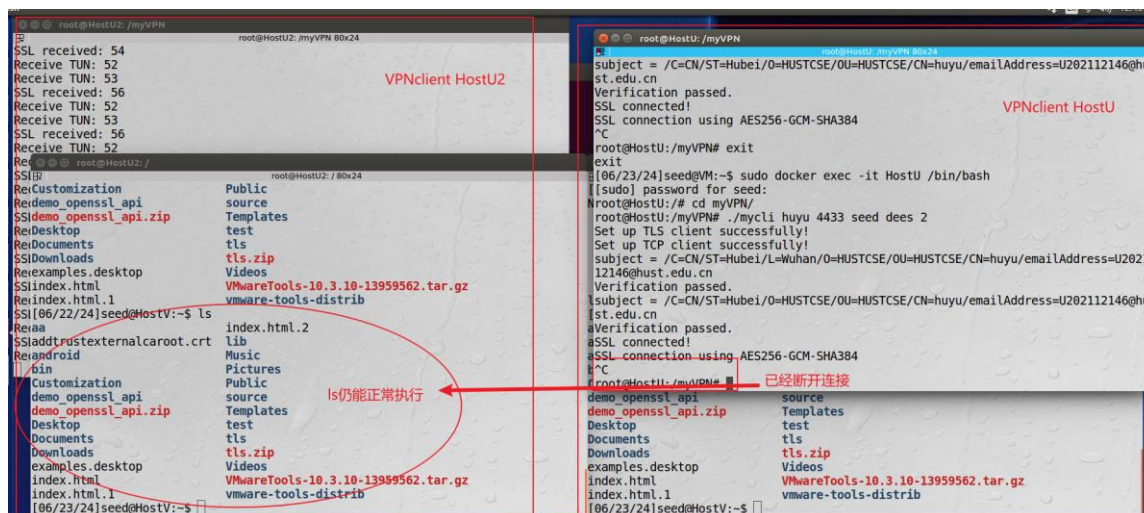


图 19 客户端间影响

如图 19 所示，断开 HostU 的 VPNclient 之后，HostU2 仍然能 telnet 通信，由此可见隧道保持，通信不受影响，运行稳定。

## 4.5 易用性和稳定性

### 1. VPN 客户端虚拟 IP 获取

手动分配的，获取分配的 2 和 5。

```
// redirect and routing
char cmd[100];
sprintf(cmd, "sudo ifconfig tun0 192.168.53.%s/24 up && sudo route add -net 192.168.60.0/24 tun0",
        argv[5]);
system(cmd);
```

图 20 手动 IP 获取

### 2. VPN 客户端虚拟 IP 配置

程序自动添加，自动配置的 192.168.53.2 和 192.168.53.5。

```
char cmd[100];
sprintf(cmd, "sudo ifconfig tun0 192.168.53.%s/24 up && sudo route add -net 192.168.60.0/24 tun0",
        argv[5]);
system(cmd);
```

图 21 自动分配 IP

### 3. VPN 客户端内网路由配置

程序自动添加。

```
// redirect and routing
char cmd[100];
sprintf(cmd, "sudo ifconfig tun0 192.168.53.%s/24 up && sudo route add -net 192.168.60.0/24 tun0",
        argv[5]);
system(cmd);
```

图 22 自动路由配置

### 4. 正常使用时的稳定性

运行没有任何问题，十分稳定。

---

## 5 体会与建议

### 5.1 心得体会

我觉得收获最大的地方是对用户端虚拟 IP 的分配和管理，以及服务器对用户端连接的处理。固定网段的虚拟 IP 用于区分不同的客户端，由服务器顺序空位指定，服务器中的映射表，将 IP 地址和 SSL 文件描述符绑定，很方便管理。对每个客户端创建一个子线程而非新建进程，节约系统资源，方便信息传递。

这次 VPN 实验，对 SSL 通信的过程，openssl 的使用有了一些了解，实验总体难度不大，调试过程也比较有趣，是很有意思的一个实验。通过本次 VPN 实验，我深刻理解了虚拟专用网络(VPN)的工作原理和实现过程。将网络安全理论知识与实践操作相结合，加深了我对 TLS/SSL VPN 技术的认识。实验中对 TUN/TAP 虚拟网络设备、IP 隧道、路由配置、公钥加密、PKI 和 X.509 证书等概念有了更加深入的了解。在实验过程中，遇到了一些配置错误和程序 bug，通过查阅资料、不断尝试和调试，我学会了如何分析问题并找到解决方案。实验让我意识到网络安全的重要性，特别是在配置 VPN 时对数据加密和身份认证的重视，增强了我的安全意识。

### 5.2 意见建议

在实验中，环境配置占据了大量时间。建议可以提供更加自动化的配置脚本，简化环境搭建过程，让学生能够更专注于实验本身。

虽然本次实验涵盖了 VPN 的基本实现，但缺乏一些高级特性的介绍，如动态密钥更新、多因素认证等。建议在未来的实验中加入更多高级特性的实践。

在任务书中，我们可以考虑引入当前广泛使用的 VPN 技术，并与其他类型的 VPN 进行比较分析，探索现代 VPN 的高级功能，如多跳连接、流量分流、带宽管理等，让学生了解 VPN 技术在实际部署中的多样性和复杂性。