

“计算机网络安全”实验报告

——VPN实验

|  |  |
| --- | --- |
| **院 系：** | 网络空间安全学院 |
| **专业班级：** |  |
| **姓 名：** |  |
| **学 号：** |  |
| **日 期：** | 2024年月日 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **评分项** | **实验报告评分**  **（50%）** | **检查单分数**  **（50%）** | **综合得分** | **教师签名** |
| **得分** |  |  |  |  |

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，详细设计说明翔实准确  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：回答准确清晰，实现方法技术优良，与设计及代码一致  18-11：回答较准确清晰，实现方法一般，与设计及代码有偏差  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：心得体会真实，意见中肯、建议合理可行，体现个人思考  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

[1 实验原理 1](#_Toc103367188)

[2 VPN系统设计 2](#_Toc103367189)

[2.1 概要设计 2](#_Toc103367190)

[2.2 详细设计 4](#_Toc103367191)

[3 VPN实现细节 8](#_Toc103367192)

[4 测试结果与分析 10](#_Toc103367193)

[5 体会与建议 16](#_Toc103367194)

[5.1 心得体会 16](#_Toc103367195)

[5.2 意见建议 16](#_Toc103367196)

# 实验原理

本实验基于TLS/SSL技术，并利用TUN虚拟网络设备创建了主机到主机间的安全通信隧道，实现了一个功能完善的虚拟专用网络工具：ltVPN。

ltVPN基于TUN技术创建虚拟网络设备，配置虚拟IP网络及路由，实现IP数据包的发送、接收与转发，利用TLS/SSL技术在主机间进行IP数据包的安全转发，实现了多主机的虚拟局域网，使虚拟局域网中的主机间数据能够安全地通过公共网络传输。

实验网络拓扑结构如图1所示，共有局域网192.168.11.0/24、互联网10.12.1.0/24、虚拟局域网192.168.12.0/24三个网络，互联网与局域网间由网关阻隔，无法直接通信，由VPN服务器、VPN客户端组成了虚拟局域网，虚拟局域网间可相互通信。若需要虚拟局域网与局域网进行相互通信，则由VPN服务器为VPN客户端配置访问局域网的路由，此时VPN服务器作为虚拟局域网与局域网间的网关，负责IP数据包在虚拟局域网与局域网间的转发。

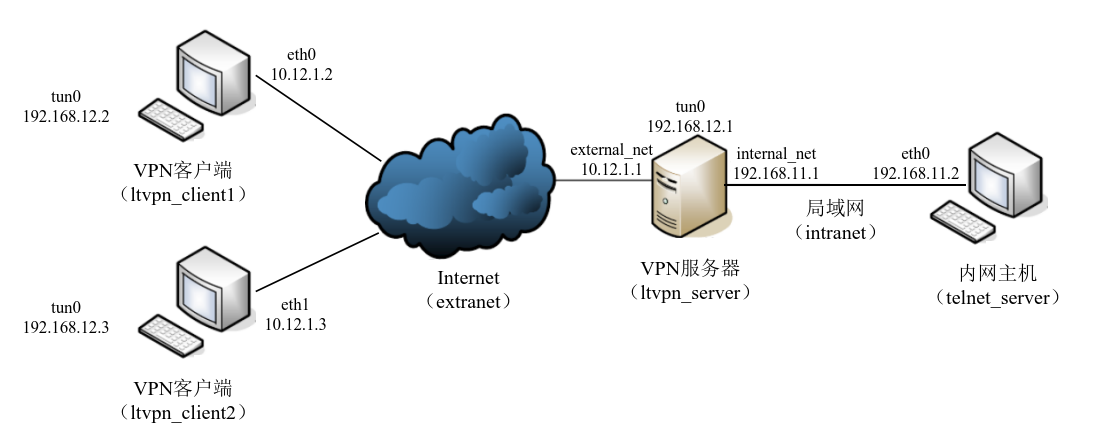


图1 实验网络拓扑结构

VPN客户端与VPN服务器间使用TLS加密隧道，客户端通过检查服务器签名证书保证服务器的准确性，服务器签名证书有效才能进行传输，避免服务器被伪造，客户端与服务器间的数据由TLS进行加密，避免数据传输过程中被监听与篡改。VPN客户端与VPN服务器连接时需要进行登陆，客户端的用户名与口令由服务器管理员进行注册，口令使用加盐哈希算法保护，避免明文口令的泄漏。

VPN网络的虚拟地址、网络路由均由VPN服务器配置文件设置，VPN客户端登陆后自动从服务器获取网络配置，避免手动配置造成的网络异常，VPN网络可基于用户进行配置，可为不同用户配置可访问网络区域，实现用户访问权限控制。

# VPN系统设计

## 概要设计

### 系统框架设计

ltVPN采用模块化设计，含配置解析模块、SSL/TLS模块、登陆模块、TUN模块及多客户端VPN模块，各模块组合实现系统的各种功能。

配置解析模块用于处理配置文件以及命令行参数，通过解析服务器、客户端对应的配置文件，为系统配置全局参数，实现服务启动自动配置，无需手动传递各项参数。

SSL/TLS模块用于创建客户端到服务器的安全通信隧道，为登陆信息与VPN数据提供安全传输，并基于服务器签名证书提供了服务器的防伪造功能。

登陆模块用于客户端的登陆验证，服务器端基于用户列表储存用户登陆信息及用户配置信息，用户成功登陆后服务器将用户配置信息发送给客户端实现客户端VPN配置。

TUN模块用于创建虚拟网卡，并基于网络配置设置虚拟IP地址、配置系统路由。

多客户端VPN模块负责每一个客户端连接的VPN数据处理，实现多主机的虚拟局域网。

### TUN框架结构

ltVPN中使用了TUN技术，TUN是虚拟网络内核驱动程序,模拟网络层设备,处理的是IP数据包，VPN系统的TUN框架设计如图2所示。

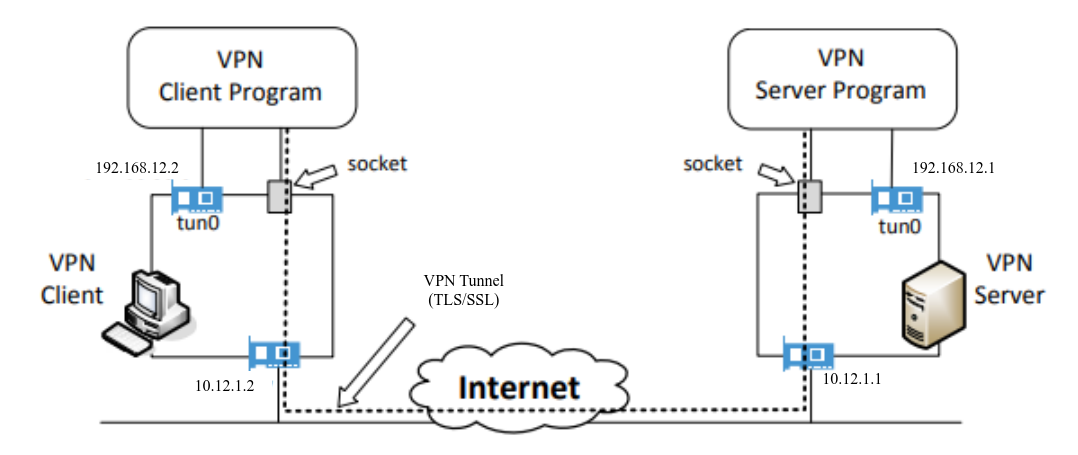


图2 VPN客户端与服务器

VPN服务器与客户端均创建tun网卡设备，网卡的数据由VPN程序进行处理，当有IP数据包发送到tun0网卡时，VPN程序接收数据包，经VPN隧道传送到对端主机，当VPN程序接收到来自VPN隧道的IP数据包时，VPN程序将该IP数据包由tun0网卡设备发出，实现不同主机间tun0网卡的虚拟连接，及组建虚拟局域网。

### 多客户端设计

在2.1.2中设计了单客户端与服务器的虚拟网络通信，在多客户端情况下，服务器与每一个客户端间都有一条VPN隧道，服务器需要监听每一个客户端对应的socket套接字，单服务器进程难以应对该情况，因此服务器为每一个客户端连接创建子进程，由子进程接收客户端对应的VPN隧道数据，子进程接收到来自VPN隧道的IP数据包时，需要将数据包发送给服务器主进程，由主进程从tun0网卡发出，子进程与主进程间的IP数据包使用管道实现。

多客户端的设计框架如图3所示，上述解决了来自客户端VPN隧道发往tun0网卡的IP数据包，来自tun0网卡发送给客户端隧道的IP数据包无法直接发出，VPN主进程接收到来自tun0的IP数据包后，需要提取IP数据包中的目的IP地址，根据目的IP地址找到对应的VPN客户端及对应的VPN隧道子进程，将IP数据包发往对应子进程的管道，子进程接收从管道中接收到来自主进程的IP数据包后，将IP数据包由VPN隧道发往对应的IP客户端。

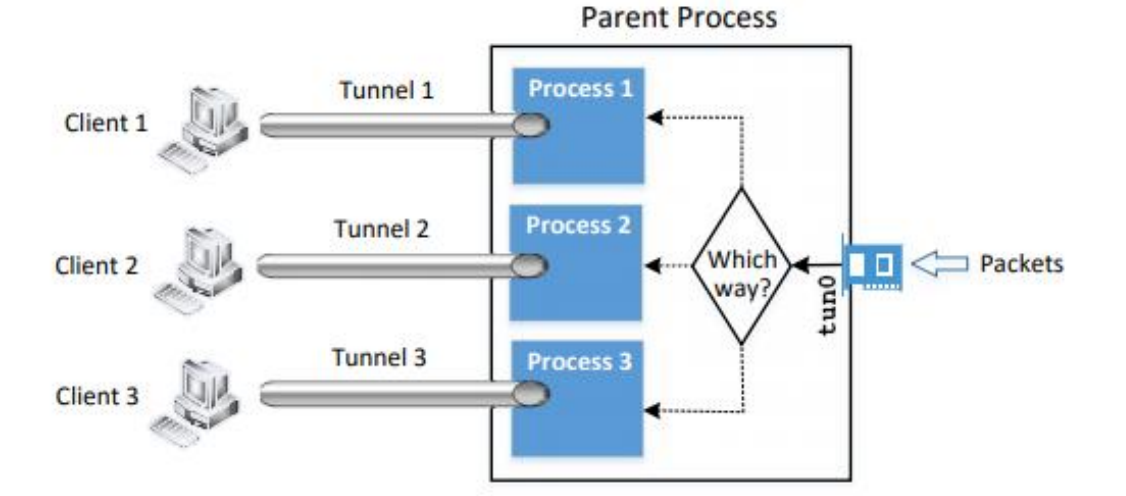


图3 多客户端设计

## 详细设计

### 系统流程图

客户端系统流程图如图4所示。

客户端程序启动后，解析命令行参数及配置文件，获取服务器地址信息、用户名密码等，随后初始化SSL并连接服务器，对服务器签名证书进行校验，校验通过后开始与服务器通信进行登陆操作，登陆成功后获取服务器发送的配置信息，根据配置信息设置tun虚拟设备以虚拟网络，完成设置后进行VPN循环。

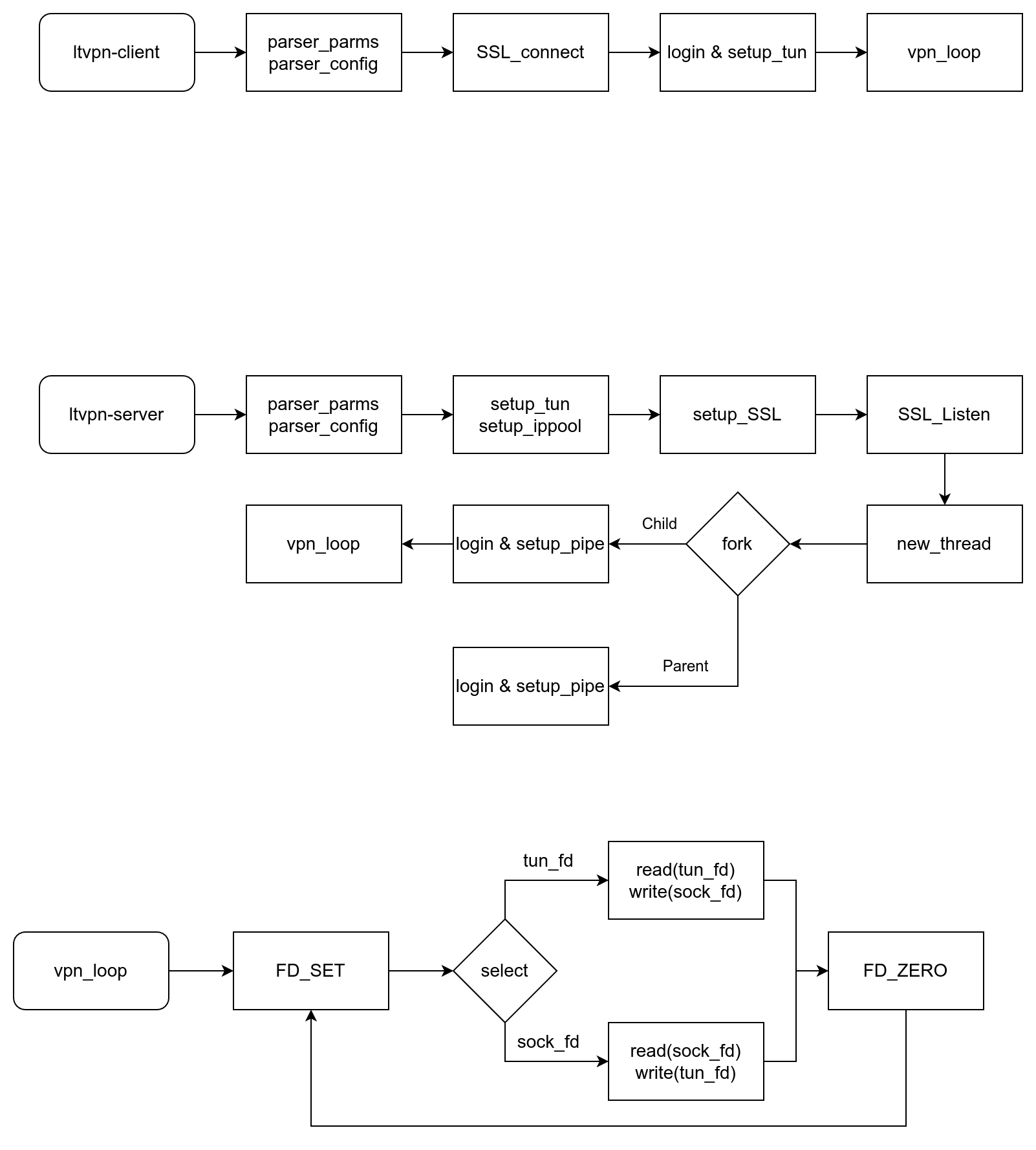


图4 客户端系统流程图

服务器端系统流程图如图5所示。

服务器程序启动后，解析命令行参数及配置文件，获取服务器私钥信息，VPN网络信息等，随后加载服务器证书，初始化SSL服务器，等待客户端连接。客户端连接后为客户端创建线程处理客户端登陆，线程中使用fork创建子进程负责与客户端进行soket通信，父进程与子进程使用pipe通信，共同完成登陆、IP地址分配、客户端网络设置，随后副进程结束客户端线程，子进程处理客户端socket数据。

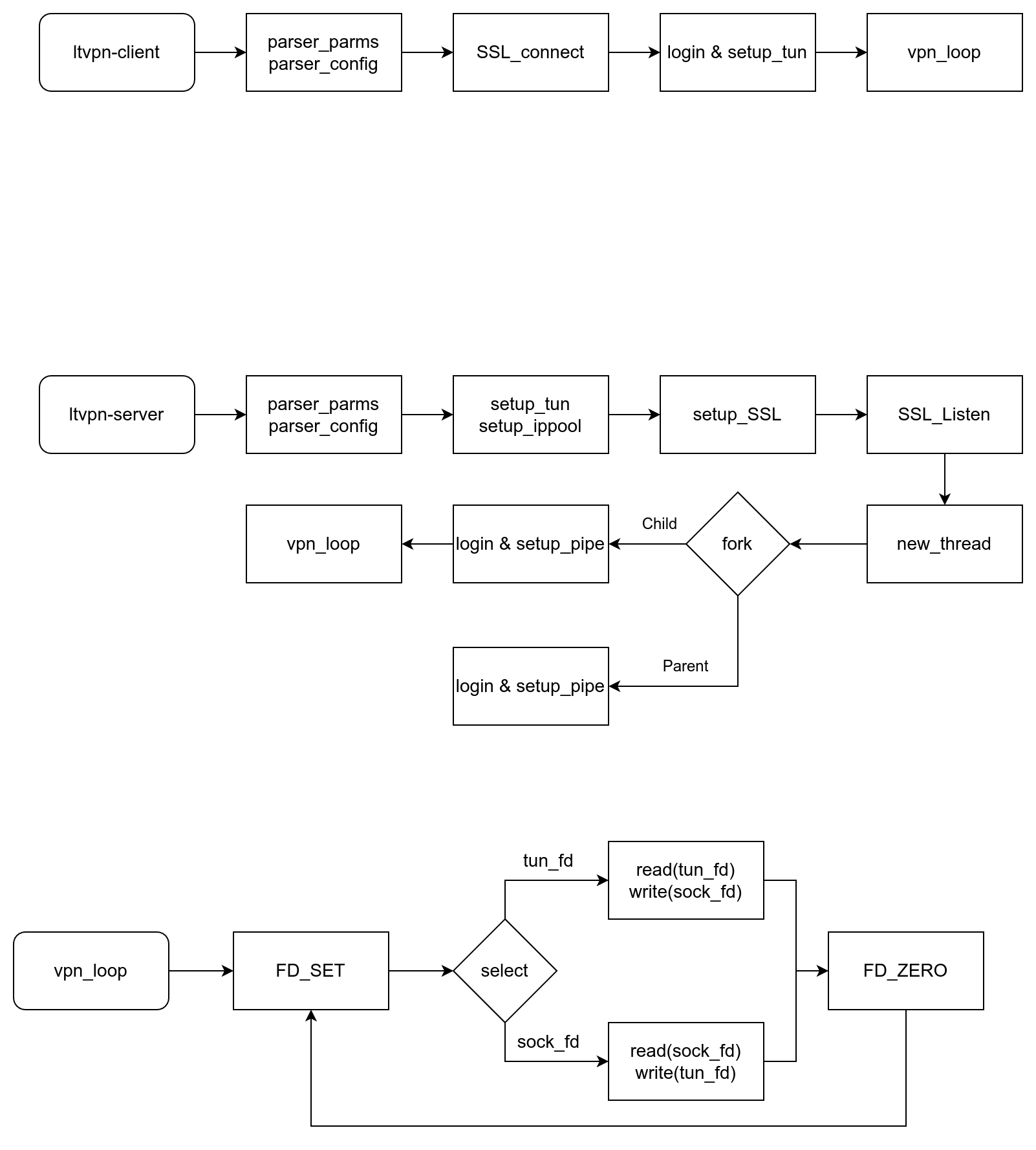


图5 服务器端系统流程图

VPN循环流程图如图6所示。

在VPN循环中需要同时监听tun虚拟设备数据和socket数据，当有IP报文发往tun虚拟设备时，select选择tun\_fd，读取IP报文并转发到sock\_fd，当socket接收到IP报文时，select选择sock\_fd，读取IP包围并转发到tun\_fd。

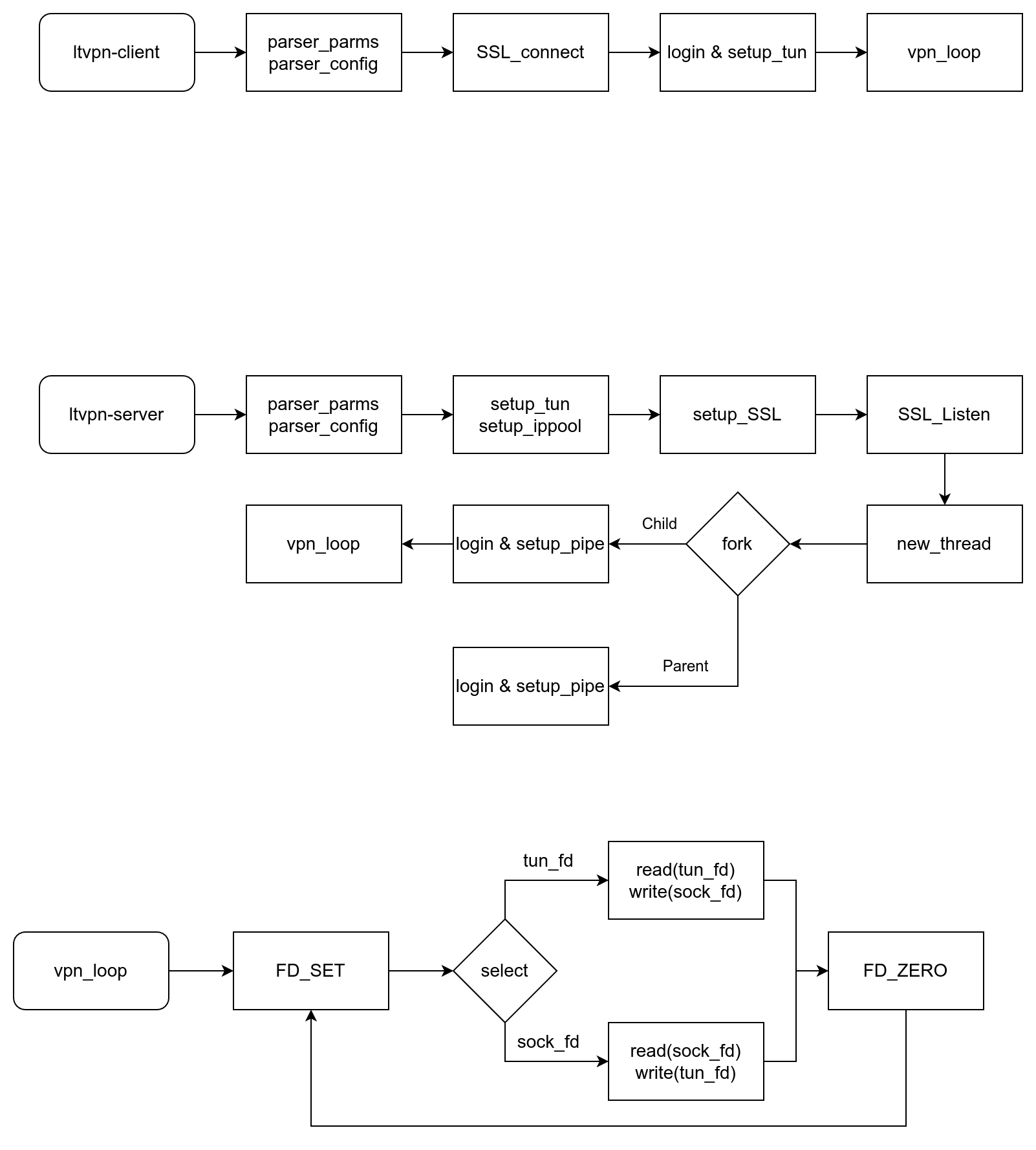


图6 VPN循环流程图

### 配置解析模块

配置解析模块包括命令行参数解析parser\_parms与配置文件解析parser\_args，命令行参数主要包括’-c’配置文件目录，缺省时自动设置’./configs’或’/etc/ltvpn/’，’-d’ daemon模式运行，指定时vpn程序将后台运行，关闭终端时vpn程序继续运行，服务器程序ltvpn-server还有运行模式参数，’run’表示启动vpn服务器，’add\_user’表示添加新用户配置。

配置文件解析根据服务器或客户端选择解析’server.conf’或’client.conf’，根据配置文件中的键值对设置全局变量信息，供vpn程序运行初始化使用。

服务器配置包含服务器证书、私钥、端口、VPN网络等，以及用户列表文件，服务器路由列表参数可缺省，缺省时VPN网络只能内部互连，无法通过网关访问局域网。服务器配置：

# 服务器证书

cert=server.crt

# 服务器私钥

key=server.key

# 私钥口令

key\_passwd=passwd

# 端口

port=8443

# VPN网络

ip\_cidr=192.168.11.0/24

# 路由网络 已包含cidr网络

network=192.168.12.0/24

# 用户列表

user\_list=user.list

客户端配置包含CA证书，服务器地址及端口，以及登陆的用户名和密码，登陆用户名与密码可缺省，缺省时在VPN程序启动时需手动输入用户名与密码。客户端配置：

# CA证书

cacert=ca.crt

# 服务器地址

server=vpn.example.com

# 服务器端口

port=8443

# 用户名

user=user\_name

# 密码

passwd=passwd

### TLS/SSL模块

TLS模块主要包含初始化SSL与初始化TCP连接等功能，在服务器端初始化SSL时，需要加载服务器签名证书及私钥，用于证明服务器身份、客户端初始化SSL需加载CA证书，用于校验服务器签名证书。

服务器端初始化TCP时绑定本地端口，随后进入监听状态，等待客户端连接，接收到客户端连接后，为客户端创建SSL连接。

客户端初始化TCP时尝试连接服务器，连接后创建与服务器的SSL通信。

### IP池

ltVPN客户端IP由服务器进行配置，服务器初始化IP池，当客户端登陆需要IP时，从IP池中进行分配，若登陆的用户已分配静态IP，则为用户设置对应的IP，并在IP池中保留该IP地址，避免IP地址的分配冲突。由于存在多线程并发情况，IP池使用加锁机制避免IP分配错误。服务器基于客户端IP在IP池中的索引保存客户端对应的pipe\_fd，当需要与对应客户端通信时，根据客户端地址，获取客户端索引，得到对应客户端子进程的pipe\_fd，完成通信。

IP池模块共有init\_ip\_pool、get\_ip\_index、reserve\_ip、get\_next\_available\_ip和reclaim\_ip功能，提供IP池的初始化、索引计算、IP分配、保留、回收等功能。默认IP池大小为256,对应/24网络。

### TUN模块

TUN模块提供创建tun虚拟网络设备、设置tun虚拟网络设备IP地址、设置路由等功能。对外提供createVPNtun、routeAdd功能，分别用于创建tun虚拟网络设备并设置IP地址、添加路由的功能，创建tun虚拟网络设备后会返回对应的文件描述符tun\_fd，用于转发VPN网络中的IP数据包。

### 登陆模块

客户端与服务器建立SSL连接后，开始客户端的登陆验证，客户端发起登陆请求login\_req，并携带用户名与加盐哈希处理后的密码，报文格式为’[login\_req]user\_name:hash\_user\_passwd’，服务器接收到登陆请求后，提取登陆请求中的用户名与密码，在用户列表中查找对应用户记录，若用户名与登陆密码匹配，查找用户记录中的静态IP地址、网络路由，如静态地址不存在，则从IP池中分配IP地址，若网络路由不存在，则为用户设置默认网络路由，随后将IP地址与网络路由配置以登陆回应login\_rep报文发送给客户端，报文格式为’[login\_rep](success)ip:network’。若用户登陆失败，则不为客户端分配IP地址，不设置路由，以登陆回应报文反馈失败信息，报文格式为’[login\_rep](failed)err\_msg’。

客户端获接收到登陆回应报文后，根据登陆状态得知是否登陆成功，登陆成功则提取登陆回应中的IP地址和路由网络、创建tun虚拟设备设置IP并添加路由网络，进入VPN循环。登陆失败则根据登陆回应报文中的错误消息提示用户。

用户密码使用加盐哈希处理，避免服务器数据泄漏造成用户明文密码泄漏，加盐哈希算法首先使用用户名对用户密码异或加密，随后进行两次md5加密，加密算法如下：

1. // 使用用户名异或密码，加盐
2. **for** (**int** i = 0; i < USER\_PASSWD\_LEN; i++)
3. {
4. user\_passwd[i] ^= user\_name[i%user\_name\_len] ^ 0x9E;
5. }
6. memset(md5\_str, '\0', **sizeof**(md5\_str));
7. md5((uint8\_t \*)user\_passwd, USER\_PASSWD\_LEN, md5\_result);
8. **for** (**int** i = 0; i < 16; i++)
9. sprintf(md5\_str+i\*2, "%2.2x", md5\_result[i]);
10. md5((uint8\_t \*)md5\_str, 32, md5\_result);
11. **for** (**int** i = 0; i < 16; i++)
12. sprintf(md5\_str+i\*2, "%2.2x", md5\_result[i]);
13. memset(buf, '\0', size);
14. snprintf(buf, size, "%s:%s", user\_name, md5\_str);

在服务器用户列表中，储存用户记录，包含用户名、用户加盐密码、用户静态IP地址、用户路由网络等，记录格式为’user\_name:hash\_user\_passwd:ip:network’，其中用户名与加盐密码不可省略，IP地址和网络可进行缺省，存在IP地址时为用户分配静态IP地址，不存在时使用IP池分配，路由网络用于限制用户可访问的网络区域，实现访问权限管理。

# VPN实现细节

## 问题1

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。

断开VPN隧道后，telnet继续输入内容，无对应的回显，但是telnet未直接退出。重启VPN隧道后，telnet能够重连，不会断开连接，telnet窗口显示VPN断开时输入的字符。

telnet基于TCP协议，是可靠的、有序的通信，TCP连接建立后，将保持ESTABLISHED状态，及时网络发生中断，连接状态也能够保持。HostU与HostV间的路由依赖客户端与服务器之间SSL隧道，当客户端关闭或SSL隧道关闭后，等价于HostU到HostV间的路由网络发生中断，不会造成TCP连接关闭，当服务器与客户端重新连接后，HostU与HostV间的路由网络恢复，TCP能够正常重连，且由于TCP是可靠传输，在隧道关闭后虽然输入无回显，但是在重新连接后，数据并不会丢失，因此能够正常显示回显。

## 问题2

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客户端与服务器交互了哪些报文？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？

客户端与服务器建立SSL连接后，开始客户端的登陆验证，客户端发起登陆请求login\_req，并携带用户名与加盐哈希处理后的密码，报文格式为’[login\_req]user\_name:hash\_user\_passwd’，服务器接收到登陆请求后，提取登陆请求中的用户名与密码，在用户列表中查找对应用户记录，若用户名与登陆密码匹配，查找用户记录中的静态IP地址、网络路由，如静态地址不存在，则从IP池中分配IP地址，若网络路由不存在，则为用户设置默认网络路由，随后将IP地址与网络路由配置以登陆回应login\_rep报文发送给客户端，报文格式为’[login\_rep](success)ip:network’。若用户登陆失败，则不为客户端分配IP地址，不设置路由，以登陆回应报文反馈失败信息，报文格式为’[login\_rep](failed)err\_msg’。

客户端获接收到登陆回应报文后，根据登陆状态得知是否登陆成功，登陆成功则提取登陆回应中的IP地址和路由网络、创建tun虚拟设备设置IP并添加路由网络，进入VPN循环。登陆失败则根据登陆回应报文中的错误消息提示用户。

## 问题3

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？

VPN服务器采用多进程技术和管道技术实现多客户端连接，服务器为每一个客户端连接创建子进程，由子进程接收客户端对应的VPN隧道数据，子进程接收到来自VPN隧道的IP数据包时，需要将数据包发送给服务器主进程，由主进程从tun0网卡发出，子进程与主进程间的IP数据包使用管道实现。

来自tun0网卡发送给客户端隧道的IP数据包无法直接发出，VPN主进程接收到来自tun0的IP数据包后，需要提取IP数据包中的目的IP地址，根据目的IP地址找到对应的VPN客户端及对应的VPN隧道子进程，将IP数据包发往对应子进程的管道，子进程接收从管道中接收到来自主进程的IP数据包后，将IP数据包由VPN隧道发往对应的IP客户端。

# 测试结果与分析

## 服务器证书生成及签名

ltVPN项目基于配置文件编写了证书生成脚本，在$CONFIG\_DIR/key.conf文件中编写对应的证书及私钥设置，运行./scripts/init\_key.sh即可一键生成对应的服务器证书、并对证书进行签名（如图7所示）。

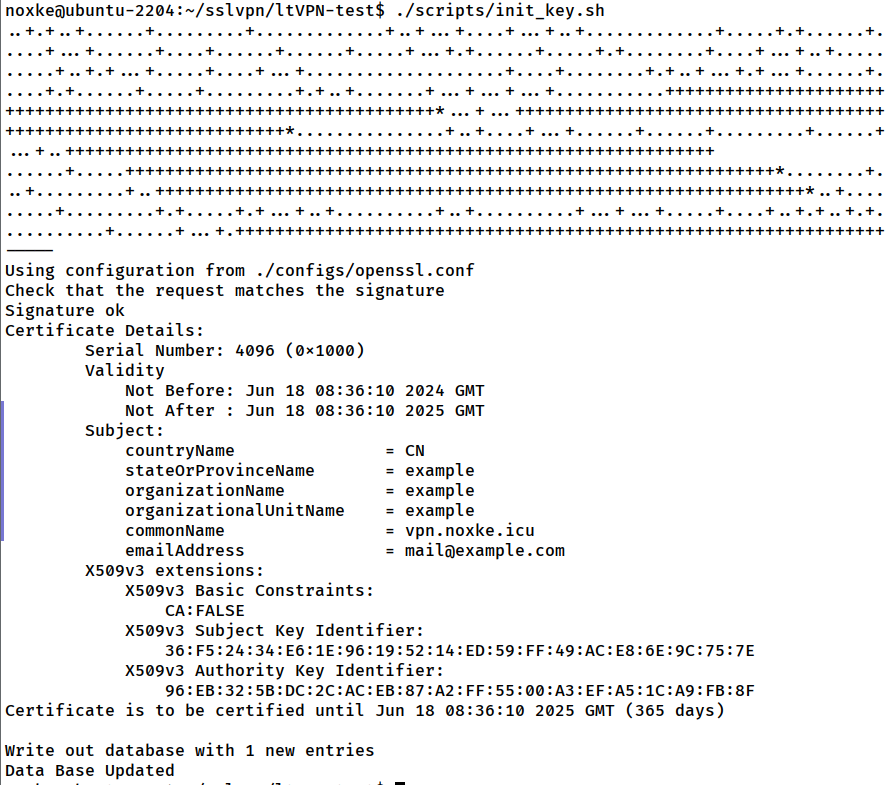


图7 签名证书生成

## 服务器新建用户

ltvpn-server使用add\_user参数启动进入用户添加模式，如图8所示，创建用户noxke1，IP地址为192.168.12.5/24，路由网络默认，创建用户noxke2，IP地址默认，路由网络192.168.11.0/24。

在此配置下，noxke1用户登陆后IP地址为静态IP 192.168.12.5/24，无局域网路由，无法访问局域网，noxke2用户登陆后IP动态分配，存在局域网路由，能够访问局域网主机。

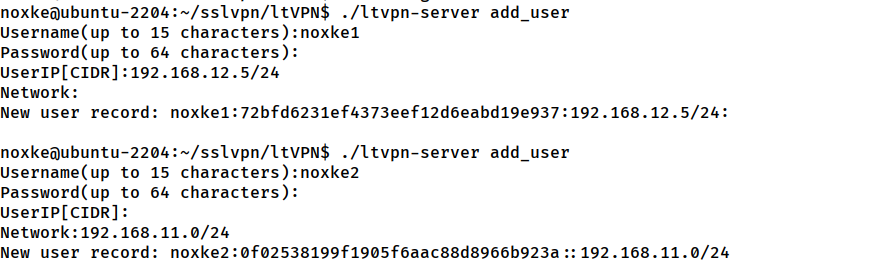


图8 新建VPN用户

## docker测试环境搭建

使用docker-compose开启测试docker环境，将自动创建局域网：internal\_network 192.168.12.0/24以及创建外部网络：external\_network 10.12.1.0/24。启动三台docker容器，容器1 telnet\_server（192.168.11.2/24），容器2 vpn\_client1（10.12.1.2/24），容器3 vpn\_client1（10.12.1.2/24），telnet与vpn\_client位于不同网络，无法之间连接，vpn\_client1与vpn\_client2均位于外部网络。

## VPN客户端与服务器连接测试

网关主机有外部网络地址（10.12.1.1/24）与局域网地址（192.168.11.1），运行./scripts/openfwd.sh脚本开启网关防火墙及转发，运行ltvpn-server run启动VPN服务器，如图9所示。

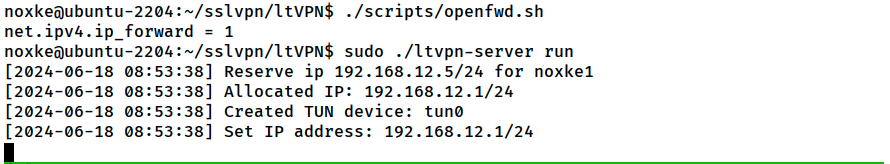


图9 VPN服务器启动

连接到vpn\_client1，使用./scripts/set\_hosts.sh脚本设置服务器host，修改client.conf使用noxke1登陆，运行./ltvpn-client启动VPN客户端，如图10所示，此时服务器端显示noxke1上线，如图11所示，查看vpn\_client1的网卡信息与路由信息，查看到虚拟网卡tun0，ip地址为192.168.12.5/24，路由仅存在访问VPN网络的路由192.168.12.0/24，如图12所示。

使用ping测试VPN客户端与VPN网关的连接，如图13所示。

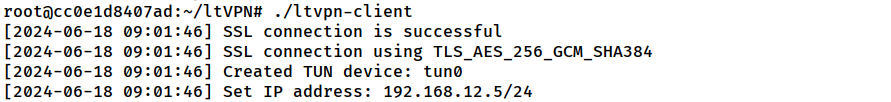


图10 VPN客户端启动

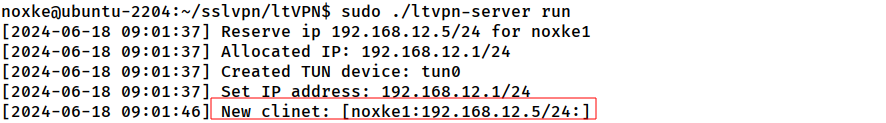


图11 VPN客户端登陆信息

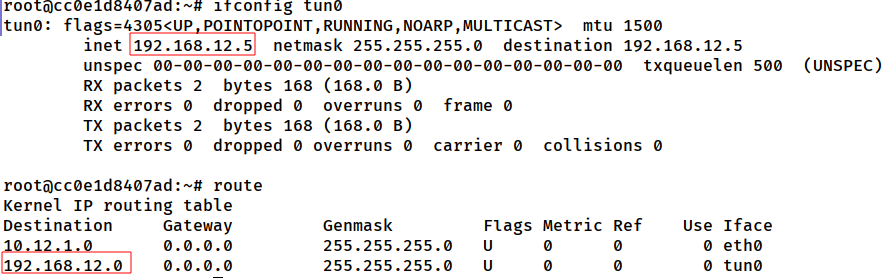


图12 虚拟网卡及路由信息

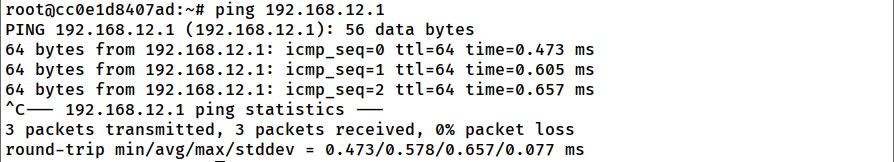


图13 VPN连接测试

根据上述测试结果，vpn\_client1成功登陆noxke1用户，并分配了静态IP 192.168.12.5/24，仅分配了VPN网络内路由，无法访问局域网，VPN客户端与服务器能够正常通信。

## 服务器证书过期测试

使用./scripts/set\_time.sh脚本将vpn\_client1时间修改到1年后，再次尝试连接启动VPN客户端，VPN客户端显示服务器证书过期，中断连接，服务器显示登陆失败，如图14、15所示。

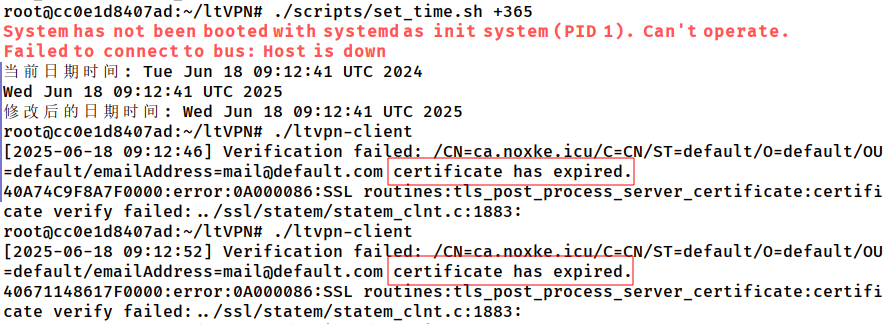


图14 证书过期测试

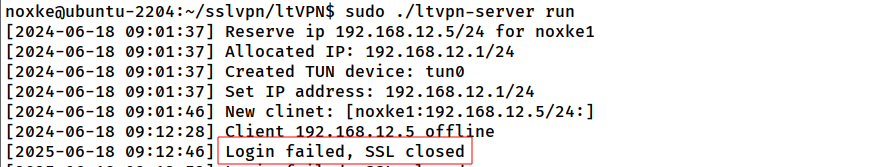


图15 客户端连接失败

## 多客户端连接测试

连接到vpn\_client2，使用./scripts/set\_hosts.sh脚本设置服务器host，修改client.conf使用noxke2登陆，运行./ltvpn-client启动VPN客户端，如图16所示，此时服务器端显示noxke2上线，如图17所示，查看vpn\_client2的网卡信息与路由信息，查看到虚拟网卡tun0，ip地址为192.168.12.2/24，路由存在访问VPN网络的路由192.168.12.0/24与访问局域网的路由192.168.11.0/24，如图18所示。

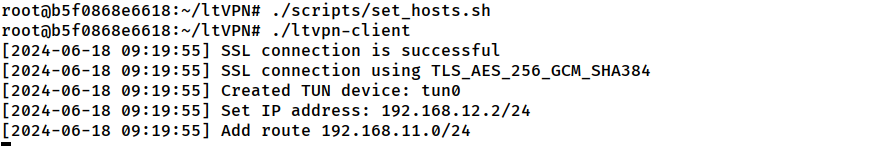


图16 VPN客户端启动

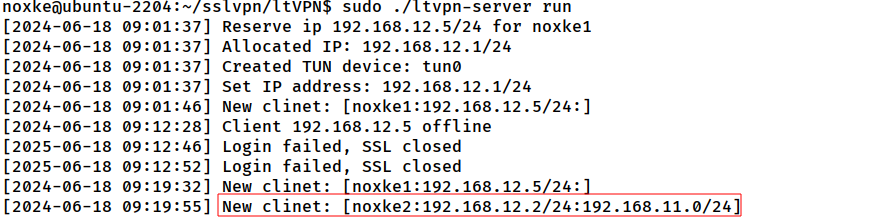


图17 VPN客户端登陆信息

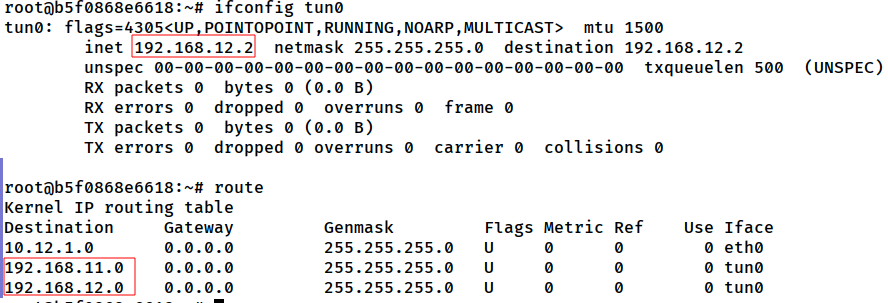


图18 虚拟网卡及路由信息

使用vpn\_client2对VPN网关与vpn\_client1进行ping测试，如图19所示，vpn\_client1与vpn\_client2间可正常通信，即虚拟局域网能能够正常通信。

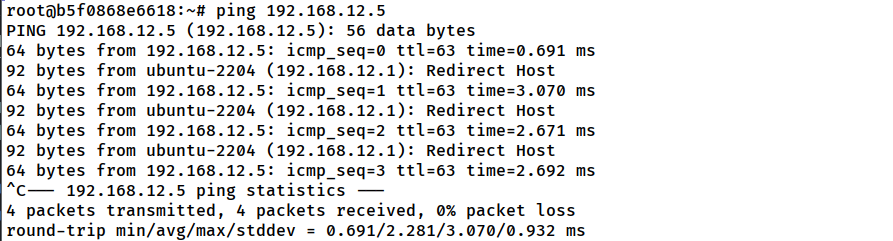


图19 VPN网络通信测试

## VPN内网访问测试

在上面的网络配置中，vpn\_client2具有到局域网192.168.11.0/24的路由，使用ping测试是否能够访问telnet\_server，如图20所示，vpn\_client2可以访问telnet\_server。

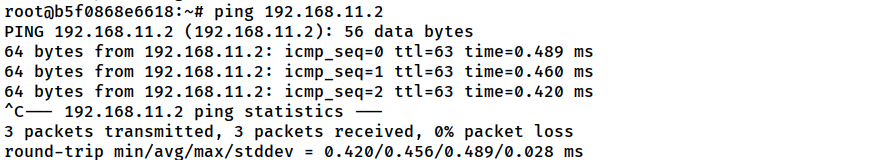


图20 内网访问测试

使用telnet测试vpn\_client2与telnet\_server间的连接，如图21所示，能够正常连接并进行命令执行。

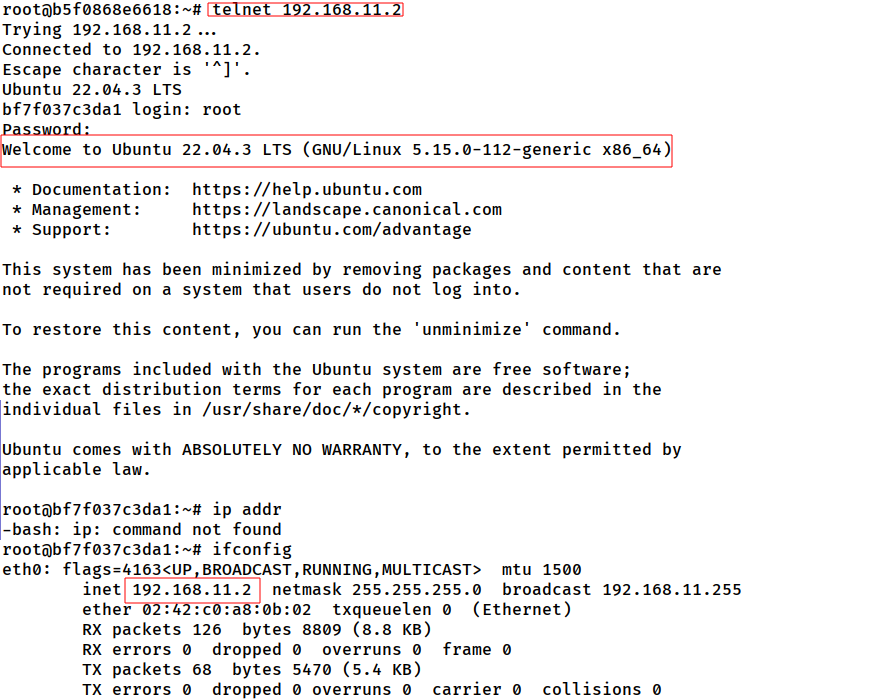


图21 telnet连接测试

## VPN连接带宽测试

在vpn\_client1与telnet\_server启动iperf3服务器，使用vpn\_client2分别测试与VPN虚拟局域网客户端的通信带宽和与局域网的通信带宽（测试CPU：i5-8265u）。

如图22所示，VPN网络内部带宽约为400Mbps，且连接稳定。

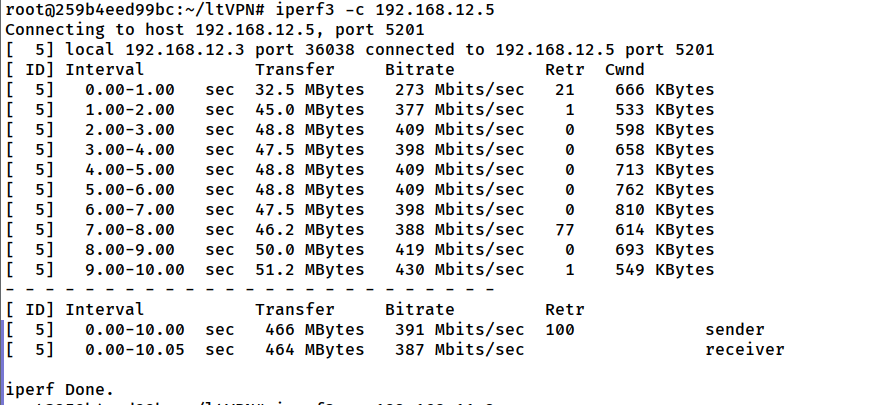


图22 VPN虚拟局域网带宽

如图23所示，VPN网络与局域网间带宽约为630Mbps，且连接稳定。

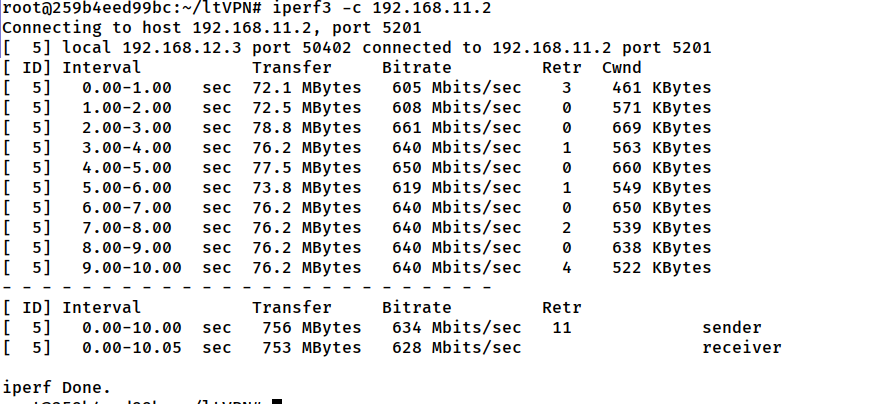


图23 VPN与局域网带宽

# 体会与建议

## 心得体会

在本次实验中，基于TLS/SSL与TUN技术实现了一个轻量化的VPN工具，所实现的ltVPN连接稳定且性能良好，能够支持多客户端同时访问，且能够支持通过VPN网关实现内部网络访问。

在ltVPN项目中，处理实验任务要求外，我额外新增了很多的功能特性，旨在简化ltVPN的使用配置，提升项目的可移植性等。首先使用了基于配置文件解析的方式，实现了VPN客户端与服务器的一键运行，无需手动配置网络，降低了ltVPN工具的使用难度。其次还使用了基于用户配置的方式为用户分配IP地址与设置网络路由，基于用户实现网络的访问控制。另外使用模块化设计，保持了代码仓库的简洁，并且提升了代码的可复用性，基于模块化设计还可新增功能模块为VPN添加更多的功能。在ltVPN项目的测试中，除了测试基本的网络连接功能，还使用了iperf3工具测试了VPN间以及VPN、局域网间的通信带宽，对ltVPN的性能有了基本的了解。

通过本次实验，除了学习TLS/SSL安全通信和基于TUN的VPN技术外，还学习了网络程序的设计思路，多进程间基于管道通信等内容，了解了如何使用多进程技术构建网络服务程序。

## 意见建议

本实验的内容丰富，难度适中，并且实验手册详细，非常适合学习使用。

个人认为本实验的不足之处是实验课时较短、在实验课堂中几乎不可能完成整个项目的构建，考虑到本学期学生要忙于就业或考研，本实验需要占用大量的课外时间，增大了学生的负担，个人认为可以考虑增加实验课程学时，使课堂内即可完成大部分任务，或减少实验任务量，提供一个实验项目，由学生补充关键代码完成实验，以此减少课外时间的占用。