

华中科技大学

网络安全学院

本科：《计算机网络安全实验》报告

题目：TLS VPN 设计与实现

姓 名 _____

班 级 _____

学 号 _____

联系方式 _____

分 数 _____

评 分 人 _____

2025 年 1 月 9 日

报告要求

1. 报告不可以抄袭，发现雷同者记为 0 分。
2. 报告中不可以只粘贴大段代码，应是文字与图、表结合的，需要说明流程的时候，也应该用流程图或者伪代码来说明；如果发现有大量代码粘贴者，报告需重写。
3. 报告格式严格按照要求规范，并作为评分标准。

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
1.1 网络拓扑	1
1.2 通信机制	1
1.3 加密原理	2
1.4 认证机制	3
1.5 多用户并发	3
2 VPN 系统设计	5
2.1 概要设计	5
2.2 详细设计	7
3 VPN 实现细节	14
3.1 问题 1	14
3.2 问题 2	15
3.3 问题 3	16
4 测试结果与分析	17
4.1 认证 VPN 服务器	17
4.2 认证 VPN 客户端	17
4.3 加密隧道通信	19
4.4 支持多客户端	21
4.5 易用性和稳定性	22
5 体会与建议	23
5.1 心得体会	23
5.2 意见建议	24

1 实验原理

本次实验旨在借助 openssl 库搭建一套简易的 VPN 系统。在实现过程中，运用 TUN/TAP 设备构建虚拟网络接口，进而打造出 IP 隧道，用于封装数据包并实现数据传输。

为切实保障数据传输的安全性，选用 TLS 协议对隧道内的数据进行加密处理。同时，引入消息验证代码（MAC），以此确保数据的完整性，有效防范数据遭到篡改以及重放攻击。此外，本实验还实现了服务器与客户端的认证机制，只有经过授权的用户才具备访问专用网络的权限。通过对服务器和客户端证书的配置，利用公钥证书对 VPN 服务器的身份予以验证，避免出现假冒服务器的情况。

在本次实验里，核心聚焦于基于 TLS/SSL 的 VPN，即 TLS/SSL VPN。这是一种借助传输层安全性协议（TLS）或安全套接字层（SSL）构建而成的虚拟专用网络（VPN）。其工作原理是在客户端与服务器之间搭建起加密的通信隧道，从而保证数据传输的安全性与隐私性。

相较于传统基于 IPSec 的 VPN，TLS/SSL VPN 具备显著优势。它无需复杂的网络配置，能够直接穿透防火墙，原因在于其使用的端口（一般为 HTTP/HTTPS 端口）通常不会被封堵。TLS/SSL VPN 让远程用户可以安全地访问内部网络资源，如同直接连接到内部网络一般。同时，它支持身份认证、数据加密以及完整性校验等功能，非常适用于对安全性和灵活性要求较高的远程访问场景。

1.1 网络拓扑

外网主机 HostU、HostU2 上运行 VPN 客户端，其 IP 地址为 10.0.2.7。VPN 网关上运行 VPN 服务器端，其在外网的 IP 地址为 10.0.2.8，在内网的 IP 地址为 192.168.60.1，内网主机 HostV 的 IP 地址为 192.168.60.101。客户端软件向服务器发起连接请求，在经过双方互相认证成功后即可建立连接，此时外网主机 HostU、HostU2 可以访问内网主机 HostV 的 telnet 服务。

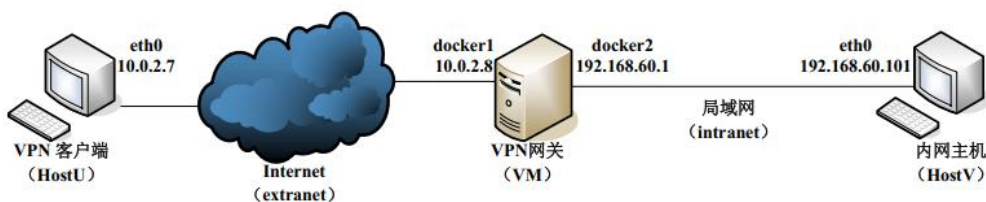


图 1-1 实验的网络拓扑

1.2 通信机制

在 VPN 的通信机制中，当客户端意图与服务器建立连接时，首先会发送一个请求以构建 SSL 安全隧道。在此过程中，客户端需通过输入用户名与密码执行身份验证操作，同时运用 CA 证书对服务器的证书展开验证。而服务器端则借助 shadow 文件中存储的密码散列值，来校验客户端提供的凭证。一旦双方身份验证均通过，SSL 隧道连接即可成功建立。

连接建立完成后，服务器会为虚拟网卡分配 IP 地址，并自动生成相应的路由规则，随后将分配的 IP 地址信息告知客户端。客户端接收到该信息后，会创建自身的虚拟网卡，并依据从服务器获取的 IP 地址进行配置。至此，客户端与服务器之间成功构建起虚拟网络连接。

数据传输过程中，采用 TLS 协议进行封装与加密。当外网主机需向内网主机发送数据时，数据先被发送至虚拟网卡，再经由 SSL 隧道安全传输至服务器。服务器对数据进行解密与解封装操作后，依据目的 IP 地址将数据转发至内网主机。内网主机的回应数据同样遵循这一传输流程。特别指出的是，当通信双方中任意一方接收到长度为 0 的报文时，该报文将被视作通话结束的信号，随后连接会被释放。

整体而言，这种通信机制不仅通过 SSL 协议实现了数据的加密与完整性保护，有力保障了数据传输的安全性，还借助虚拟网卡与路由规则，赋予了 VPN 客户端访问内网资源的能力。该设计使得 VPN 技术成为在不安全网络环境中，安全扩展专用网络的一种行之有效的手段。

1.3 加密原理

加密原理的核心目标是保障数据传输的安全性与机密性，这一目标通过融合非对称加密与对称加密两种方式达成。在实验场景中，客户端与服务器首先借助非对称加密算法，例如 RSA 算法，实现相互身份验证并安全地协商出一个会话密钥。在此过程里，RSA 算法的公钥与私钥发挥着关键作用。服务器的公钥被用于加密会话密钥，随后该加密后的会话密钥被安全传输至客户端，由于只有服务器的私钥能够对其进行解密，这就确保了密钥交换过程的安全性。

一旦会话密钥被安全协商完毕，客户端与服务器便会采用对称加密算法，诸如 DES 或 RC4，对传输的数据进行加密与解密操作。对称加密算法的特点是在数据加密和解密过程中使用相同的密钥，相较于非对称加密算法，其在执行效率上更高，因而适用于大量数据的加密处理。通过这种方式，VPN 隧道内的数据传输得以实现高效的安全防护。

SSL 协议不仅对密钥交换与数据加密的流程予以明确界定，还保障了数据传输的完整性与可靠性。基于 SSL 协议，每个通信节点均使用唯一的会话密钥。如此一来，即便多个客户端同时连接至同一 VPN 服务器，数据的安全性依然能够得到维持，原因在于每个通信会话均是独立加密的。



图 1-2 加密原理

1.4 认证机制

认证机制作为确保数据传输安全性的核心环节，涵盖服务器认证与客户端认证两大关键部分。

服务器认证旨在助力客户端精准确认服务器身份，有效规避连接至假冒服务器的风险。此过程一般借助公钥证书得以实现。首先，创建一个证书颁发机构（CA），并为其生成自签名证书。由于该证书为自签名形式，故被视作可信的根证书。随后，运用此 CA 为 VPN 服务器生成一对 RSA 公私钥，并向 CA 请求为服务器的公钥签发证书。在实验操作中，利用 OpenSSL 库依次完成生成密钥、创建证书签名请求（CSR），以及由 CA 对这些请求进行签名，最终生成服务器证书等一系列步骤。

客户端认证主要聚焦于核验试图连接 VPN 的用户是否具备相应权限。在实验场景下，客户端在与服务器建立连接时，需提供用户名和密码。这些凭据通过已搭建的 VPN 隧道安全传输至服务器。服务器随即调用 login 函数，通过查阅影子文件（/etc/shadow）中的记录，对用户名和密码的准确性展开验证。若服务器在影子文件中成功匹配到用户名与密码的散列值，则认证通过，客户端被赋予访问 VPN 的权限；若未找到匹配项，连接将被即刻断开。

1.5 多用户并发

在实际的 VPN 网络部署架构中，VPN 服务器需要具备强大的并发处理能力，以应对多个客户端同时发起的连接请求。每一个客户端连接到服务器后，都会被分配一条独立的 VPN 隧道，并在该隧道上建立一个唯一的 TLS 会话，以此确保数据传输的安全性与独立性。

VPN 服务器的核心任务是高效、准确地处理来自不同客户端的数据包，确保这些数据包能够无误地抵达目标客户端。这一过程涉及到复杂的数据包路由与转发机制。

具体而言，VPN 服务器的主进程在每个 VPN 隧道建立之初，会为其创建一个独立的子进程。这些子进程与主进程协同工作，共同完成数据包的处理任务。当数据包通过 VPN 隧道传输至服务器时，相应的子进程会迅速捕获数据包，并通过 TUN 网络接口将其转发至专用网络。

反之，当数据包从专用网络到达 TUN 接口时，主进程将负责接收这些数据包。主进程会依据数据包的目标地址或其他相关标识，快速判断该数据包应被转发至哪个 VPN 隧道。一旦确定目标隧道，主进程便会精准地将数据包路由至对应的子进程，由子进程完成后续的数据包发送操作。

为了进一步优化服务器对不同客户端消息的接收与定向转发效率，系统引入了管道通信技

术。每个子进程都会持续监控其对应的管道接口，一旦有数据到达，便能立即读取。在完成数据处理后，子进程会通过管道接口将响应消息发送回对应的客户端，实现数据的高效交互。

具体过程如下图所示：

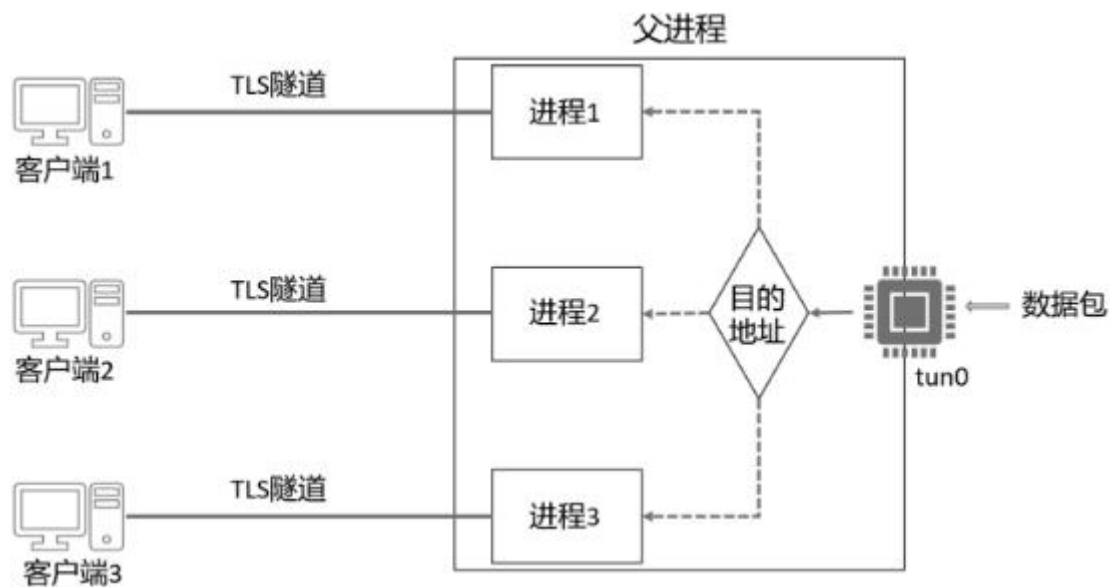


图 1-3 多用户并发

2 VPN 系统设计

2.1 概要设计

2.1.1 系统架构

系统由服务器端和客户端组成，采用客户端-服务器（C/S）架构。服务器端负责处理来自客户端的连接请求，进行用户身份验证，建立隧道并转发数据；客户端负责与服务器建立连接，发送用户身份验证信息，创建隧道并接收数据。

2.1.2 模块划分

服务器端

TLS/SSL 模块：负责服务器端的 TLS/SSL 协议初始化和配置，包括证书和私钥的加载、SSL 上下文的创建和设置等。通过 `setupTLSServer()` 函数实现。

TCP 模块：负责创建 TCP 服务器套接字，绑定端口并监听客户端连接请求。通过 `setupTCPServer()` 函数实现。

TUN 设备模块：负责创建 TUN 设备，用于虚拟网络接口的创建和数据的读取。通过 `createTunDevice()` 函数实现。

管道文件模块：用于在服务器端创建管道文件，以便与客户端进行数据通信。通过 `mkfifo()` 函数创建管道文件，并在 `listen_pipe()` 线程中读取管道文件数据。

用户身份验证模块：负责验证客户端发送的用户名和密码是否正确。通过 `login()` 函数实现，使用系统密码文件进行验证。

数据转发模块：负责将从客户端接收到的数据写入 TUN 设备，以及将从 TUN 设备读取的数据发送给客户端。通过 `sendOut()` 函数和 `listen_tun()` 线程实现。

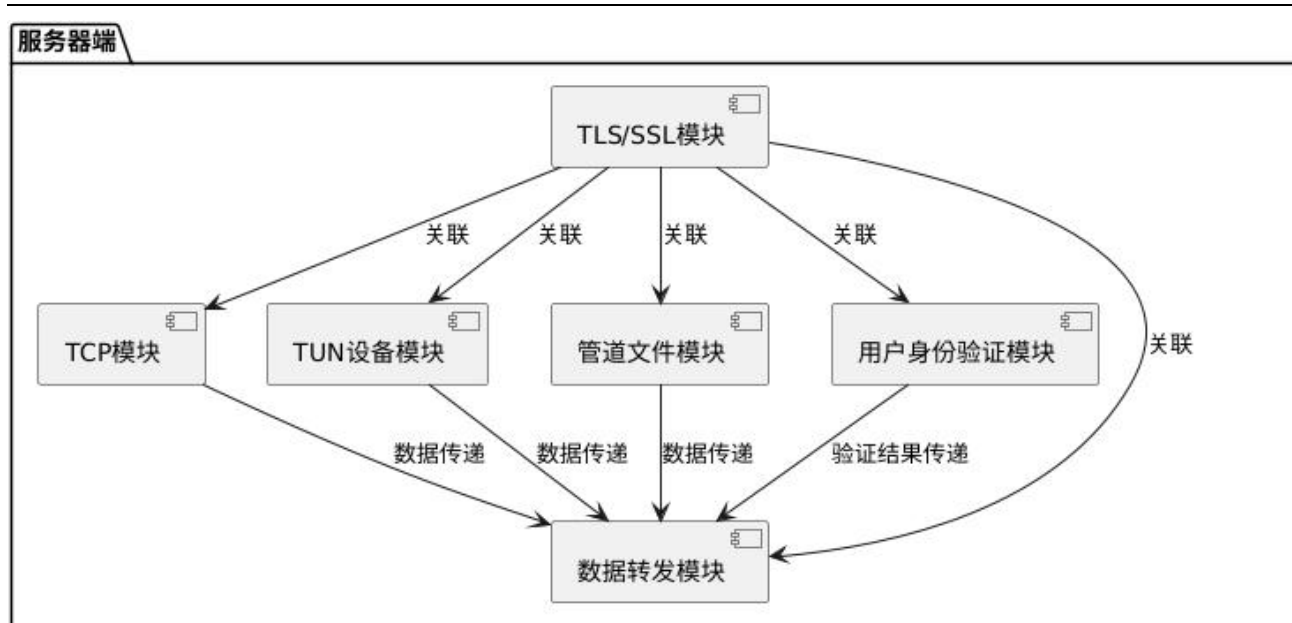


图 2-1 服务器端架构

客户端

TLS/SSL 模块：负责客户端的 TLS/SSL 协议初始化和配置，包括证书验证、SSL 上下文的创建和设置等。通过 `setupTLSClient()` 函数实现。

TCP 模块：负责创建 TCP 客户端套接字，连接到服务器。通过 `setupTCPClient()` 函数实现。

TUN 设备模块：负责创建 TUN 设备，用于虚拟网络接口的创建和数据的读取。通过 `createTunDevice()` 函数实现。

数据转发模块：负责将从 TUN 设备读取的数据发送给服务器，以及将从服务器接收到的数据写入 TUN 设备。通过 `listen_tun()` 线程和主函数中的 `SSL_read()` 和 `write()` 实现。

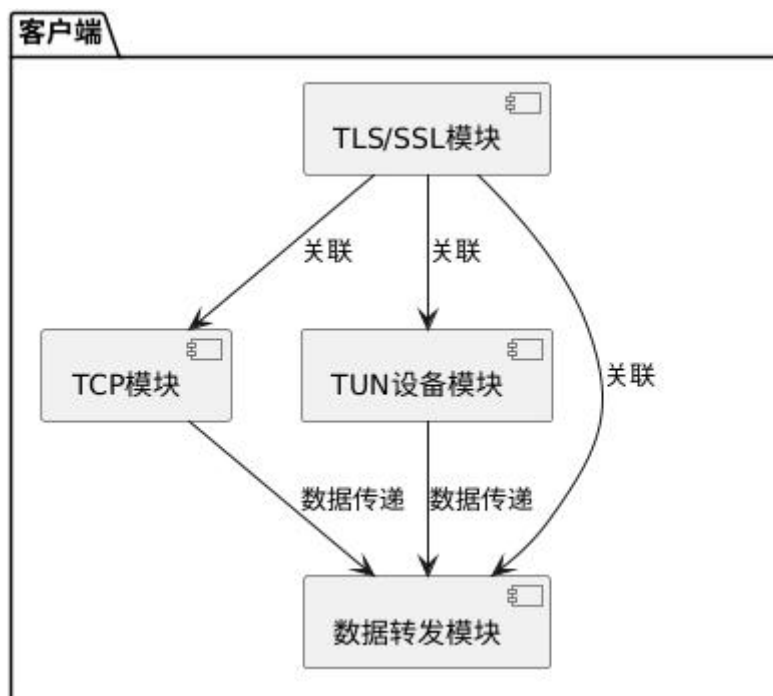


图 2-2 客户端架构

2.1.3 工作机制

服务器端

服务器端首先初始化 TLS/SSL 协议，加载证书和私钥，创建 SSL 上下文以支持安全通信。接着，它创建一个 TCP 服务器套接字，绑定到指定端口并开始监听客户端的连接请求。当客户端连接时，服务器端进行用户身份验证，验证通过后，创建一个与客户端 IP 对应的管道文件用于数据传输。服务器端启动一个线程持续监听 TUN 设备的数据，将接收到的数据根据目的 IP 地址写入相应的管道文件。同时，服务器端从客户端接收数据并将其转发到 TUN 设备，实现数据的双向传输。

客户端

客户端首先初始化 TLS/SSL 协议，加载证书并设置证书验证回调函数，以确保与服务器的安全通信。然后，客户端创建一个 TCP 套接字并连接到服务器。连接成功后，客户端创建一个 TUN 设备并配置虚拟网络接口。客户端启动一个线程监听 TUN 设备的数据，将接收到的数据发送给服务器。同时，客户端向服务器发送用户身份验证信息，包括用户名、密码和本地 IP 地址的最后一个字节。客户端从服务器接收数据并将其写入 TUN 设备，实现数据的双向传输。

2.2 详细设计

2.2.1 服务器端详细设计

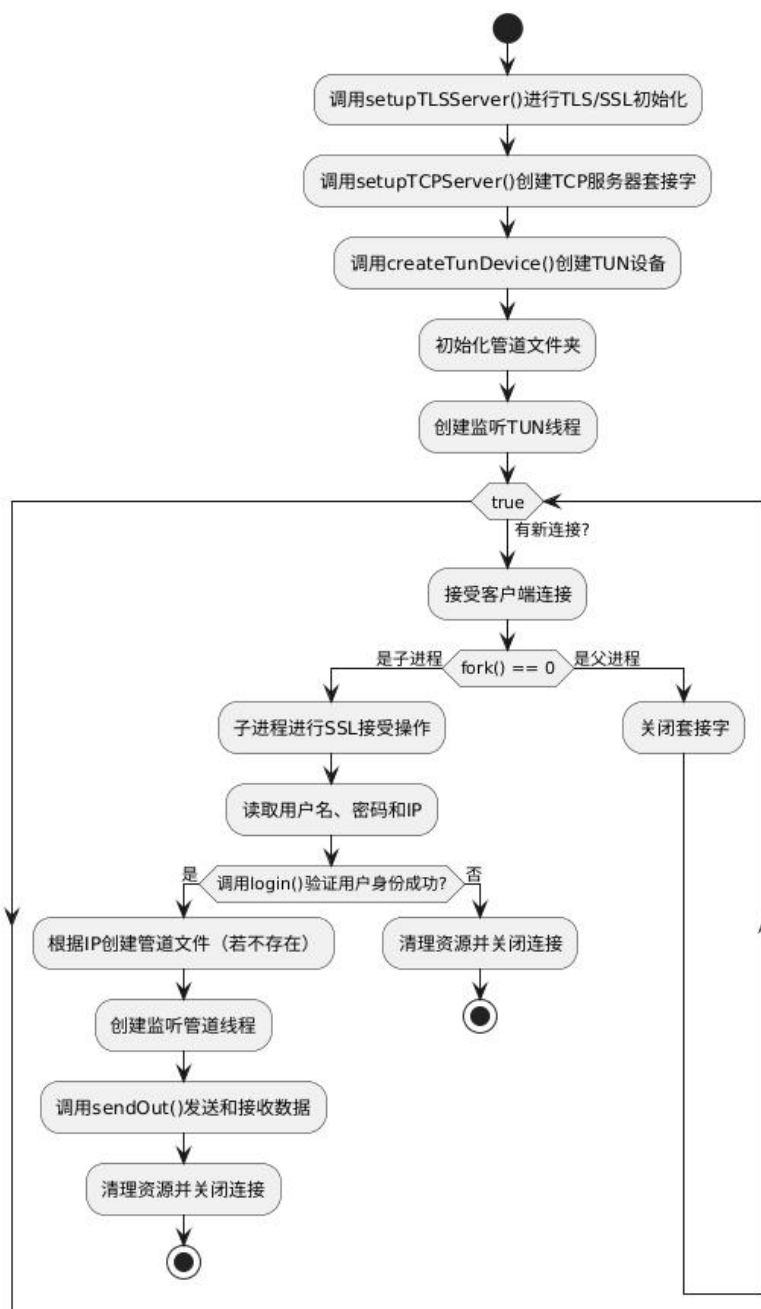


图 2-1 服务器端流程图

服务器端首先初始化 TLS/SSL 协议，加载证书和私钥，然后创建 TCP 服务器套接字并监听客户端连接请求。当客户端连接时，服务器创建一个子进程进行处理，完成 SSL 握手建立安全连接，并接收用户身份验证信息。验证通过后，服务器根据客户端 IP 创建管道文件，启动线程监听管道文件数据并发送给客户端，同时从客户端接收数据写入 TUN 设备，实现数据的双向传输。处理完毕后，关闭连接并继续监听新的客户端请求。下面分函数依次介绍其流程、

功能、数据结构等。

(1) setupTLSServer()

流程：

1. 初始化 OpenSSL 库；
2. 加载错误字符串和 SSL 算法；
3. 创建 SSL 上下文对象，设置为 TLSv1.2 方法；
4. 设置 SSL 上下文的证书验证方式为不验证；
5. 加载服务器端的证书文件和私钥文件；
6. 创建并返回一个 SSL 对象。

主要功能：初始化 TLS/SSL 协议，加载证书和私钥，创建 SSL 上下文和 SSL 对象，为后续的 SSL 通信做准备。

(2) setupTCPServer()

流程：

1. 创建一个 TCP 服务器套接字；
2. 绑定套接字到本地所有可用接口的 4433 端口；
3. 开始监听客户端的连接请求，最多允许 5 个连接排队等待。

主要功能：创建 TCP 服务器套接字并监听客户端连接请求。

数据结构：

- struct sockaddr_in sa_server: 服务器地址结构体。

(3) createTunDevice()

流程：

1. 打开 /dev/net/tun 设备文件，创建一个 TUN 设备；
2. 设置 TUN 设备的标志为 IFF_TUN (TUN 设备) 和 IFF_NO_PI (不包含协议信息)。

主要功能：创建一个 TUN 设备，用于虚拟网络接口的创建和数据的读取。

数据结构：

- struct ifreq ifr: 网络接口请求结构体。

(4) listen_tun()

流程：

1. 循环读取 TUN 设备的数据；

2. 当接收到符合条件的数据时（数据长度大于 19 且第一个字节为 0x45），根据数据中的目的 IP 地址，构建管道文件路径；

3. 尝试打开管道文件，如果成功，则将数据写入管道文件。

主要功能：监听 TUN 设备的数据，将接收到的数据写入对应的管道文件。

(5) login()

流程：

1. 根据用户名获取对应的密码信息；

2. 使用 `crypt()` 函数对输入的密码进行加密，并与存储的密码进行比较；

3. 如果密码匹配，返回 1 表示验证成功；否则返回 0 表示验证失败。

主要功能：验证客户端发送的用户名和密码是否正确。

数据结构：

➤ `struct spwd *pw`：密码结构体。

算法：

crypt 加密：`crypt()` 函数通过将用户输入的明文密码和一个随机生成的盐值结合，使用特定的加密算法（如 SHA-512）进行加密，生成加密后的密码。验证时，使用相同的盐值对输入密码重新加密，然后将结果与存储的加密密码进行比较，如果相同则验证通过。另外，它使用一种称为“影子密码”（shadow password）的机制，将加密后的密码存储在 `/etc/shadow` 文件中，而不是 `/etc/passwd` 文件中，以提高安全性。

(6) listen_pipe()

流程：

1. 打开管道文件，读取其中的数据；

2. 将读取到的数据通过 SSL 对象发送给客户端；

3. 当管道文件读取完毕（读取到 0 字节）时，关闭管道文件并删除该文件。

主要功能：监听的管道文件数据，将数据发送给客户端。

数据结构：

➤ `PIPEDATA *ptd`：管道数据结构体，包含管道文件路径和 SSL 对象。

(7) sendOut()

流程:

1. 从 SSL 对象接收数据;
2. 将接收到的数据写入 TUN 设备;
3. 循环执行上述操作, 直到没有数据可读取;
4. 最后关闭 SSL 连接并输出提示信息。

主要功能: 从客户端接收数据并写入 TUN 设备, 实现数据的转发。

2.2.2 客户端详细设计

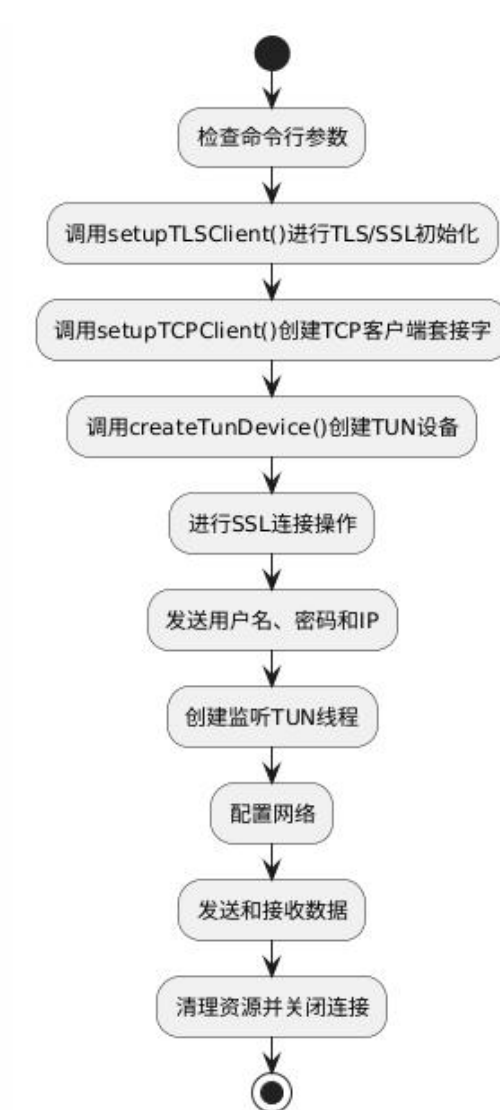


图 2-2 客户端流程图

客户端首先检查命令行参数, 然后初始化 TLS/SSL 协议并连接到服务器。完成 SSL 握手后, 它发送用户身份验证信息。接着, 创建 TUN 设备并配置网络接口, 同时启动一个线程监听 TUN 设备的数据。主函数还负责从服务器接收数据并将其写入 TUN 设备, 实现数据的双向

传输。处理完成后，取消监听线程并关闭连接。下面分函数依次介绍其流程、功能、数据结构等。

(1) verify_callback(int preverify_ok, X509_STORE_CTX *x509_ctx)

流程：

1. 获取当前证书的主体名称；
2. 打印证书的主体名称；
3. 如果验证通过，打印验证通过的信息；
4. 如果验证失败，获取错误代码并打印错误信息；
5. 返回验证结果。

主要功能：证书验证回调函数，用于在 SSL 握手过程中验证服务器证书。

(2) setupTLSClient(const char *hostname)

流程：

1. 初始化 OpenSSL 库；
2. 加载字符串和 SSL 算法；
3. 创建 SSL 上下文对象，设置为 TLSv1.2 方法；
4. 设置 SSL 上下文的证书验证方式为验证对等证书，并设置验证回调函数；
5. 加载 CA 证书路径；
6. 创建 SSL 对象；
7. 设置 SSL 对象的主机名验证参数。

主要功能：初始化 TLS/SSL 协议，加载 CA 证书，创建 SSL 上下文和 SSL 对象，为后续的 SSL 通信做准备。

(3) setupTCPClient(const char *hostname, int port)

流程：

1. 获取主机名对应的 IP 地址；
2. 创建一个 TCP 客户端套接字；
3. 构建服务器地址结构体；
4. 连接到服务器。

主要功能：创建 TCP 客户端套接字并连接到服务器。

数据结构：

- struct sockaddr_in server_addr: 服务器地址结构体。
- struct hostent *hp: 主机信息结构体。

(4) createTunDevice()

流程：

1. 打开 /dev/net/tun 设备文件，创建一个 TUN 设备；
2. 设置 TUN 设备的标志为 IFF_TUN (TUN 设备) 和 IFF_NO_PI (不包含协议信息)。

主要功能： 创建一个 TUN 设备，用于虚拟网络接口的创建和数据的读取。

数据结构：

- struct ifreq ifr: 网络接口请求结构体。

(5) listen_tun(void *threadData)

流程：

1. 从线程数据中获取 TUN 设备的文件描述符和 SSL 对象；
2. 循环读取 TUN 设备的数据；
3. 当接收到符合条件的数据时（数据长度大于 19 且第一个字节为 0x45），检查数据中的目的 IP 地址是否匹配；
4. 如果匹配，将数据通过 SSL 对象发送给服务器；
5. 如果不匹配，打印错误信息。

主要功能： 监听 TUN 设备的数据，将接收到的数据发送给服务器。

数据结构：

- PTHDATA ptd: 线程数据结构体，包含 TUN 设备的文件描述符和 SSL 对象。

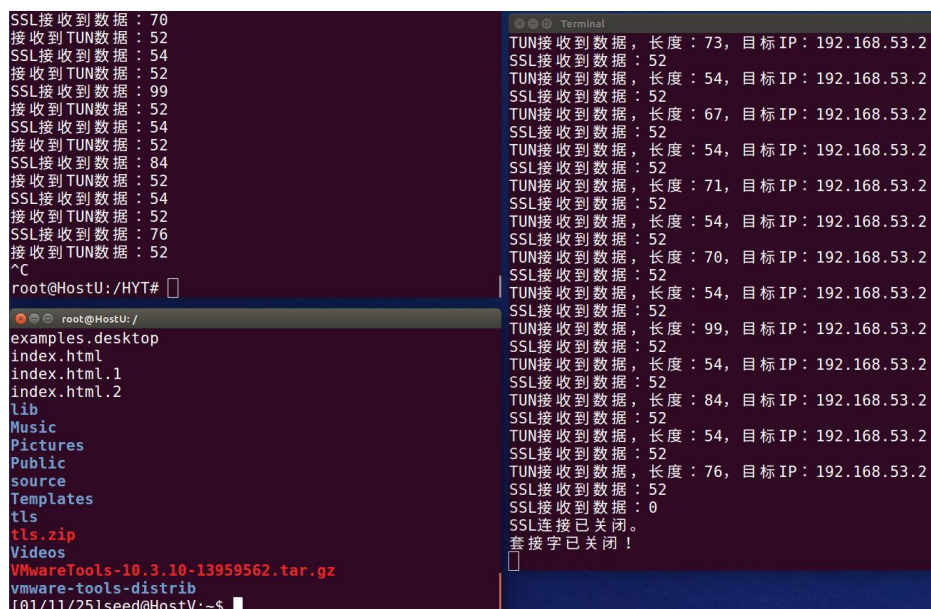
3 VPN 实现细节

3.1 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。

(1) 观察到的内容

断开 VPN 隧道后，在 telnet 窗口输入内容，窗口没有反应，如图 3-1 所示。



```
SSL接收到数据: 70
接收到TUN数据: 52
SSL接收到数据: 54
接收到TUN数据: 52
SSL接收到数据: 99
接收到TUN数据: 52
SSL接收到数据: 54
接收到TUN数据: 52
SSL接收到数据: 84
接收到TUN数据: 52
SSL接收到数据: 54
接收到TUN数据: 52
SSL接收到数据: 76
接收到TUN数据: 52
^C
root@HostU: /HYT#

root@HostU: /
examples.desktop
index.html
index.html.1
index.html.2
lib
Music
Pictures
Public
source
Templates
tls
tls.zip
Videos
VMwareTools-10.3.10-13959562.tar.gz
vmware-tools-distrib
[01/11/25]seed@HostV: ~$

TUN接收到数据, 长度: 73, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 67, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 71, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 70, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 99, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 84, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 76, 目标IP: 192.168.53.2
SSL接收到数据: 52
SSL接收到数据: 0
SSL连接已关闭。
套接字已关闭!
```

图 3-1 断连后输入指令

这是因为断开 VPN 后，tun 设备不存在了，HostU 通过 tun 路由到 HostV 的报文无法发送。

(2) 为什么重新连接需要将 vpnserver 和 vpnclient 都退出后再重复操作

同时杀死服务器端和客户端可以确保双方的状态完全重置。这样可以避免任何潜在的状态不一致问题，例如积压的报文、半开放的连接等。重新启动双方可以确保从一个干净的状态开始，避免任何历史状态对新连接的影响。

这种做法可以确保连接的稳定性和可靠性，特别是在实验环境中，为了确保每次实验的条件一致，可以采取这种彻底重置的方法。

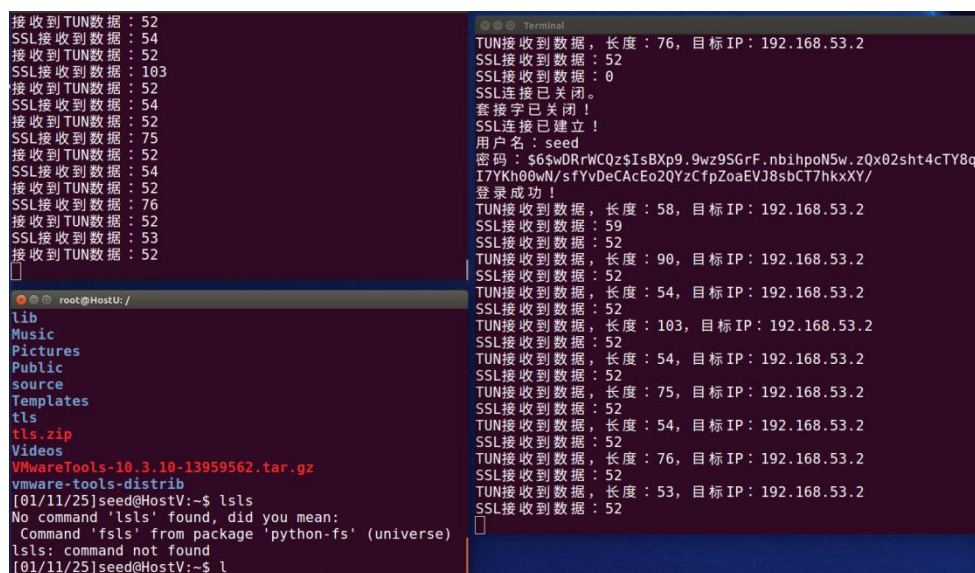
而若是仅断开客户端，服务端由于仍处于数据转发的循环中，将持续监听原连接端口，这

会导致它无法响应新的客户端连接请求，从而阻止新客户端建立连接。

若是仅断开服务端，客户端在服务端重启时仍处于原有的隧道处理循环中，由于连接的发送端口保持不变，客户端将无法发出新的连接请求，导致无法重新建立连接。

(3) 正确重连后，telnet 连接会发生什么？

首先，窗口中会弹出许多信息，如图 3-2 所示。



```
接收到的TUN数据: 52
SSL接收到数据: 54
接收到的TUN数据: 52
SSL接收到数据: 103
接收到的TUN数据: 52
SSL接收到数据: 54
接收到的TUN数据: 52
SSL接收到数据: 75
接收到的TUN数据: 52
接收到的TUN数据: 76
接收到的TUN数据: 52
SSL接收到数据: 53
接收到的TUN数据: 52

root@HostU:/
lib
Music
Pictures
Public
source
Templates
tls
tls.zip
Videos
VMwareTools-10.3.10-13959562.tar.gz
vmware-tools-distrib
[01/11/25]seed@HostV:~$ ls
No command 'ls' found, did you mean:
Command 'fs' from package 'python-fs' (universe)
ls: command not found
[01/11/25]seed@HostV:~$ l

TUN接收到数据, 长度: 76, 目标IP: 192.168.53.2
SSL接收到数据: 52
SSL接收到数据: 0
SSL连接已关闭。
套接字已关闭！
SSL连接已建立！
用户名: seed
密码: $6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.z0x02sht4cTY8q
I7YKh00wN/sfYvDeCAcEo2QYzCfpZoaEVJ8sbCT7hkxXY/
登录成功！
TUN接收到数据, 长度: 58, 目标IP: 192.168.53.2
SSL接收到数据: 59
SSL接收到数据: 52
TUN接收到数据, 长度: 90, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 103, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 75, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 54, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 76, 目标IP: 192.168.53.2
SSL接收到数据: 52
TUN接收到数据, 长度: 53, 目标IP: 192.168.53.2
SSL接收到数据: 52
```

图 3-2 重连后的窗口

这是因为断开 VPN 后，键盘输入的内容依然会作为 TCP 报文发出，只是无法到达 HostV。但是，TCP 会进行报文重传，当 VPN 重新连接后，丢失的 TCP 报文通过重传就可以重新发送到 HostV，因此会弹出许多回显信息。

然后再输入指令，发现 telnet 恢复连接了。

3.2 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？

在 SSL VPN 登录协议中，客户端首先通过 TCP 连接到服务器的 4433 端口，并完成 SSL 握手以建立安全连接。接着，客户端依次发送三个报文：用户名、密码和本地 TUN 设备 IP 地址的最后一个字节。这些报文都是明文字符串，用于服务器端的身份验证和配置。服务器端读取这些报文后，使用 crypt 函数对密码进行加密验证。如果验证通过，服务器端会根据客户端

发送的 IP 地址的最后一个字节创建管道文件，并启动一个线程监听该管道文件，以便将数据发送回客户端。如果验证失败，服务器端将关闭连接。整个过程中，报文的发送和接收顺序是固定的，客户端先发送用户名，然后是密码，最后是 IP 地址的最后一个字节，服务器端依次读取并处理这些报文，以完成登录过程并准备后续的数据传输。

3.3 问题 3

你的 VPN 服务器支持多客户端采用了什么技术？并说明如何实现。

采用了多进程、多线程以及命名管道技术。

每有一个新的客户端连接，服务器端使用 fork 系统调用创建子进程来处理连接。子进程继承了父进程的文件描述符，包括监听套接字和新接受的客户端套接字；子进程关闭监听套接字，因为子进程只需要处理当前的客户端连接；处理完毕后，子进程关闭 SSL 连接和客户端套接字，并退出。

每个子进程在处理客户端连接时，创建一个线程来监听 TUN 设备的数据。listen_tun 函数在一个循环中读取 TUN 设备的数据，并根据数据中的目的 IP 地址将数据写入对应的管道文件。如果验证通过，子进程还会创建另一个线程来监听管道文件的数据，并将数据发送给客户端。listen_pipe 函数在一个循环中读取管道文件的数据，并通过 SSL 连接发送给客户端。当管道文件读取完毕（读取到 0 字节）时，线程关闭管道文件并删除该文件。

每个子进程代表一个客户端的 SSL 连接，并拥有一个与之对应的、唯一命名的管道文件。当主进程在 tun0 接口接收到来自内核的 IP 数据包时，它根据数据包的目的 IP 地址，将 IP 包转发至相应的管道文件，实现进程间的通信。

4 测试结果与分析

4.1 认证 VPN 服务器

首先在 HostU 查看时间信息，如图 4-1 所示。

```
root@HostU:/HYT# date
Sat Jan 11 19:49:37 CST 2025
root@HostU:/HYT#
```

图 4-1 查看时间

然后修改时间信息，使证书过期，如图 4-2 所示。

```
root@HostU:/HYT# date -s 2034/11/23
Thu Nov 23 00:00:00 CST 2034
root@HostU:/HYT#
```

图 4-2 修改时间

在主机启动服务器，然后在 HostU 启动客户端，并输入正确的账户密码，发现登录失败，报错信息为证书到期，如图 4-3 所示。

```
root@HostU:/HYT# sudo ./myclient hyt 4433 seed dees
2
TLS客户端设置成功！
TCP客户端设置成功！
证书主题：/C=CN/ST=hubei/L=wuhan/O=hust/OU=cse/CN=HY
T/emailAddress=u202211915@hust.edu.cn
错误：验证失败：certificate has expired.
3073305088:error:14090086:SSL routines:ssl3_get_serv
er_certificate:certificate verify failed:s3_clnt.c:1
264:
root@HostU:/HYT#
```

图 4-3 证书过期

最后将时间修改回来，便于后续实验。

4.2 认证 VPN 客户端

首先在主机上启动 VPN 服务器，然后在 HostU 上启动客户端，并输入正确的账户密码，

成功登录，结果如图 4-4、图 4-5 所示。

```
SSL连接已建立！  
用户名：seed  
密码：$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8q  
I7YKh00wN/sfYvDeCAcEo2QYzCfpZoaEVJ8sbCT7hkxXY/  
登录成功！  
█
```

图 4-4 服务器登录成功信息

```
root@HostU:/HYT# sudo ./myclient hyt 4433 seed dees  
2  
TLS客户端设置成功！  
TCP客户端设置成功！  
证书主题：/C=CN/ST=hubei/L=wuhan/O=hust/OU=cse/CN=HY  
T/emailAddress=u202211915@hust.edu.cn  
验证通过。  
证书主题：/C=CN/ST=hubei/O=hust/OU=cse/CN=HYT/emailA  
ddress=u202211915@hust.edu.cn  
验证通过。  
SSL连接成功！  
SSL连接使用：AES256-GCM-SHA384
```

图 4-5 客户端登录成功信息

客户端退出登录后，重新登录，输入错误的用户名，发现无法登录，如图 4-6 所示。

```
root@HostU:/HYT# sudo ./myclient hyt 4433 seem dees  
2  
TLS客户端设置成功！  
TCP客户端设置成功！  
证书主题：/C=CN/ST=hubei/L=wuhan/O=hust/OU=cse/CN=HY  
T/emailAddress=u202211915@hust.edu.cn  
验证通过。  
证书主题：/C=CN/ST=hubei/O=hust/OU=cse/CN=HYT/emailA  
ddress=u202211915@hust.edu.cn  
验证通过。  
SSL连接成功！  
SSL连接使用：AES256-GCM-SHA384  
连接已关闭！  
root@HostU:/HYT# █
```

图 4-6 用户名错误

再次登录，输入错误的密码，登录失败，客户端和服务器信息分别如图 4-7、图 4-8 所示。

```
root@HostU:/HYT# sudo ./myclient hyt 4433 seed deem
2
TLS客户端设置成功！
TCP客户端设置成功！
证书主题：/C=CN/ST=hubei/L=wuhan/O=hust/OU=cse/CN=HYT/emailAddress=u202211915@hust.edu.cn
验证通过。
证书主题：/C=CN/ST=hubei/O=hust/OU=cse/CN=HYT/emailAddress=u202211915@hust.edu.cn
验证通过。
SSL连接成功！
SSL连接使用：AES256-GCM-SHA384
连接已关闭！
root@HostU:/HYT#
```

图 4-7 客户端登录失败

```
SSL连接已建立！
用户名：seed
密码：$6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN/sfYvDeCAcEo2QYzCfpZoaEVJ8sbCT7hkxXY/
错误：密码错误！
错误：登录失败！
套接字已关闭！
```

图 4-8 服务器端报错信息

4.3 加密隧道通信

从 HostU ping 主机 192.168.60.101，成功 ping 通，如图 4-9 所示。

```
root@HostU:/# ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=0.621 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=0.473 ms
```

图 4-9 HostU ping HostV

同时，客户端与服务器端也出现了数据流信息，如图 4-10 所示。


```
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
SSL接收到数据 :84  
接收到TUN数据 :84  
  
e=0.268 ms  
64 bytes from 192.168.60.101: icmp_seq=25 ttl=63 tim  
e=0.469 ms  
64 bytes from 192.168.60.101: icmp_seq=26 ttl=63 tim  
e=0.256 ms  
64 bytes from 192.168.60.101: icmp_seq=27 ttl=63 tim  
e=0.270 ms  
64 bytes from 192.168.60.101: icmp_seq=28 ttl=63 tim  
e=0.274 ms  
64 bytes from 192.168.60.101: icmp_seq=29 ttl=63 tim  
e=0.272 ms  
64 bytes from 192.168.60.101: icmp_seq=30 ttl=63 tim  
e=0.274 ms  
64 bytes from 192.168.60.101: icmp_seq=31 ttl=63 tim  
e=0.640 ms
```

图 4-10 数据流信息

然后打开 wireshark，添加对 4433 端口的监听，如图 4-11 所示。

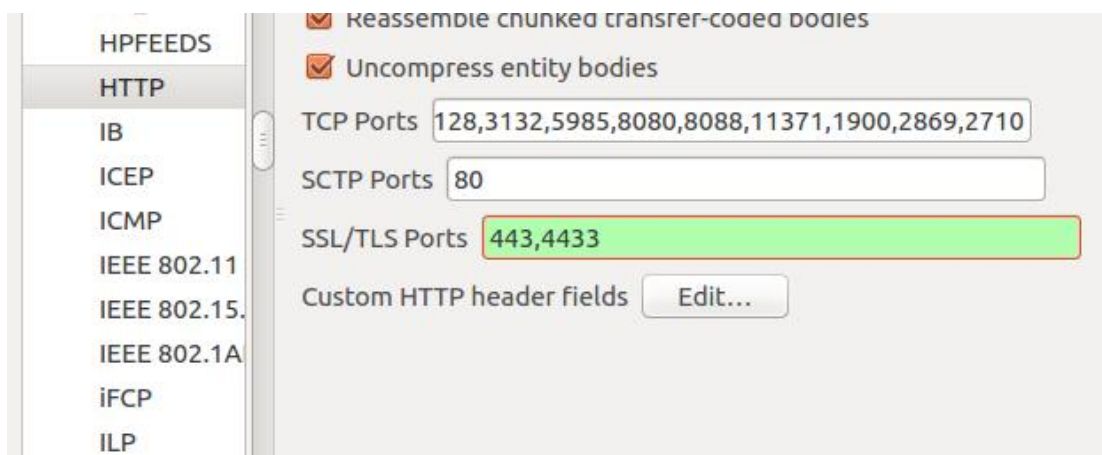


图 4-11 监听 4433 端口

监听 docker1，截获数据包如图 4-12 所示。

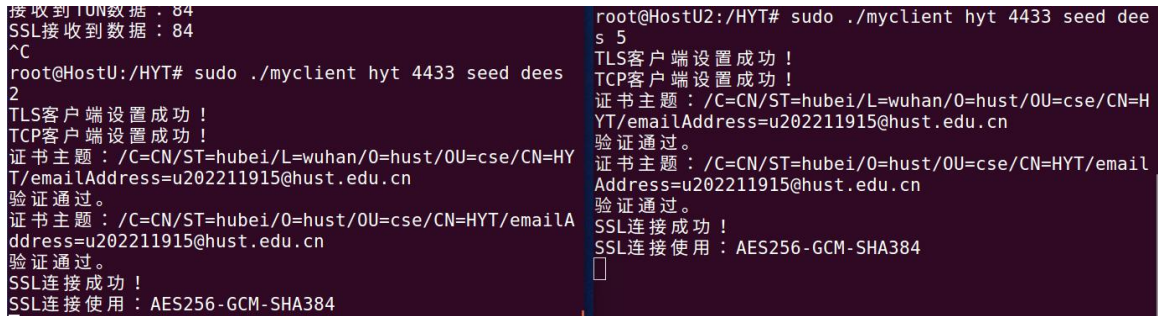
10.0.2.7	10.0.2.8	TCP	66 35536 → 4433 [ACK] Seq=2462593004 Ack=4016882436
10.0.2.7	10.0.2.8	TLSv1.2	179 Application Data
10.0.2.8	10.0.2.7	TLSv1.2	179 Application Data
10.0.2.7	10.0.2.8	TCP	66 35536 → 4433 [ACK] Seq=2462593117 Ack=4016882549
10.0.2.7	10.0.2.8	TLSv1.2	179 Application Data
10.0.2.8	10.0.2.7	TLSv1.2	179 Application Data
10.0.2.7	10.0.2.8	TCP	66 35536 → 4433 [ACK] Seq=2462593230 Ack=4016882662
10.0.2.7	10.0.2.8	TLSv1.2	179 Application Data
10.0.2.8	10.0.2.7	TLSv1.2	179 Application Data
10.0.2.7	10.0.2.8	TCP	66 35536 → 4433 [ACK] Seq=2462593343 Ack=4016882775
10.0.2.7	10.0.2.8	TLSv1.2	179 Application Data
10.0.2.8	10.0.2.7	TLSv1.2	179 Application Data
10.0.2.7	10.0.2.8	TCP	66 35536 → 4433 [ACK] Seq=2462593456 Ack=4016882888
10.0.2.7	10.0.2.8	TLSv1.2	179 Application Data
10.0.2.8	10.0.2.7	TLSv1.2	179 Application Data
10.0.2.7	10.0.2.8	TCP	66 35536 → 4433 [ACK] Seq=2462593569 Ack=4016883001
10.0.2.7	10.0.2.8	TLSv1.2	179 Application Data
10.0.2.8	10.0.2.7	TLSv1.2	179 Application Data

图 4-12 截获数据包

数据包的协议类型中包括了 TCP 与 TLSv1.2, 可以看出其经过了隧道封装且隧道为 TLS。

4.4 支持多客户端

使用两个客户端 HostU、HostU2 同时登录服务器，如图 4-13 所示。



```

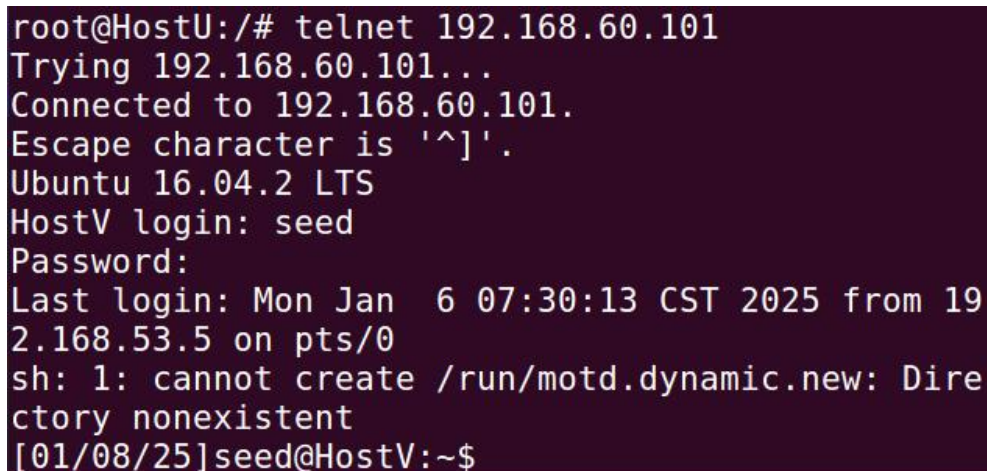
root@HostU:/HYT# sudo ./myclient hyt 4433 seed dees
2
接收收到TUN数据: 84
SSL接收到数据: 84
^C
TLS客户端设置成功!
TCP客户端设置成功!
证书主题: /C=CN/ST=hubei/L=wuhan/O=hust/OU=cse/CN=HYT/emailAddress=u202211915@hust.edu.cn
验证通过。
证书主题: /C=CN/ST=hubei/O=hust/OU=cse/CN=HYT/emailAddress=u202211915@hust.edu.cn
验证通过。
SSL连接成功!
SSL连接使用: AES256-GCM-SHA384

root@HostU2:/HYT# sudo ./myclient hyt 4433 seed dees
5
TLS客户端设置成功!
TCP客户端设置成功!
证书主题: /C=CN/ST=hubei/L=wuhan/O=hust/OU=cse/CN=HYT/emailAddress=u202211915@hust.edu.cn
验证通过。
证书主题: /C=CN/ST=hubei/O=hust/OU=cse/CN=HYT/emailAddress=u202211915@hust.edu.cn
验证通过。
SSL连接成功!
SSL连接使用: AES256-GCM-SHA384

```

图 4-13 两客户端登录

然后开启 HostV 的 opensshd 服务，两个客户端同时测试 telnet 连接。可以发现，连接均成功，且通信正常，如图 4-14、图 4-15 所示。

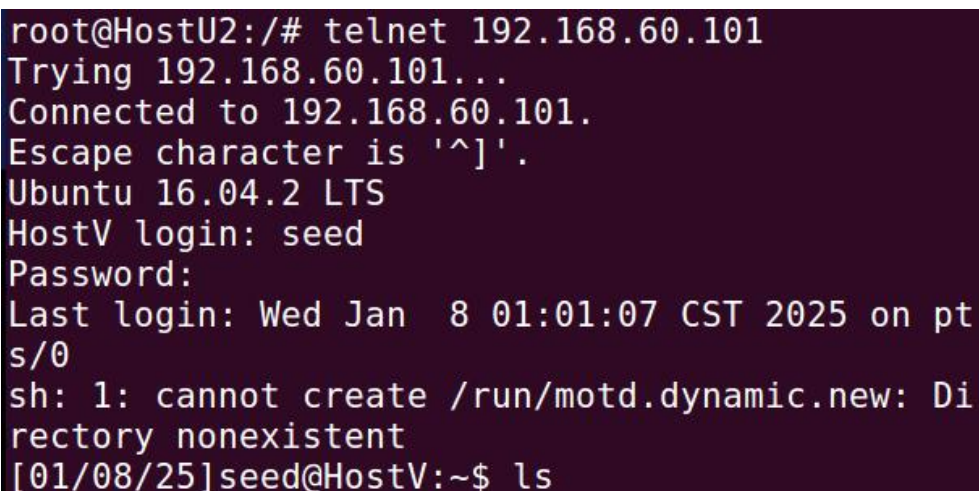


```

root@HostU:/# telnet 192.168.60.101
Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
HostV login: seed
Password:
Last login: Mon Jan  6 07:30:13 CST 2025 from 192.168.53.5 on pts/0
sh: 1: cannot create /run/motd.dynamic.new: Directory nonexistent
[01/08/25]seed@HostV:~$

```

图 4-14 HostU 通信正常



```

root@HostU2:/# telnet 192.168.60.101
Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
HostV login: seed
Password:
Last login: Wed Jan  8 01:01:07 CST 2025 on pts/0
sh: 1: cannot create /run/motd.dynamic.new: Directory nonexistent
[01/08/25]seed@HostV:~$ ls

```

图 4-15 HostU2 通信正常

接下来断开 HostU2 的 telnet 连接，如图 4-16 所示。

```

接收到TUN数据：52
接收到TUN数据：53
SSL接收到数据：53
接收到TUN数据：52
^C
root@HostU2:/HYT#

```

图 4-16 断开 HostU2 的连接

然后在 HostU 的连接中输入指令，依然有数据流反馈，说明在断开其中一个 telnet 连接后，另一连接仍能保持，如图 4-17 所示。

```

接收到TUN数据：52
接收到TUN数据：53
SSL接收到数据：53
接收到TUN数据：52
接收到TUN数据：54
SSL接收到数据：618
接收到TUN数据：52
SSL接收到数据：76
接收到TUN数据：52
接收到TUN数据：53
SSL接收到数据：53
接收到TUN数据：52
接收到TUN数据：53
SSL接收到数据：53
接收到TUN数据：52
SSL接收到数据：52
SSL接收到数据：0
SSL连接已关闭。
套接字已关闭！
SSL接收到数据：54
TUN接收到数据，长度：618，目标IP：192.168.53.2
SSL接收到数据：52
TUN接收到数据，长度：76，目标IP：192.168.53.2
SSL接收到数据：52
TUN接收到数据，长度：53，目标IP：192.168.53.2
SSL接收到数据：53
SSL接收到数据：52
SSL接收到数据：53
TUN接收到数据，长度：53，目标IP：192.168.53.2
SSL接收到数据：52

```

```

root@HostU:/
examples.desktop
index.html
index.html.1
index.html.2
lib
Music
Pictures
Public
source
Templates
tls
tls.zip
Videos
VMwareTools-10.3.10-13959562.tar.gz
vmware-tools-distrib
[01/11/25]seed@HostV:~$ ls

```

图 4-17 HostU 保持 telnet 通信

4.5 易用性和稳定性

VPN 客户端的虚拟 IP 获取采用了手动分配的方式。客户端的虚拟 IP 配置与内网路由配置均采用了程序自动添加的方式，代码如图 4-18 所示。

```

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 4-18 配置代码

程序在运行时非常稳定，多客户端运行时依然没有异常。

5 体会与建议

5.1 心得体会

通过本次计算机网络安全实验，我对 VPN 技术的复杂性和重要性有了深刻的认识。实验过程中，我不仅掌握了 VPN 的基本原理和实现方法，还对网络安全的关键技术有了更深入的理解。在实验开始前，我通过阅读实验指导手册，对 TLS/SSL VPN 的原理有了初步的了解。然而，真正动手实现时，才发现理论知识与实际操作之间存在很大的差距。例如，配置 TUN/TAP 设备、设置路由规则、生成证书等操作，都需要参照指导内容才能一步步完成。

在实验中，我实现了 TLS/SSL 协议对数据的加密处理，确保了数据传输的安全性。同时，通过公钥证书和影子文件对服务器和客户端进行身份认证，有效防止了假冒服务器和未经授权的用户访问。这让我深刻认识到，加密和认证是网络安全的两个重要支柱，缺一不可。只有通过严格的加密和认证机制，才能确保数据在不安全网络中的安全传输。在实验过程中，我还遇到了许多问题，如证书验证失败、数据传输不畅、连接断开等。通过使用 Wireshark 抓包、查看日志文件、调试代码等方法，我逐步解决了这些问题。这不仅提升了我的调试能力，还让我学会了如何在复杂系统中定位和解决问题。

为了支持多客户端连接，实验中采用了多进程和多线程技术与命名管道技术。每个客户端连接时，服务器端会创建一个子进程来处理该连接，子进程又会创建多个线程来处理 TUN 设备和管道文件的数据读取和发送。这种多进程与多线程的协同工作模式，不仅提高了系统的并发处理能力，还确保了数据传输的高效性和稳定性。通过这次实验，我对多进程和多线程以及命名管道的使用有了更深入的理解和实践经验。

虽然本次实验是个人完成的，但在实验过程中，我与同学进行了多次交流和讨论。通过交流，我不仅解决了许多技术难题，还从同学那里学到了许多新的思路和方法。这让我明白了在复杂的项目中，团队合作和交流的重要性。通过本次实验，我不仅掌握了 TLS/SSL VPN 的设计与实现方法，还对网络安全的关键技术有了更深入的理解。实验过程中的许多步骤让我受益匪浅，从理论学习到实际操作，从细节处理到问题解决，从多进程与多线程的协同工作到加密与认证的重要性，每一个环节都让我成长了许多。这次实验不仅提升了我的技术能力，还培养了我的耐心和细致，为我未来在网络安全领域的学习和工作打下了坚实的基础。

5.2 意见建议

建议添加一些理论学习的文件，如 TLS 协议的详细介绍，在实践的同时也提高理论基础。

原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

已阅读并同意以下内容。

判定为不合格的一些情形：

- (1) 请人代做或冒名顶替者；
- (2) 替人做且不听劝告者；
- (3) 实验报告内容抄袭或雷同者；
- (4) 实验报告内容与实际实验内容不一致者；
- (5) 实验代码抄袭者。

作者签名：