计算机网络专题训练

实验三：IPv4 over IPv6 隧道协议实验

实验报告

王奥丞（2014011367）

梁泽宇（2014011381）

1 实验目的

IPV4 over IPV6，简称“4over6”，是 IPV4 向 IPV6 发展进程中，向纯 IPV6 主干网过渡提出的一种新技术，可以最大程度地继承基于 IPV4 网络和应用，实现 IPV4 向 IPV6 平滑的过渡。

该实验通过实现 IPV4 over IPV6 隧道最小原型验证系统，让同学们对 4over6 隧道的实现原理有更加深刻的认识。

2 实验步骤

2.1 前端（Java 端）

前端负责所有 UI 相关的操作，包括 [TODO]。

2.2 后端（C 端）

后端负责处理所有通信方面的操作，包括连接建立、VPN 的启动、tun 和计时器的操作、消息收发、通信状态处理等。

2.3 前后端交互

C 和 Java 之间通过 JNI 和 Socket 通信，具体流程如下：

- 启动 VPN 时，Java 端创建一个线程，在其中调用阻塞 C 函数 startVpn，将控制权转移到 C；
- C 建立和 IPv6 服务器的连接，用 JNI 回调 Java 创建 tun 设备、计时器，再创建一个 Socket 绑定在 localhost，负责接受 Java 端的消息；
- 当 C 需要回调 Java 端来更新 ui 时，用 JNI 调用直接调用 Java 端，而不是使用 Socket 给 Java 端发送数据。这样的好处是 C 和 Java 之间的 Socket 就可以做成单向的，不需要考虑同步的问题，而且 C 直接用 JNI 调用 Java 实现起来简单，因为只是简单的函数调用；
- 当 Java 需要通知 C 停止连接时，用 Local Socket，C 接收到该消息后，退出循环，线程退出。

3 一些实现细节

- 由于 Android 的 UI 操作不是线程安全的，所以 Android 规定所有 UI 操作只能在主线程中进行。这里在主线程中使用 handler 接受其他线程发送的更新 UI 消息，并且更新 UI，其他线程不直接更新 UI，而是向该 handler 发送消息；
- C 这一部分使用了单线程 IO 多路复用，用 select 等待多个 fd，当有一个 fd 可写/可读时，select 返回，所以用 linux 系统调用将 timer 也做成 fd 就可以同时用 select 处理。用 select 提高了效率，而且是单线程不用考虑多线程同步的问题。

4 代码架构

5 遇到的问题和解决

5.1 用 builder 创建 tun 设备之后, IPv6 Socket 的 log 显示有很多错误的数据包

比如，我们首先到手机连接的 ap 上用 tcpdump 抓包，发现建立连接之后就没有发送任何数据了，但是 log 显示我们发送了数据，这是为什么呢？

- 说明我们写到 socket 的数据并没有发送出去，我们首先应该检查路由表。iptables -L 需要 root 权限，android 模拟器可以拿到 root 权限，然而 android 模拟器不支持 ipv6，而且在手机上又没有 root 权限，没办法检查路由表；
- 我们 adb shell 连接到手机上，ping 服务器的 Ipv6 地址, 发现能 ping 通, 这说明路由表应该是好的；
- 然后我们再次打开 VPN（adb 那边还在一直 ping 着），VPN 连接建立之后，ping 突然就不通了，这说明就是 VPN 连接的建立导致了这个错误；
- 由于 IPv6 Socket 建立成功了，说明这时候 IPv6 连接还是好的，所以只能是建立 tun 设备的时候影响了 IPv6 Socket；

- 我们检查了一下建立 tun 设备的 builder，发现路由我们已经设置 0.0.0.0/32 了，但是还是不知道怎么回事；
- 我们在 github 上搜索了一下 OpenVPN 的源码：

```
219     public boolean socket_protect(int socket) {
220             boolean b = mService.protect(socket);
221             return b;
222
223         }
```

  可以发现, OpenVPN 在 C 中建立 tun 设备，然后 JNI 回调 Java，用 protect 来“保护”Socket；

- 我们查阅了 Android 的文档，也说需要在 Socket 上调用 protect，阻止该 Socket 通过 VPN。于是，我们加上 protect——就好了……

5.2 发现 ping 和 DNS 查询基本都能成功，但是网页打不开，微信偶尔能发出去

我们的解决过程：

- 为了找出这是为什么，研究了一下 pcap 数据文件的格式；
- 在建立好 tun 设备之后，将 tun 设备的所有流量转换成 Wireshark 能识别的 pcap 格式，保存到 SD 卡；
- 然后复制到电脑上用 Wireshark 打开查看，发现 HTTP 请求一般是发到一半，TCP 连接就断开了，并且发送了很多 retransmission 的包；
- 不明所以，于是到课上问老师，说可能是用 Java 建立 Socket 的原因；
- 后来改成了 C 建立 Socket，HTTP 请求就能成功了，但是还是有点慢，而且 log 显示 Socket 发了很多不合法（length 不等于包长）的包——我们以为是服务器为了测试客户端鲁棒性故意发的；
- 后来课上跟老师讨论了 mtu 的问题，将 mtu 设为 800，还是没有根本解决问题，又改成了接收到不合法数据包后继续读数据，还是没有彻底解决这个问题……