网络专题训练

4over6 隧道协议实验报告

马　也　2013011365　计 34
沈哲言　2013011371　计 34

May 31, 2016

1　实验目的

- 掌握 Android 下应用程序开发环境的搭建和使用
- 掌握 IPv4 over IPv6 隧道的工作原理

2　实验原理

2.1　4over6 隧道模式原理

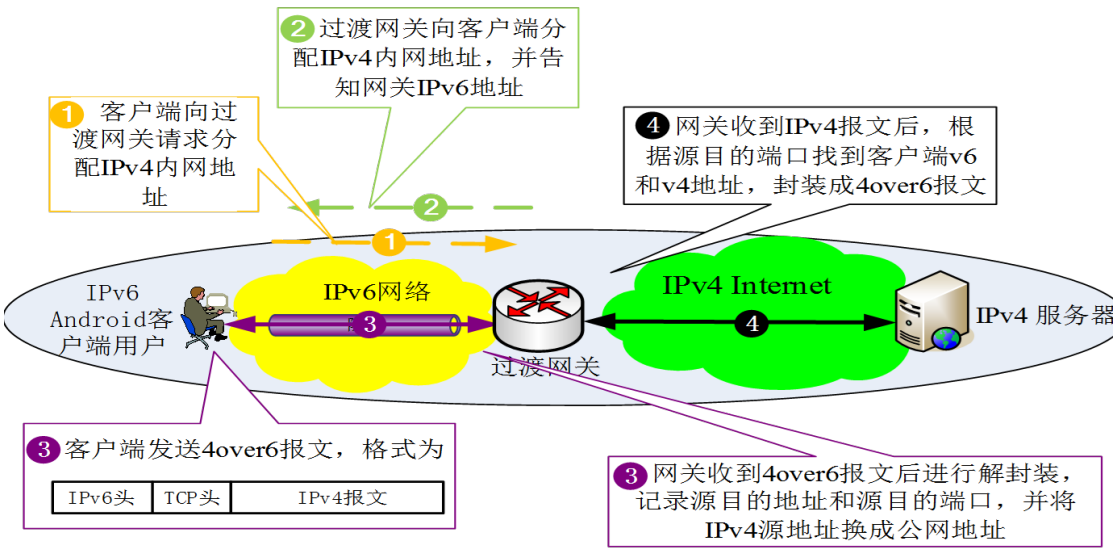

Figure 2.1: 4over6 隧道模式原理图

IPv4 over IPv6 隧道模式是指，将 IPv4 的报文装载到 IPv6 网络的数据段，通过隧道模式进行传递，这样就可以使用 IPv6 的网络来访问 IPv4 的资源。主要流程如下：

安卓客户端向过渡网关请求分配 IPv4 内网地址；过渡网关分配 IPv4 内网地址，并提供对应的 IPv6 网络地址；接着，安卓客户端发送 4over6 报文，过渡网关接受报文并进行分析，得到源地址和目的地址，将 IPv4 地址转化为公网地址，发送到公网之中；当过渡网关收到公网的 IPv4 报文之后，根据记录好的映射关系，重新封装成 4over6 报文，发给对应的内网用户，完成数据的转发和接受。

在以上流程中，用户处于 IPv6 的网络环境之中，过渡网关横跨 IPv4 和 IPv6，并在其中提供 IPv4 和 IPv6 地址的转换和分发。在本实验中，过渡网关部分的代码已经完成，我们需要完成用户端的代码，以实现 IPv4 over IPv6 隧道协议。

2.2 安卓 VPN Service 原理

在本实验中，我们采用了 VPN Service 作为基本框架来实现 IPv4 over IPv6 隧道协议。VPN Service 是安卓提供的一套 API 接口，方便编程人员创建 VPN 服务。当打开该服务之后，安卓系统将所有应用程序发送的 IP 包都根据 iptables，使用 NAT 转发到 TUN 虚拟网络设备上，其端口为 tun0。当打开 VPN Service 之后，我们可以获取 tun0 的文件描述符，这样就可以读取或者写入数据以实现发送或者接收数据。

也就是说，在打开 VPN Service 之后，我们可以监控系统所有的网络进程，并从 tun 中获得系统中所有 IP 包。这样，我们就可以将 IP 包封装在 IPv6 报文之中，使用隧道协议将报文发送到服务器，实现数据的发送；当我们收到服务器反馈的 IPv6 报文后，我们将其进行拆解，得到真正的 IPv4 包，再写入到 tun 文件之中，以实现数据的接收。

2.3 JNI 与 NDK

在本次实验中，由于经常要和以字节为单位的数据打交道，并且需要精细地管理内存，因此我们采用 C 来实现 IPv4 over IPv6 的核心功能。要在以 Java 为语言的安卓环境中使用 C 来进行编程，我们需要使用 JNI 和 NDK。

Java Native Interface（JNI）规定了一套 Java 的原生接口，规定了 Java 和其他语言交互或者是在其他平台下运行时的接口，让我们可以使用其他语言（如 C、C++、汇编等）访问 Java 中的类和对象。而 Native Development Kit（NDK）是 Google 提供的一套方便开发者的开发套装（SDK），方便编程人员将 C 或者 C++ 程序打包并加

入到 APK 之中，供安卓程序使用。

3 实验设计与内容

3.1 程序整体设计流程

本次实验的编码主要分为两个部分，Java 实现的安卓客户端（前端）和 C 实现的收发进程（后端），前端和后端之间通过管道（named pipe）来进行连接并传递数据。实验整体流程如下：

Figure 3.1: 整体工作流程

3.2 前端工作流程

3.2.1 工作流程

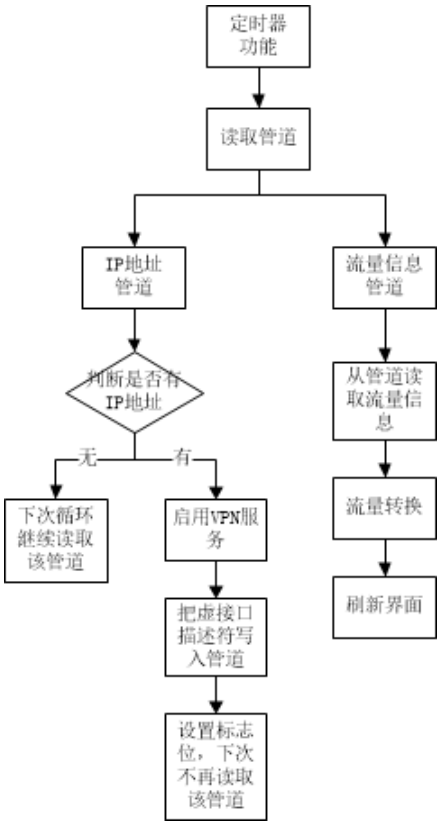

Figure 3.2: 前端工作流程

前台流程如上图所示，前台的前台定时器主要功能为定时刷新显示界面，显示流量信息，定时器功能解析如下：

1. 开启定时器之前，创建一个读取 IP 信息管道的全局标志位 flag，默认置 0；
2. 开始读取管道，首先读取 IP 信息管道，判断是否有后台传送来的 IP 等信息；
3. 假如没有，下次循环继续读取；
4. 有 IP 信息，就启用安卓 VPN 服务 (此部分在后面有详细解释)；
5. 把获取到的安卓虚接口描述符写入管道传到后台；
6. 把 flag 置 1，下次循环不再读取该 IP 信息管道；
7. 读取流量信息管道；
8. 从管道读取后台传来的实时流量信息；
9. 把流量信息进行格式转换；
10. 显示到界面；

11. 界面显示的信息有运行时长、上传和下载速度、上传总流量和包数、下载总流量和包数、下联虚接口 V4 地址、上联物理接口 IPV6 地址。

3.2.2 具体实现

这一部分主要包括了安卓 APP 界面的实现、UI 主线程、VPN 服务、后台定时器刷新线程以及后台 C 线程的实现。

界面： 界面使用了安卓开发中较为常用的 LinearLayout 来实现主界面，在界面上布局了两个 Button（分别用于启用和终止服务）以及六个 TextView（分别显示服务持续时间、上传下载速度、上传下载流量和 IP 信息等等）

UI 主线程： MainActivity 的主要任务是初始化全局的变量、启动两个后台的线程并打开 VPN 服务。首先主线程通过 getLocalIpAddress 函数获取本机的 IPV6 地址，只有成功获取到地址才会继续后续的服务。其次为启动服务按钮注册监听事件，若按下则会启动两个后台的线程 IVI 和 BackGround。

IVI 线程： 通过 JNI 调用后台的 C 函数启动后台 C 线程；

BackgGround 线程： 这个线程的工作比较多，首先它需要连接 IP 信息的管道（由后台 C 线程负责创建），接收来自 C 线程的虚接口的 IPV4 地址，然后启动 VPN 服务，并把 TUN 的文件描述符通过管道传回 C 线程；最后启动定时器服务例程刷新页面。

VPN 服务： 继承一个 VpnService 的类，根据传入的参数（IP 地址，DNS 服务器等等）初始化 Vpn 服务并启动，获取到 TUN 虚接口的文件描述符后，通过管道传回 C 线程

定时器刷新例程： 集成了 java.util.TimerTask 类，连接了后台 C 线程创建的流量信息管道，进行一定的格式转换之后显示在界面上。

3.3 后端工作流程

3.3.1 工作流程

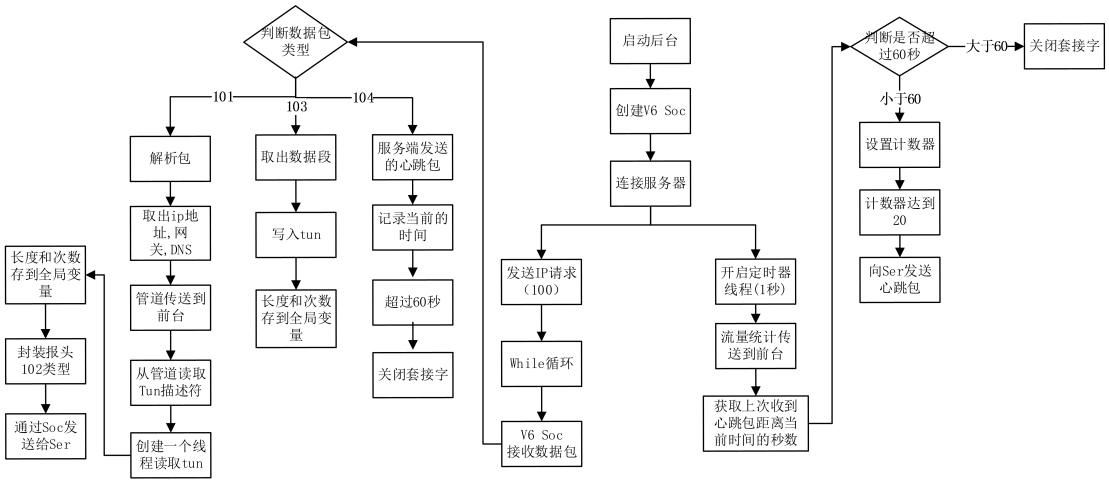

Figure 3.3: 后端工作流程

1. 创建 IPV6 套接字；

2. 连接 4over6 隧道服务器；

3. 开启定时器线程（间隔 1 秒）：

   (a) 读写虚接口的流量信息写入管道；

   (b) 获取上次收到心跳包距离当前时间的秒数 S；

   (c) 假如 S 大于 60，说明连接超时，就关闭套接字；

   (d) S 小于 60 就每隔 20 秒给服务器发送一次心跳包。

4. 发送消息类型为 100 的 IP 请求消息；

5. while 循环中接收服务器发送来的消息，并对消息类型进行判断；

   (a) 101 类型 (IP 响应)：

      i. 取出数据段，解析出 IP 地址，路由，DNS；

      ii. 把解析到的 IP 地址，路由，DNS 写入管道；

      iii. 从管道读取前台传送来的虚接口文件描述符；

      iv. 创建读取虚接口线程：

         A. 持续读取虚接口；

B. 记录读取的长度和次数；

C. 封装 102(上网请求) 类型的报头；

D. 通过 IPV6 套接字发送给 4over6 隧道服务器。

(b) 103 类型 (上网应答)：

i. 取出数据部分；

ii. 写入虚接口；

iii. 存下写入长度和写入次数。

(c) 104 类型 (心跳包)：

i. 记录当前时间到一个全局变量。

3.3.2 具体实现

后端使用 C 语言 SOCKET 编程实现的，其有一个主函数 main 用于 Java 线程执行，其主要流程参见图 3.3，主要阐述如下：

首先，创建和服务器连接时使用的 Socket，并进行地址绑定，建立 TCP 连接。注意这里创建的是 IPv6 的 Socket，创建和连接方法为：

```
client_socket = socket(AF_INET6, SOCK_STREAM, 0);


server_socket.sin6_family = AF_INET6;
server_socket.sin6_port = htons(SERVER_PORT);
inet_pton(AF_INET6, SERVER_ADDR, &server_socket.sin6_addr);


connect(client_socket, (struct sockaddr *) &server_socket,
        sizeof(server_socket));
```

以上代码创建了客户端 Socket，绑定了服务器地址，并进行 TCP 连接，其中`AF_INET6`指明是 IPv6，`SOCK_STREAM`指明是 TCP 连接。

连接成功之后，新建一个定时器线程 `timer_thread`，用于每次隔 1s 执行一次操作，其主要负责：每秒给前端发送统计数据，每 20s 给服务器发送心跳包，每秒检查服务器心跳包是否超时（60s）。发送的统计数据包括这 1s 内接受数据包的数量、发送数据包的数量、接受数据包的总大小、发送数据包的总大小。

回到主函数，在连接成功之后，向服务器发送第一个 IP 请求包 IP_REQ，然后开始循环等待服务器回应：

```
while(!isClosed) {
// Receive Package

// Parse Package

if(!hasIP && msg.type == MSGTYPE_IP_REC) {
// Receive IP
}
else if(msg.type == MSGTYPE_DATA_RECV) {
// Receive data and update stats
}
else if(msg.type == MSGTYPE_HEARTBEAT) {
// Update heart beat time to now
heartbeat_recv_time = time((time_t*)NULL);
}
else {
// Unknown
}
}
```

收到的包分为三个类别：收到 IP 包，收到数据包，收到心跳包。下面分别进行阐述：

IP_REC

当收到 IP 包，就证明和服务器的连接已经建立好了，这时候需要依次完成如下步骤：写入 IP_REC中的数据到管道之中，从管道中读取 tun 文件句柄号，创建新的线程循环读取 tun 文件。由于 C 语言中字符串处理操作较为复杂，故对 IP_REC包中的数据不做处理直接发送给 Java，由前端获取每一段的信息，开启 VPN Service。对于 tun 文件，前端只需要给后端发送其句柄号（在 Linux 系统中每个进程中每个打开的文件都

有其特有且唯一的句柄号），后端就可以对这个 tun 文件进行读写了。因此，最后创建新的线程循环读取 tun 文件的内容。

在读取 tun 文件的时候，有一个小的优化，为了不让循环检查 read 始终阻塞进程“忙等”，这里使用 Linux 中的 select 函数进行“闲等”，具体可以参阅 Linux 中 select 函数的文档。

DATA_REC

当收到数据包，就代表收到了服务器转发的数据，这时候直接将其 data 字段写入 tun 文件之中即可，并更新统计信息。更新统计信息时需要注意加锁操作，防止竞争问题。

HEARTBEAT

当收到心跳包的时候，更新收到心跳包的时间即可（因为心跳包的发送和超时检查在定时器线程）。

4 实验结果与分析

在本实验的测试过程中，使用了安装了安卓系统的华为手机进行了测试，网络环境为在寝室用极路由搭建的 Ipv6 局域网环境。在手机开启了 VPN 服务之后，我们测试了各种传输对象和传输环境，包括但不限于：内网环境下的 Info 站点的图片及文字；外网环境下百度首页、阅读、图片等各种百度服务（浏览器端），微信文字聊天、图片、小视频发送以及视频通话功能（客户端）；以及一些 APP 的后台推送功能。

通过测试，我们发现手机的上网功能基本可用，说明了我们的 VPN 服务功能和 4 over 6 实现基本正确。在 IPv6 网络稳定的情况下，打开 info 主页这种文字较多的网页所用的延迟大概为一到两秒，这说明了我们实现的高效性；在测试微信视频聊天的时候，我们发现手机收到的视频图像几乎没有卡顿，仅仅是发送出去的视频有略微的卡顿，这也证明了我们实现的高效与鲁棒。

5 实验中遇到的问题

在实验中，我们主要遇到了三个比较大的问题。

第一个问题是，当我们后端 C 代码和前端 VPN Service 结合在一起之后，就发送不出去心跳包了。经过调试，我们发现是开启 VPN Service 之后，原来创建的 C 的 socket 也会被导入到 tun 文件中，这就形成了自环。解决方法比较简单，我们使用安卓中提供的 protect 函数保护之前建立好的 socket 就可以了。至于这个 socket 则通过一开始的 named pipe 和着 IP 地址一并传送到前端即可。

第二个问题是，在编码完毕后，我们在调试信息中发现，经常有类型不等于预设的几种类型的报文，并且这些报文之前一定紧跟着一个最大长度的报文，也就是说，IPv4 报文在传输的过程中被截断了。由于我们和服务器之间的通信没有 size 和 offset 等设置，所以一旦被截断也不可能在客户端进行拼接和恢复。这就导致了我们在一开始测试微信发送图片以及视频聊天时速度奇慢，几乎不能正常使用，打开 info 网页至少需要 5-6 秒才能打开，并且图片几乎加载不出来。

为了解决这个问题，我们尝试了许多办法，包括设置 MTU 长度为小于 1500 的一个值，发现几乎没有改变。另外，我们还尝试接收时不一次性接收完，而是持续接收，直到接收 size 个字节才停止，但是也没有明显地改善。最后，我们发现，首先接受 4 个字节（size），然后接受 size 次，每次一个字节，这样的操作可以极大减少报文被分拆的可能，提高有效传输速度，所以我们最后采用了这种做法，也得到了实验的证实。

第三个问题是有关于 UI 线程阻塞的问题，UI 线程作为维护安卓界面的主线程，在其中不应该设置非常多耗时且有可能阻塞的操作，在和管道交互时如果将读取等操作放在主线程很有可能导致界面的阻塞，故应该新增一个线程进行 IO 操作并刷新界面。

6 实验心得与体会

在这次实验中，我们掌握了 Android 下应用程序开发环境的搭建和使用，理解了 4over6 的原理，并实现了一个测试通过，效果良好，具有鲁棒性的 4over6 安卓 APP。除此之外，我们对于原理课上所讲的 4over6 又有了更深的理解和体会，并通过自己的切身实践，锻炼和编码能力，深化了对于转发、隧道等概念的理解，达到了该大实验的目的。

7 对实验的意见和建议

本实验有效锻炼了同学们的编码能力和实践能力，但是我们认为还存在些许的瑕疵。根据我们的完成情况，前端同学几乎只需要做一个 Activity 界面，并学习 Android VPN Service 部分文档即可，后端同学则只是涉及到 socket 编程和一些简单的 C 语言知识，后端全部代码加起来不过 300 行左右。代码编写时间也不会太长，仅仅是调试时花费了一些时间。我个人感觉，从难度和编码量上，与其他课程的大实验相比，还是略显简单。我的建议是，提出许多具有开放性的扩展任务（如好看的 UI、XXX 的下载速度、服务端代码重构、流量提醒、切换 IP、后台模式等等），并按照扩展任务的完成情况进行评分，这样可以调动同学们的积极性，并且丰富实验内容。

除此之外，可以考虑扩充服务器端和客户端之间的传送协议，加入诸如校验和、offset 等报文头，支持客户端上的报文拼接和验证等等，提高系统的鲁棒性。

以上就是一些小小的建议，最后感谢老师和助教给予的帮助和指点。