基于LSP的SOCKS5代理客户端设计

范雄飞[1]，汪 霖[2]

（1.二炮指挥学院，湖北 武汉 430012；2.通信指挥学院，湖北 武汉 430010）

摘 要：随着Internet技术的飞速发展和广泛应用，网络已经成为人们工作和生活中不可缺少的一部分，而网络中的代理技术更是得到了普遍应用，如IE、QQ、MSN等常用的网络工具都提供了代理支持。但目前仍然有很多的应用程序尚不支持代理，通过深入研究Winsock和SOCKS5代理技术，基于LSP完成了SOCKS5代理通用客户端的设计与实现。

关键词：LSP；SOCKS5；Winsock；网络；代理

中图分类号：TP393.03 **文献标志码**：A **文章编号**：1673-9833(2009)05-0102-04

The Design of SOCKS5 Proxy Client Based on LSP

Fan Xiongfei[1]，Wang Lin[2]

（1. The Second Artillery Command College，Wuhan 430012，China；
2. Communication Command Academy，Wuhan 430010，China）

Abstract：With the rapid development and widely application of internet, network is becoming an indispensable part in the people work and live. Network agent is put into more general application, such as IE、QQ and MSN. Through the study of Winsock and SOCKS5 proxy technology, the design and implementation of general SOCKS5 client based on the LSP are completed.

Keywords：LSP；SOCKS5；winsock；network；proxy

1 LSP技术

在Windows网络编程中，Windows提供了标准的API接口，称为Windows Sockets 2（Winsock），其结构如图1所示[1]，它是Windows下得到广泛应用的、开放的、支持多种协议的网络编程接口，经过不断完善，它已成为Windows网络编程的事实上的标准，该接口大大方便了多个应用程序（或进程）在同一台机器或网络中进行通信。同时，Winsock还在它提供的API和Windows的协议栈之间提供了一个标准的服务提供者接口SPI（service provider interface），以允许通过第三方的服务提供者来监视应用程序对Winsock的调用，从而允许开发人员在不重写代码或替换Winsock2 DLL（ws2_32.dll）的情况下，为Winsock应用程序提供额外的服务。

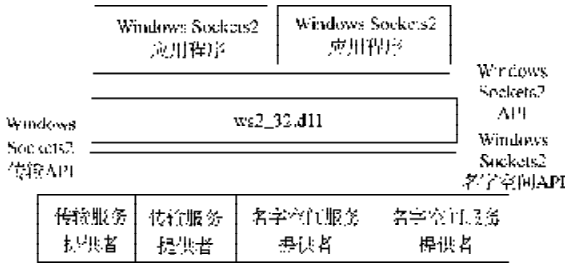

图1 Winsock2结构图

Fig. 1 Winsock2 structure

SPI又分为传输服务提供者（transport service provider, TSP）和名字空间服务提供者（name space provider, NSP），其中传输服务提供者支持1个或多个

收稿日期：2009-07-20

作者简介：范雄飞（1976-），男，湖北武汉人，二炮指挥学院副教授，主要研究方向为指挥自动化，通信技术及应用等，
E-mail：fxfpl@sina.com

协议，所以通常称为协议堆栈，它为用户提供了建立连接、传输数据、流控制和差错控制等服务；名字服务提供者提供了名字服务，从而允许 1 个或多个友好名字与协议地址相关联，允许独立于协议的名字解析。SPI 允许 2 种类型的传输服务提供者：基础服务提供者（base service provider, BSP）和分层服务提供者（layered service provider, LSP）。BSP 实现了传输协议（如 TCP）的真正细节，而 LSP 仅实现了高层自定义的通信函数，并依赖于已经存在的下层基础服务提供者，完成与远端服务器的真正数据交换。而且，Winsock 2 SPI 是较灵活的，它允许多个 LSP 分层叠放，如图 2 所示[1-4]。

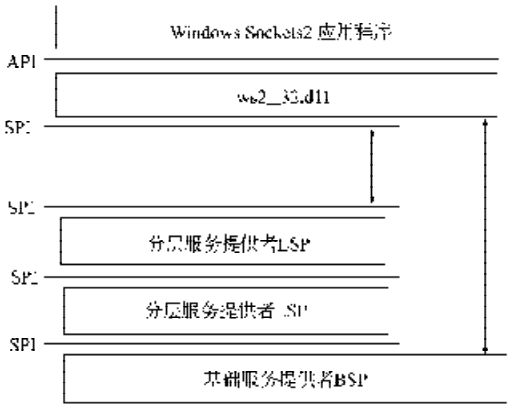

图 2　TSP 结构图

Fig. 2　TSP structure

当 1 个 Winsock 应用程序发起 1 个 Winsock 调用时，系统将路由这个调用到最顶层的 LSP，在那里，LSP 可以执行需要的任务，模仿或扩展 1 个已经存在的服务提供者，例如实现加密、过滤、透明代理等功能；然后，本层 LSP 再传递请求到它下面的 1 层 LSP；最后，这个调用将到达 BSP，并交给主机可用的协议驱动程序进行处理。

Winsock SPI 允许 2 种不同的 LSP 模型[1]：Installable File System LSP（IFS LSP）和 non-Installable File System LSP（non-IFS LSP）。这 2 种 LSP 开发模型从 Windows2000 开始已经可用。其中，IFS LSP 不需要用户创建自己的 SOCKET 句柄，它使用下层提供者的句柄，它可以选择实现 Winsock SPI，不需要全部实现所有的 SPI 函数，不需要处理 Windows I/O 完成端口和 WSPAsyncSelect 模型。由此可见，IFS LSP 并不能支持所有 Windows I/O 重叠操作的后处理过程，如在异步 I/O 操作中，等待操作结果，并执行后处理操作是不行的，必须使用 non-IFS LSP。但 non-IFS LSP 必须自己实现所有的 SPI 函数，处理复杂的 Windows I/O 完成端口模型，必须创建自己的句柄并映射到下层提供者。微软已经为用户提供了基本的 non-IFS LSP 框架，用户也可以很容易地在其基础之上扩展实现所有的 SPI 函数，同时也能在每次操作完成之后进行相应的处理。

2　SOCKS5 协议

SOCKS 是网络代理协议，包含了 2 个组成部分，SOCKS 服务器和 SOCKS 客户端（如图 3 所示），它的基本目的是允许 SOCKS 客户端能够通过 SOCKS 服务器获取对远程主机的完全访问，而不需要 SOCKS 客户端直接连接到远程主机。SOCKS 又分为 V4 和 V5 这 2 个版本，其中 V5 在 V4 的基础增加了对 UDP 代理的支持，并支持对客户端的授权和认证，相比之下，V5 更灵活安全。SOCKS5 协议[5-6]描述在 RFC 1928 文档中，针对 TCP 和 UDP 协议的客户端，其代理服务器的处理方式略有不同。

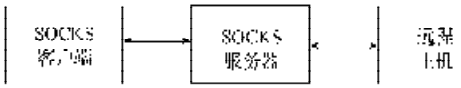

图 3　SOCKS 组成

Fig. 3　SOCKS composition

由于 SOCKS5 协议的内容很容易从网上获取，所以本文不再赘述。这里只给出基于 TCP 的 SOCKS5 代理客户端与代理服务器的简要工作流程：

1）客户端与代理服务器建立 TCP/IP 连接；

2）客户端向代理服务器发送第 1 个协商封包，请求 SOCKS 协议的版本和验证方式的索引，这里版本信息为 0x05，验证方式的索引一般为 0x02，即使用用户名 / 密码验证方式；

3）当代理服务器接收到客户端的请求后，它会向客户端发回应答信息，包括确认的版本信息，必须与客户端请求的版本号一致，这里为 0x05，如果代理服务器为客户端选择的也是用户名 / 密码验证方式，则返回 0x02；

4）客户端根据代理服务器的返回信息进入验证子协商处理过程，客户端向代理服务器发送第 2 个协商封包，包括用户名和密码信息；

5）当代理服务器接收到客户端的用户名和密码后，向客户端返回验证成功与否的信息；

6）客户端向代理服务器发送第 3 个协商封包，即客户端需要连接的目的主机的 IP 地址和端口号。

当代理服务器接收到客户端发送的目标主机 IP 地址和端口后，它会向客户端返回应答信息，表示服务器是否成功连接远程主机，如果应答信息表示成功，则客户端就可以通过代理服务器与远程主机透明地传输数据了。

3　代理客户端的设计

由上述的理论基础，本文将采用 LSP 来实现通用

的SOCKS5代理客户端。根据SOCKS5协议，要拦截用户应用程序中的TCP数据封包，并通过代理服务器传递到远程主机，其基本过程如图4所示[6]。

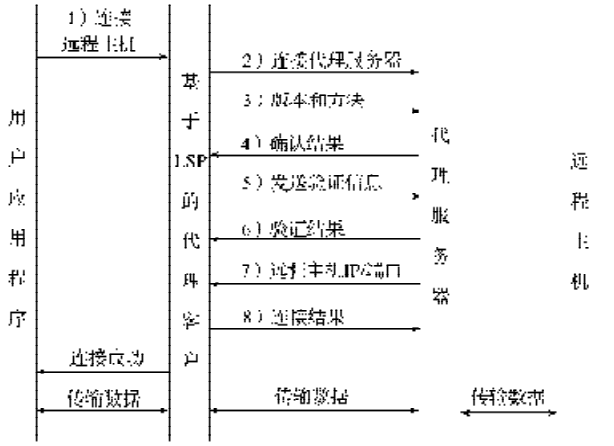

图4 应用程序通过代理连接远程主机的通信过程

Fig. 4 The communication process of application programs connecting to the remote host through the proxy

1）用户应用程序调用与远程主机的连接函数；

2）拦截用户应用程序中连接远程主机的函数，修改其IP和端口到代理服务器的IP和端口，并将原来的IP和端口保存，然后连接到代理服务器；

3）如果连接成功，并不立即通知用户应用程序连接成功，而是让用户应用程序等待，这时发送版本请求信息和验证方法的序号；

4）代理服务器发回确认的应答信息；

5）基于LSP的代理客户端向代理服务器发送用户名/密码验证信息；

6）代理服务器发回验证结果应答信息；

7）当验证成功后，向代理服务器发送目的IP和端口信息；

8）代理服务器发回它与远程主机的连接结果应答信息；

9）当代理服务器返回成功的应答信息之后，基于LSP的代理客户端才通知用户应用程序成功连接远程主机，用户应用程序即可以通过代理服务器与远程主机实现透明通信。

3.1 确定拦截函数

从图4可以看出，基于LSP的代理客户端首先需要拦截用户连接远程主机的函数，在基于Winsock的应用程序中可以通过connect、WSAConnect和ConnectEx（仅WindowsXP）3个函数发起连接调用[1-4]。其中connect、WSAConnect直接映射到LSP中的WSPConnect函数，在代理客户端中直接用WSPConnect函数处理，并调用下层服务提供者的WSPConnect函数即可。而ConnectEx函数是扩展函数，它需要通过调用WSAIoctl函数获取其函数指针，通过函数指针来调用，所以必须在代理客户端中先拦截WSAIoctl函数，在WSPIoctl向用户返回自定义的ExtConnectEx函数，而代理客户端自定义的ExtConnectEx函数可以调用下层提供者的ConnectEx函数。所以，在代理客户端中需处理WSPConnect函数和ExtConnectEx函数，在这2个函数中将用户应用程序提供的目标主机IP地址和端口信息替换成代理服务器的IP地址和端口信息。值得注意的是，这里需要保存由用户应用程序提供的原目标远程主机IP地址和端口信息，这主要有2方面的原因：1）在后面需要向代理服务器发送这个目标主机的IP地址和端口信息；2）用户应用程序可能会调用getpeername函数获取远程主机的IP地址和端口信息，这时应该返回这个已保存的真实的远程目标主机的IP地址和端口信息，而不是代理服务器的IP地址和端口信息。最后在WSPConnect函数和ExtConnectEx函数中分别调用下层服务提供者的WSPConnect函数和ConnectEx函数。

3.2 确定连接成功

在代理客户端中发起对代理服务器的连接之后，需要判断是否连接成功，根据Winsock的I/O处理方式，有如下几种情况。

3.2.1 阻塞模式调用[1-2]

这种模式很简单，在调用下层服务提供者的WSPConnect函数和ConnectEx函数返回后，代理客户端直接就能够判断连接成功或失败，如果连接成功，则可以进行后续的处理过程，并等待后续的处理过程完成之后才向用户返回。

3.2.2 非阻塞模式调用[2-3]

这种模式较复杂，在调用下层服务提供者的WSPConnect函数和ConnectEx函数返回后，代理客户端不能直接判断连接成功或失败，需要通过其它的函数来判断。

1）select模型

在select模型中，当下层的WSPSelect函数返回时，如果前面连接的套接字在WriteFds中，则说明连接获得成功。

2）WSAEnumNetworkEvents模型

在WSAEnumNetworkEvents模型中，当下层的WSPEnumNetworkEvents返回时，如果在WSANETWORKEVENTS结构的lNetworkEvents成员中包函了FD_CONNECT事件，而且没有错误发生，则说明连接成功。

3）WSAAsyncSelect模型

在WSAAsyncSelect模型中，在代理客户端中需要创建自定义窗口来接收下层服务提供者的事件通知，并将接收到的事件转发给应用程序。如果在代理客户端的窗口消息处理函数中接收到FD_CONNECT事件，而且没有错误发生，则说明连接成功。

3.2.3 ConnectEx 函数调用

当ConnectEx函数工作于阻塞模式时，同3.2.1。当为ConnectEx函数指定了LPOVERLAPPED结构，即执行重叠I/O操作时情况比较复杂。在代理客户端中采用了完成端口和专门的工作者线程来转发用户应用程序中所有重叠I/O操作，当连接成功之后，代理客户端中的工作者线程会接收到成功连接的通知，这样就可以进行后续处理了。

3.3 协商过程处理

当代理客户端与代理服务器成功连接之后，不能立即向用户应用程序返回，或通知用户应用程序连接成功，需要做进一步的协商过程处理。由于select函数不会改变套接字的阻塞和非阻塞模式，所以代理客户端实现中采用下层服务提供者的WSPSelect函数来处理与代理服务器之间的数据封包的收发，当协商过程完成后：

1）如果是阻塞模式调用，则从LSP提供的WSPConnect函数或ExtConnectEx返回；

2）如果是通过WSPSelect判断连接成功，则从代理客户端提供的WSPSelect函数返回；

3）如果是通过WSPEnumNetworkEvents判断连接成功，则从代理客户端提供的WSPEnumNetworkEvents函数返回；

4）如果是通过WSAAsyncSelect判断连接成功，则调用WPUPostMessage函数通知用户应用程序已经连接成功；

5）如果是通过ConnectEx和重叠I/O判断连接成功，则调用WPUCompleteOverlappedRequest函数通知用户应用程序连接成功。

4 代理客户端的实现

通过上述设计，可以实现基于LSP实现SOCKS5的通用代理客户端程序，它能按照用户的需求将不支持SOCKS5代理的用户应用程序通过SOCKS5代理服务器与远程主机连接，并与远程主机进行透明地通信。由于服务提供者是以动态链接库的形式存在，由调用ws2_32.dll的所有上层应用程序共享，所以在实现代理客户端的动态链接库时必须注意进程和线程之间的同步问题[3-4]。

5 结语

本文在剖析LSP技术和SOCKS5协议的基础上，通过详细分析在LSP中如何确定需要拦截的函数、如何判断连接成功以及如何与代理服务器协商等过程，成功地实现了SOCKS5的通用代理客户端程序，使不支持SOCKS5代理的用户应用程序也能够通过SOCKS5代理服务器与远程主机进行透明地通信。该技术同样适用于基于LSP的代理型防火墙的设计，还可应用于对网络封包的拦截分析与处理，具有较强的实用和参考价值。

参考文献：

[1] Microsoft. MSDN Library[EB/OL]. [2008-09-25]. http://www.microsoft.com.

[2] Jones Anthony，Ohlund Jim. Network Programming for Microsoft Windows[M]. 2nd ed. Washington：Microsoft Press，2002.

[3] Petzold Charles. Programming Windows[M]. 5th ed. Washington：Microsoft Press，2002.

[4] Russinovich Mark E，Solomon David A. Microsoft(r) Windows(r) Internals：Microsoft Windows Server(tm) 2003, Windows XP, and Windows 2000[M]. 4th ed. Washington：Microsoft Press，2004.

[5] Leech M，Ganis M，Lee Y，et al. RFC 1928-SOCKS Protocol Version5[EB/OL]. [1996-03-28]. http://www.faqs.org/ rfcs/rfc1928.html.

[6] Leech M. RFC 1929-Username-Password Authentication for SOCKS V5[EB/OL]. [1996-03-28]. http://www.faqs.org/rfcs/rfc1929.html.

（责任编辑：李玉珍）