Proberoute 在 Windows 上的實現

龔存

Abstract

Proberoute 在 Windows 上提供兩種方式發送和接收 IP 報文：使用 WinPcap/Npcap API 或原始套接字（Raw Socket），本文簡述這兩種方式的實現和注意點。

I. 使用原始套接字

和 Unix 操作系統一樣，Windows 也提供原始套接字（Raw Socket）類型，從而提供普通的 TCP 和 UDP 套接字所不具備的三種能力：

- 進程可以讀寫 ICMPv4、IGMPv4 和 IGMPv4 等分組。
- 進程可以讀寫內核不處理其協議字段的 IPv4 數據報。
- 進程可以使用 IP_HDRINCL 套接字選項自行構造 IPv4 首部。

Proberoute 默認使用 libpcap/WinPcap API 捕獲探測回應報文，當環境變量 PROBE_RECV 不為空時，proberoute 通過原始套接字接收探測回應報文。

A. 原始套接字的創建

Proberoute 為發送端和接收端分別創建原始套接字。

1) 創建發送端原始套接字

Proberoute 對於 Windows 和 Unix 的發送端原始套接字的創建步驟一致：

1) 把第二個參數指定為 SOCK_RAW 并調用 socket(2) 函數，以創建一個原始套接字：

```
int rawfd;
rawfd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
```

注意在 Windows Sockets (Winsock2) 編程中，rawfd 的類型應該是 SOCKET 而不是 int, 所指的套接字描述符並不同於 Unix 中所定義的文件描述符（File Descriptor），但為了保持代碼的一致性，Proberoute 採用 POSIX 規範，對於 Windows 和 Unix 平台，均採用一致的寫法。

2) 在這個原始套接字上按以下方式開啟 IP_HDRINCL 套接字選項：

```
const int on = 1;
setsockopt(rawfd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on));
```

Proberoute 不在發送端的原始套接字上執行 bind 和 connect 操作。

2) 創建接收端原始套接字

由於 Windows 上需要開啟原始套接字的混雜模式，並且使用重疊 I/O 接收數據，proberoute 在 Windows 創建接收端原始套接字步驟如下：

1) 通過 WSASocket() 創建原始套接字，并指定重疊 I/O 標誌：

```
SOCKET rawfd = INVALID_SOCKET;
rawfd = WSASocket(AF_INET, SOCK_RAW, IPPROTO_IP,
                  NULL, 0, WSA_FLAG_OVERLAPPED);
```

2) 為了能在接收端套接字上開啟混雜模式，必須和本地接口綁定：

```
/* Bind Local Interface Address */
if (bind(rawfd, addr, addrlen) == SOCKET_ERROR) {
    fprintf(stderr, "bind failed with error: %d\n", WSAGetLastError());
    WSACleanup();
    return -1;
}
```

B. 原始套接字的輸出

Proberoute 構造完整的 IPv4 報文，包括 IP 首部和之後的各類協議報文，以及計算并填充校驗和碼等，然後通過 sendto(2) 系統調用發出報文。由於安全限制，TCP 數據包不能通過原始套接字發送，必須通過 WinPcap 所提供的 NPF 協議驅動程序發出。

C. 原始套接字的輸入

和 BSD Unix 不同，Windows 可以在原始套接字上設置混雜模式（SIO_RCVALL ），從而能夠接收到所有流經網卡的數據，而 BSD Unix 上接收到的 UDP 分組或 TCP 分組絕不傳遞到任何原始套接字。Windows 通過 WSAIoctl() 設置原始套接字為混雜模式，必須指定協議類型為 IPPROTO_IP ，并且對該套接字必須明確和一個本地接口進行綁定：

```
if ((WSAIoctl(rawfd,                        /* descriptor identifying a socket */
              SIO_RCVALL,                   /* dwIoControlCode                 */
              &Optval,                      /* lpvInBuffer                     */
              sizeof(Optval),               /* cbInBuffer                      */
              NULL,                         /* lpvOutBuffer output buffer      */
              0,                            /* size of output buffer           */
              (LPDWORD)&dwBytesReturned,    /* number of bytes returned        */
              NULL,                         /* OVERLAPPED structure            */
              NULL                          /* completion routine              */
    )) == SOCKET_ERROR) {
    fprintf(stderr, "WSAIoctl failed with error: %d\n", WSAGetLastError());
    return -1;
}
```

設置好混雜模式之後，即可調用接收函數 WSARecvFrom() 捕獲 IP 數據包。

D. 使用重疊 I/O 接收數據

重疊 I/O （Overlapped I/O）是 Windows 環境下實現異步 I/O 最常用的方式。由於套接字的默認狀態是阻塞的，這就意味著當發出一個不能立即完成的套接字調用時（輸入、輸出、接受及發起連接），其進程將被投入睡眠，而重疊 I/O 能夠以非阻塞的模式同時處理多個 I/O 請求，從而優化 I/O 處理性能。作為異步 I/O 模型，重疊 I/O 的工作機制是：告知內核啟動某個操作，並讓內核在整個操作（包括將數據從內核複製到應用緩衝區）完成后通知應用進程：

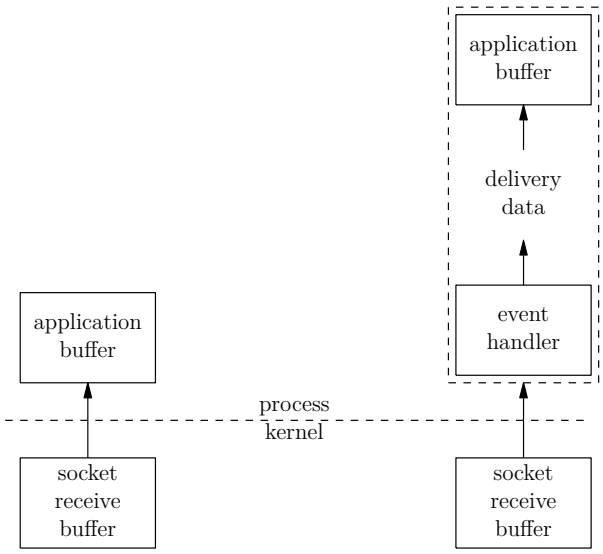

Fig. 1. Difference between Block I/O and Overlapped I/O

儘管 proberoute 進程不需要同時投遞多個套接字的 I/O 請求，因此沒有明顯的必要使用重疊 I/O，但由於開啟混雜模式下，過於嘈雜的接口對於 I/O 響應的要求也隨之增加。由於重疊 I/O 通知應用進程直接使用數據，從而減少了一次從套接字緩衝區到應用程序緩衝區的顯式拷貝，因此 proberoute 選擇使用重疊 I/O 來接收原始套接字數據。

E. 未來改進

儘管 proberoute 採用重疊 I/O 接收數據，但在某些過於嘈雜的網絡接口上，所要接收的對探測數據包的回應報文（ICMP 等）仍可能被大量噪音淹沒，因此一個未來可能的改進是將發送和接收任務劃分到不同進程：

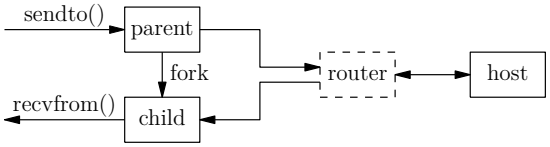

Fig. 2. Send and receive using two processes

同時也應該注意到，對於原始套接字，內核中實際並不存在真正的套接字發送緩衝區，內核僅僅是簡單地將執行路由操作確定外出接口，然後直接把數據報加入數據鏈路層輸出隊列。

II. 使用 WinPcap/Npcap API

WinPcap/Npcap （由於 Npcap 已經作為 WinPcap 的最新版本，以下不再對兩者進行區分，統一簡稱 WinPcap）作為 libpcap 在 Windows 操作系統上的實現，包含一個內核空間數據包過濾器（NPF Device Driver），一個底層動態鏈接庫（Packet.dll）和一個高層并獨立於操作系統的動態鏈接庫（wpcap.dll）：

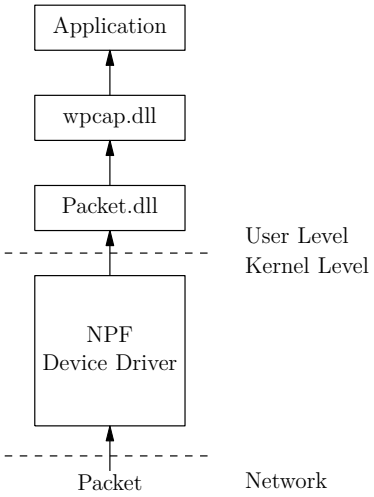

Fig. 3. Main components of WinPcap

網絡上的數據包通過物理接口卡（Network Interface Card，NIC）與 WinPcap 設備驅動程序傳遞到操作系統的內核空間，這個設備驅動程序，稱之為 Netgroup Packet Filter（NPF），實現了對數據包的捕獲、發送甚至流量統計等功能，并可以將數據包交給上層應用程序做進一步的處理，如解析數據包內容并顯示。

Windows 網絡驅動程序接口規範 NDIS（Network Driver Interface Specification）定義了網絡適配器（更準確地說是管理網絡適配器的驅動）與協議驅動程序（Protocol Driver）之間的通訊。NDIS 的主要目的是扮演一個包裝器，允許協議驅動程序不依賴特定的適配器或特定的 Win32 操作系統來發送和接收網絡上的數據包。

NDIS 支持三種網絡驅動程序：

網卡驅動程序 直接管理 NIC。

中間層驅動程序 是上層驅動程序（諸如協議驅動程序）與微端口驅動程序（Miniport Driver）之間的一個接口。

協議驅動程序 用於實現一個網絡協議，如 IPX/SPX 或 TCP/IP 等，它在一個或多個網絡接口卡上提供服務。一個協議驅動程序為其上面的應用程序層的客戶提供服務，同時與一個或多個網卡驅動程序或其下面的中間層 NDIS 驅動程序連接。

NPF 被設計為一個協議驅動程序，下圖顯示了 NPF 在 NDIS 棧中的位置：

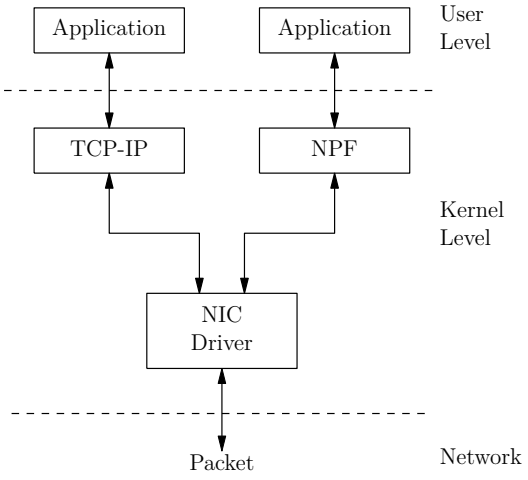

Fig. 4. NPF inside NDIS

協議驅動程序和操作系統的交互是異步的，與 NDIS 的交互也是異步的：當 NPF 調用一個 NDIS 函數時，該調用立即返回；當處理結束時，NDIS 調用一個指定的 NPF 回調函數來通知該函數已經完成。

NPF 能夠執行許多操作，包括：

- 數據包捕獲
- 數據包發送
- 網絡統計
- 數據包轉儲到磁盤

由於安全限制，TCP 數據包不能通過 Windows Socket 原始套接字發送，但通過 NPF 提供的數據包發送能力，用戶程序可以通過 pcap_sendpacket(3) 調用系統函數 WriteFile(2) 寫入 NPF 設備文件，從而發送 TCP 數據包。

III. 關於在 Windows 上對 PPP/VPN 連接進行路由探測

由於 proberoute 是通過捕獲對探測數據包的回應報文來進行路由探測的，因此要在 PPP/VPN 連接上進行路由探測的前提是能夠捕獲 PPP/VPN 連接上的數據。而在 Windows 上，由於不能直接通過原始套接字設置混雜模式來接收 PPP/VPN 連接上的數據（出錯類型為 WSAESOCKTNOSUPPORT：Socket type not supported），而在 Unix 系統上是可以直接在原始套接字上接收 PPP/VPN 數據，因此在 Windows 上捕獲 PPP/VPN 數據必須通過 NDIS 協議驅動。

WinPcap 舊版本（3.1 及以上版本）通過 Microsoft NetMon 驅動器捕獲 PPP/VPN 數據（該驅動器伴隨某些版本的 WinPcap 自動安裝），WinPcap 可以在 Windows 2000/XP (x86)/2003 (x86) 等操作系統中，通過“NdisWanAdapter”（3.1,5 版本之前）或“GenericDialupAdapter”（3.1,5 版本及之後）適配器捕獲 PPP/VPN 數據[1]。但 WinPcap （包括 4.1.3 版本）不支持在 Windows Vista 及以上版本的操作系統中捕獲

[1]通過命令“NetMonInstaller.exe i ”可以觀察是否檢測到 PPP/VPN 連接適配器。

PPP/VPN 數據。

Npcap 作為 WinPcap 的最新版本，在 NDIS 6 的基礎上，通過改進 NPF 協議驅動，從而支持在高版本 Windows 系統上捕獲 PPP/VPN 數據。在安裝了 Npcap 的 Windows 系統上，Proberoute 首先通過 GetAdaptersInfo() 獲得適配器類型（MIB_IF_TYPE_PPP ），然後在適配器“NPF_NdisWanIp”上捕獲 PPP/VPN 數據。但是由於安全限制，TCP 數據報仍然不能通過 NPF 在 PPP/VPN 連接上發送，意味著 proberoute 不支持在 PPP/VPN 上通過 TCP 協議探測路由。