Proberoute - 更為強大的路由探測工具

龔存

Abstract—經典的 **Traceroute** 程序由 Van Jacobson [1] 于 1987 年編寫, 通過發送 UDP 封包偵測路由, 在各類操作系統上均有實現, 如 Windows 上的 **tracert**. 較新的 Linux 發行版上有更為先進的實現 [2], 支持使用多種協議探測. 其他工具比如 **Nmap** 也提供了*--traceroute* 選項來探測路由. **Proberoute** 融合了以上各類工具, 主要針對 IPv4 和內部網絡 (當然也可以用於 Internet), 提供了強大的路由探測功能. 本文將簡單介紹 Proberoute 的實現以及區別于其他工具的特點.

Index Terms—Traceroute, TTL, RFC 792, RFC 793, RFC 1122, LibPcap, WinPcap.

1 簡介

經典的 Traceroute 程序利用 IP 首部的 TTL 字段 (存活時間) 和 ICMP 超時報文 (type 11, code 0) 來獲得路由器 IP [3]: 發送一份 TTL 字段為 1 的 IP 封包給目的主機, 處理這份封包的第一個路由器將 TTL 減 1, 丟棄該報文, 并發回一份 ICMP 超時報文, 這樣就獲得了該路徑中的第一個路由器的地址, 然後 Traceroute 程序發送一份 TTL 為 2 的封包, 這樣我們就得到了第二個路由器的地址. 繼續這個過程直至該封包到達目的主機. 但是目的主機哪怕接收到 TTL 值為 1 的 IP 封包, 也不會丟棄該封包并產生一份 ICMP 超時報文, 這是因為封包已經到達其最終目的地. 那麼我們如何判斷是否已經到達目的主機了呢? 傳統方法是發送一份 UDP 封包給目的主機, 但選擇一個大於 30000 的值作為 UDP 端口號, 假設目的主機的任何一個應用程序都不可能使用該端口, 這樣當該封包到達時, 將使目的主機產生一份 ICMP" 端口不可達" 錯誤報文, Traceroute 程序所要的就是區分接收到的 ICMP 報文是超時還是端口不可達, 以判斷什麼時候結束.

Proberoute 使用相同的原理, 所不同的是 Proberoute 使用原始套接字 (Raw Socket) 和分組捕獲函數庫 **LibPcap**/**WinPcap** 實現路由探測, 同時支持 UDP, TCP 以及 ICMP 三種協議, 當判斷封包是否到達主機時:

- 使用 UDP 時方法如前述, 通過主機反饋 ICMP 端口不可達報文判斷, 參加 RFC 1122.
- 使用 TCP 時默認使用 SYN 封包, 也可以使用 ACK, FIN, URG 等其他非 RST 標誌的封包, 通過主機回應相應的 TCP 報文判斷是否到達目標主機以及端口是否打開, 參考 RFC 793.
- 使用 ICMP 時默認使用 Echo 報文, 也可以使用 Timestamp 或其他報文. 當發送 Echo 或 Timestamp 報文時, 收到對應的 Reply 報文說明到達主機, 參考 RFC 792. 當直接發送 Reply 報文偵測時, RFC 792 並沒有定義主機或路由器未發出 Echo 請求但收到 Reply 的行為, 但大部份主機會迴應 ICMP" 端口不可達" 錯誤報文.

2 改進

相對於傳統的 Traceroute 以及各類變種, Proberoute 最大的特點在於吸收了著名開源軟件 **Nmap** 的防火墻/IDS 躲避與哄騙技術, 同時支持並發探測, 可以同時發送 TCP, UDP 以及 ICMP 三種報文. 另外, Proberoute 採用了 LibPcap 和 WinPcap 分組捕獲函數庫, 從而支

持 Windows, Mac OS X, AIX, GNU/Linux 等各類主流操作系統. 由於分組捕獲函數庫在 *PPP/VPN/IPsec* 等通道連接上的捕獲功能有限, 因此 Proberoute 採用了雙捕獲器機制, 即同時使用 *recvfrom(2)* 方法從 Raw Socket 上接收 IP 數據 [1]. 通過多種協議同時探測和雙捕獲機制 (分組捕獲函數庫和 recvfrom(2) 方法), 達到路由探測的最好結果.

在主機探測方面, Proberoute 提供錯誤校驗和 (*--badsum*)[2][4], 錯誤 IP 長度 (*--badlen*)[3][5] 等選項, 也加入了寬鬆源路徑選項 (*-j*), 可以指定封包所經過的路由器, 這個功能在 Windows 上具有但在 AIX 上卻不支持, 不幸的是, 大部分路由器由於安全原因禁用了該特性, 實際上大部分路由器都丟棄了具有 IP 選項的封包, 儘管如此, 該功能也是值得嘗試的.

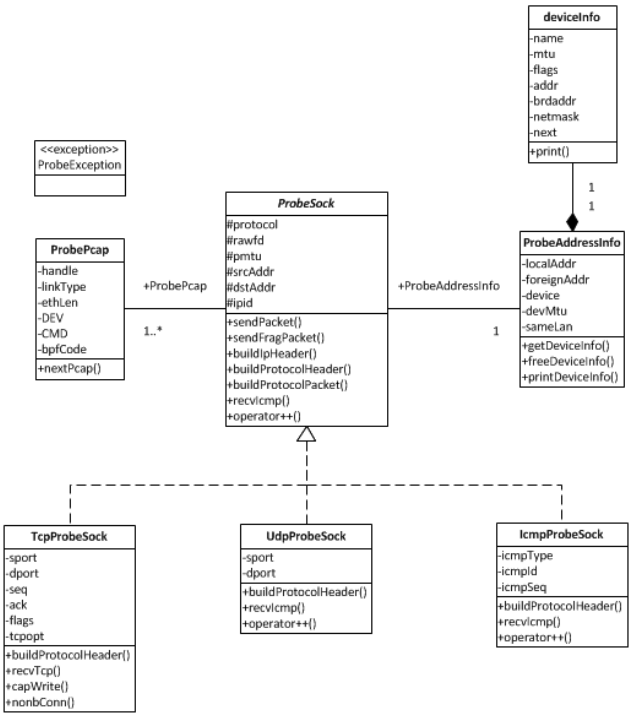
在防火墻/IDS 躲避與哄騙方面, Proberoute 支持源地址欺騙 (*-S*), 源端口欺騙 (*-g*), 報文分片 (*-F*)[4]以及指定接口 MTU(*-s*) 等選項 [4], 另外 Proberoute 還支持直接發送 TCP ACK 報文或 ICMP Reply 報文進行反向探測.

在偵測路徑 MTU 方面, 傳統的 Traceroute 在遇到 ICMP" 需要分片" 錯誤報文時 (type 3, code 4), 通過候選列表猜測下一跳路徑 MTU, Proberoute 通過 ICMP Unreachable Error (Fragmentation Required) 的"MTU of next-hop" 字段來獲得準確的下一跳路徑 MTU 大小, 只有當該字段為 0 時, Proberoute 才會從候選列表猜測下一跳路徑 MTU. 另外, 由於很多嚴格防火墻往往只開通 TCP 端口, 雖然使用 TCP SYN 可以探測路由, 卻無法在不傳輸報文的情況下探測路徑 MTU, Proberoute 使用一種別出心裁的方法, 通過 *connect(2)* 建立連接后再發送帶載荷的失序 TCP ACK 報文來探測路徑 MTU[5].

3 實現

Proberoute 使用 C++98 編譯, 需要使用 LibPcap/WinPcap 函數庫, 沒有使用 C++11/14 及 Boost 函數庫等其他擴展功能. 構造原始 IP 封包的簡易方法是使用 **libnet** 函數庫, 但由於 libnet 函數庫沒辦法很好地在 BSD 系統 (AIX, Mac OS X 等) 上安裝和編譯, 所以 Proberoute 實現了所需使用的 IP, UDP, TCP, ICMP 等報頭的構造及細節, 可以認為 Proberoute 實現了一個迷你 libnet 函數庫. Class diagram 及 Sequence diagram 如下:

Fig. 1. Class diagram



在作者的 Mac OS X 10.12.3 (macOS Sierra) 系統上分別使用 Proberoute 和 Traceroute 對國內部份銀行主頁進行路由探測, 使用的命令如下[6]:

1. Proberoute 在 Windows 上使用 **Cygwin** 編譯, 使用 recvfrom(2) 不能在 Raw Socket 上接收數據 (必須使用 Winsock IOCTL SIO_RCVALL 方法)

2. 很多防火牆出於性能原因不檢查校驗和, 但主機一定會丟棄校驗和錯誤的報文.

3. 用以觸發 ICMP" 參數問題" 錯誤消息, RFC 1122 要求路由器必須 (MUST) 產生此消息, 而主機則應該 (SHOULD) 產生此消息.

4. 同樣由於性能原因, 很多防火牆不檢查分片的報文, 分片的最小單位是 8 字節, 因此甚至有可能將 20-byte TCP 報頭分成 3 片.

5. Windows 系統上不支持該功能

6. 在對比測試中, Proberoute 只採用了基本探測功能, 如果配合其他高級功能使用, 效果還會更好.

Fig. 2. Sequence diagram (double captures)

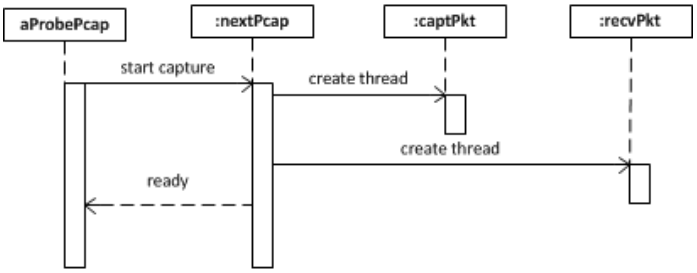


Fig. 3. Sequence diagram (connectionless)

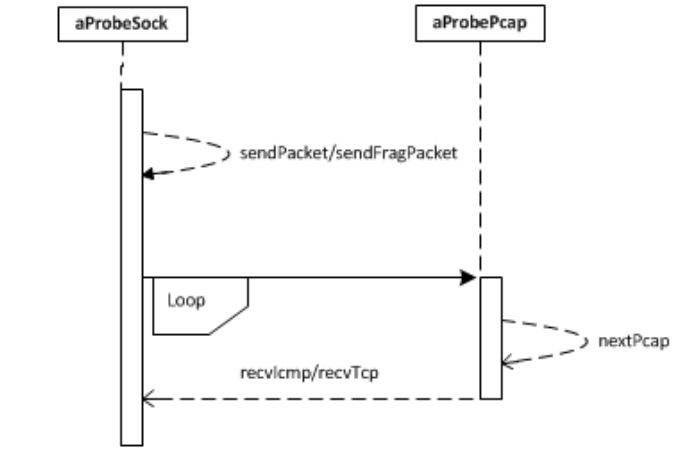
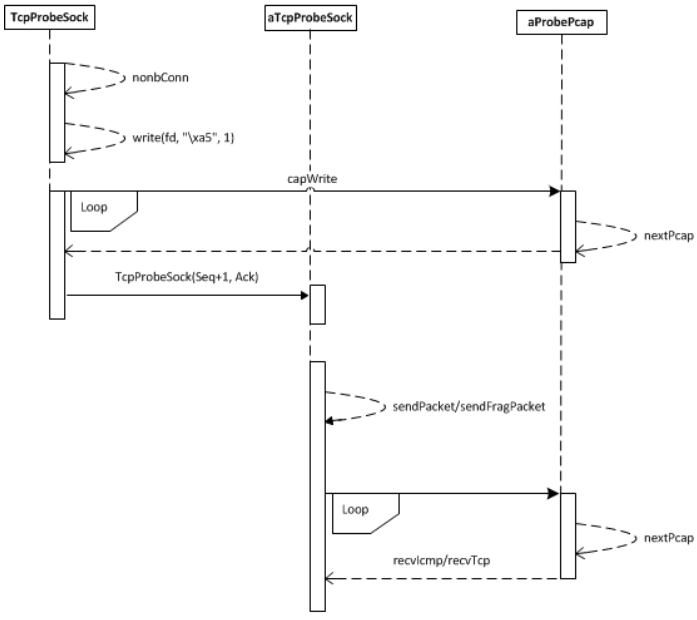


Fig. 4. Sequence diagram (connection)




```
$ proberoute -A -w1 <target> 80
$ traceroute -n -w1 <target>
```

圖5和圖6是探測某一主機時兩者的輸出, 其中某一跳點全為 * 號表明在該跳點未能偵測到任何 IP 地址 (圖5中第 10, 11 等跳點), 這是由於防火墙規則設置所致, 如禁止 UDP 等協議的報文, 或者不回應 ICMP 錯誤報文等. Proberoute 在第 18 個跳點時發現目的主機, 而 Traceroute 在第 18 個跳點以後不能偵測到目的主機.

Fig. 5. Output of Proberoute

```
$ sudo ./proberoute -A -w1 www.ccb.com 80
proberoute to www.ccb.com (219.142.89.78) from
192.168.0.100 (en0) with TCP UDP ICMP protocol
destination: 0.0.0.0, gateway: 192.168.0.1, mask: 0.0.0.0
1 hops min, 30 hops max
outgoing MTU = 1500
 1  192.168.0.1  10.645 ms  0.119 ms  0.170 ms
 2  192.168.0.1  !F  10.741 ms  !F  0.084 ms *
 3  182.93.63.222  11.155 ms
    182.93.63.226  10.856 ms
    182.93.63.222  11.281 ms
 4  182.93.63.221  22.115 ms  0.096 ms  0.164 ms
 5  202.175.54.69  10.530 ms
    202.175.54.77  0.145 ms
    202.175.54.69  11.104 ms
 6  202.97.94.33  21.375 ms  10.260 ms  0.142 ms
 7  202.97.94.101  22.401 ms * *
 8  202.97.94.129  22.193 ms  0.130 ms  11.095 ms
 9  202.97.34.145  64.778 ms  0.110 ms  0.088 ms
10  * * *
11  * * *
12  106.120.254.2  64.089 ms
    219.142.5.46  0.195 ms
    106.120.254.2  10.501 ms
13  219.142.5.46  54.579 ms  0.140 ms  10.977 ms
14  219.142.89.123  64.539 ms  0.151 ms  65.888 ms
15  219.142.89.78  !TTL  52.854 ms
    !TTL  11.102 ms  !TTL  43.382 ms
16  219.142.89.123  64.147 ms  0.107 ms  63.118 ms
17  * * *
18  219.142.89.78  65.639 ms  21.178 ms  42.433 ms
Port 80 open
```

為了簡單起見, 下表僅僅列出沒有探測到的 hop 數以及是否探測到目的主機.[7]

TABLE 1
Proberoute vs Traceroute

Target	Proberoute	Reachable	Traceroute	Reachable
www.pbc.gov.cn	0	Y	1	N
www.boc.cn	4	Y	6	N
www.abchina.com	0	Y	0	Y
www.icbc.com.cn	0	Y	0	Y
www.ccb.com	3	Y	8	N
www.bankcomm.com	0	Y	7	N
www.cmbchina.com	0	Y	2	N

可以發現 Proberoute 除了有部分跳點未能偵測外, 其他跳點全部可以發現, 並且所有目的主

7. 由於路由選路不同, 表中結果可能不可重複.

Fig. 6. Output of Traceroute

```
$ traceroute -n -w1 www.ccb.com
traceroute to www.ccb.com (219.142.89.78),
64 hops max, 52 byte packets
 1  192.168.0.1  4.626 ms  1.437 ms  0.990 ms
 2  * * *
 3  182.93.63.222  7.098 ms
    182.93.63.226  5.361 ms
    182.93.63.222  4.771 ms
 4  182.93.63.225  5.292 ms
    182.93.63.221  8.827 ms
    182.93.63.225  6.773 ms
 5  202.175.54.69  3.858 ms  3.328 ms
    202.175.54.77  3.516 ms
 6  202.97.94.33  12.459 ms  10.884 ms *
 7  202.97.94.93  11.250 ms * *
 8  202.97.94.133  17.648 ms
    202.97.94.121  16.428 ms
    202.97.94.125  14.134 ms
 9  * * 202.97.34.145  55.463 ms
10  * * *
11  * * *
12  106.120.254.2  57.248 ms
    106.120.254.6  56.123 ms
    219.142.5.46  50.885 ms
13  219.142.5.46  51.540 ms *  54.549 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
```

機都可以到達. 而 Traceroute 只有少數目的主機可以到達, 并且跳點有不同程度的丢失.

另外, 幾乎全部的 Traceroute 實現 (包括 Windows 和 GNU/Linux), 以及更為全面的 Nmap 工具, 在判別目的主機是否到達上都有一個不易察覺的失誤, 那就是只對比回應的 IP 是否是目的主機, 而忽略了目的主機也可能發送 ICMP 超時錯誤. 事實上隨著負載均衡器的廣泛使用, 有很多目的主機地址實際是負載均衡器的虛擬 IP, 會經過負載均衡器的路由再派往真正的目的主機, 這樣就會出現路由器和目的主機有相同 IP 的情況, 區別它們的辦法仍然是判斷是否由於 TTL 耗盡而收到路由器發出的 ICMP 超時錯誤信息. 在 Proberoute 的輸出圖 5中, 可以發現第 15 hop 和第 18 hop 的 IP 是相同的, 都是目的主機 IP, 但是第 15 hop 收到了 ICMP 超時錯誤, 說明是路由器而不是目的主機, 因此 Proberoute 繼續探測并于第 18 hop 發現真正的主機.

References

[1] 在 **traceroute** 使用手冊上說, 程序由 Steve Deering 提議, 由 Van Jacobson 實現, 并由許多其他人根據 C.Philip Wood, Tim Seaver 以及 Ken Adelman 等人提出的令人信服的建議或補充意見進行調試. 詳見: https://en.wikipedia.org/wiki/Traceroute

[2] Details of Traceroute for Linux at http://traceroute.sourceforge.net/

[3] Stevens, W. R. 1994. "TCP/IP Illustrated, Vol. 1: The Protocols", Chapter 8, *Traceroute Program*

[4] Lyon, G. 2008. "Nmap Network Scanning", Chapter 10, *Detecting and Subverting Firewalls and Intrusion Detection Systems*

[5] Arkin, O. 2001. "ICMP Usage in Scanning", Section 4.1, *Triggering ICMP Parameter Problem error messages*