A Distributed Tool for VoIP Stress Testing

Chenglin Tsai Quincy Wu
Graduate Institute of Communication Engineering National Chi Nan University
No. 1, University Road, Puli, Nantou 545, Taiwan
{s96325512,solomon}@ncnu.edu.tw

摘要

隨著數位科技的蓬勃發展，網際網路的使用大幅增加，進而發展出各種不同類型的網路應用服務；VoIP(Voice over Internet Protocol)是其中一項受到注目的產業。由於 VoIP 使用者越來越多，進行 VoIP 的壓力測試也成為重要的議題，因此在本文中，我們在 FreeBSD 7.0 的環境下發展出 D-SIPp(Distributed SIPp)程式，在分散式的環境模擬多對一建立通話的動作，並且傳送 RTP(Real-time Transport Protocol) 封包，藉由大量的通話建立，對接收端做壓力測試。

關鍵詞：distributed tools, RTP, SIP, SIPp, stress test, VoIP。

1. 前言

網際網路的蓬勃發展加上通訊科技的日新月異，電話通訊的應用在網際網路也成為一項熱門的產業。VoIP(Voice over Internet Protocol)是一種透過 IP(Internet Protocol)網路來進行語音傳輸的技術，相較於傳統的 PSTN(Public switched telephone network)，具有大幅節省通話及建設成本的特點，因此極具有競爭力。

在 VoIP 的前景一片看好的同時，所衍生出來的問題是，通話過程中是否會產生封包遺失、延遲或是更嚴重的使得通話(session)無法建立的情形。假若有一個機構，他有一台專屬的 VoIP 伺服器，凡是機構內的使用者要與外界進行 VoIP 通話，都必須透過此伺服器。因此假設在同一時間內，此機構有大量的使用者需要與外界進行 VoIP 通話，可能會發生通話品質上的問題。因此，此伺服器是否能保持穩定的運作成為我們所關心的議題。在本文中，我們說明如何發展出一套分散式的網路電話流量測試工具，發送指令到多台測試用的電腦，再從這些測試電腦執行 SIP 指令與 VoIP 伺服器端建立上百通的通話，藉由這些通話產生大量的 RTP 封包，對 VoIP 伺服器進行壓力測試。

2. SIP Overview

Session Initiation Protocol [1]，簡稱 SIP，是由 IETF 工作小組所制定的，在 OSI(Open System Interconnection)七層架構中屬於應用層的控制協定。它是一種藉由網際網路傳送語音及多媒體資料的技術，並且利用純文字的方式來傳送控制信令，內容包括建立、修改和終止多媒體的通話。SIP 的架構簡單、擴展性高，因此成為 VoIP 最主要運用的通訊協定。SIP 是一個主從式(Client-Server)的架構，當 Client 發出請求(Request)，而伺服器收到請求(Request)之後，會回覆適當的回應(Response)給 Client。

2.1 Basic Operations of SIP

在 SIP 的環境中，最主要定義兩個元件，一個為 UA(User Agent)，另一個為代理伺服器(Proxy Server)，下面說明兩者的功能。

User Agent：是一種終端設備，可以是軟體的程式，也可以是硬體的實體電話機。它包含 UAC(User Agent Client)跟 UAS(User Agent Server)兩部份，其中 UAC 只負責產生請求(Request)以及回覆回應(Response)，而 UAS 要負責接收請求(Request)以及產生回應(Response)。

代理伺服器：負責接受使用者或其他代理伺服器所發送的請求(Request)，並且轉送請求(Request)到其他代理伺服器或是接收端。

圖 1 所示為 SIP 建立與終止通話的流程。

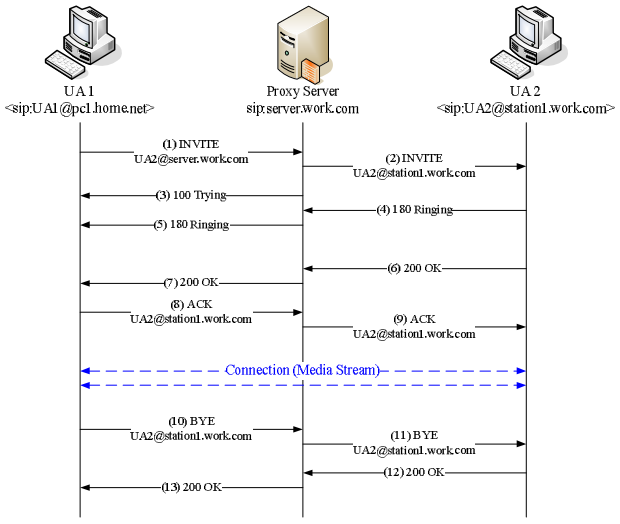


圖 1 SIP 建立與終止通話流程

(1) 首先，UA1 要邀請 UA2 進行通話時，UA1 會先發出 INVITE 請求(Request)給代理伺服器。

(2) 代理伺服器收到 UA1 所發出 INVITE 請求後，會轉送給 UA2。

(3) 代理伺服器會回送一個 100 Trying 訊息給 UA1，同時轉送 INVITE 請求給 UA2。

(4) 當 UA2 收到 INVITE 請求，就會回送一個 180 Ringing 訊息給代理伺服器。

(5) 當代理伺服器收到 180 Ringing 訊息後會轉送給 UA1。

(6) 當 UA2 答應這個邀請時，便會送出一個 200 OK 訊息給代理伺服器。

(7) 當代理伺服器收到此 200 OK 訊息後會轉送給 UA1。

(8) UA1 收到 200 OK 訊息之後，得知對方接受邀請，於是發出 ACK 訊息給代理伺服器。

(9) 代理伺服器收到 ACK 訊息後會轉送給 UA2，當 UA2 收到最後 ACK 的確認後，表示雙方完成呼叫連線，UA1 與 UA2 直接建立通話不需透過代理伺服器。

(10) 若 UA1 要結束通話時，則發出一個 BYE 請求給代理伺服器。

(11) 代理伺服器收到 BYE 請求後轉送給 UA2。

(12) UA2 收到 BYE 請求時，則會發出 200 OK 訊息給代理伺服器。

(13) 代理伺服器收到後轉送給 UA1；當 UA1 收到 200 OK 訊息之候，雙方結束連線。

3. Distributed SIP Pressure Test

當我們要針對伺服器進行壓力測試時，由於伺服器往往採用較昂貴的硬體設備，因此在測試時必須使用更加昂貴的 SmartBits 等特殊設備，才能測出伺服器的承載極限。因此本文中我們提出以多個 UAC 對一個伺服器進行壓力測試的概念，如圖二的架構，首先我們要從 Master 端發送 SIP 指令到 N 台 Slave 端的電腦，使得 Slave 端的電腦能夠與 Receiver 端進行通話，藉此來對 Receiver 進行壓力測試。另外，為了使 D-SIPp 這套軟體運作順利，我們需要運用一些功能，接下來的小節將會一一介紹。

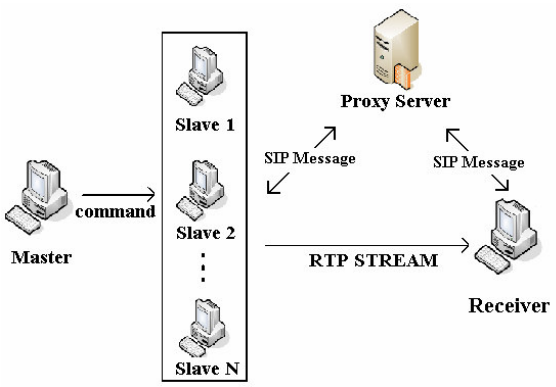


圖 2 D-SIPp 測試架構

3.1 啟動 Slave

當 Master 要傳送 SIP 指令到 N 台 Slave 端電腦時，首先必須要確認 Slave 的電腦是否有開啟，因此我們運用了 fping 這套軟體來幫助我們做快速的判別。

fping 的操作原理如同我們熟知的 ping，它能夠在本機端發送 ICMP(Internet Control Message Protocol)[2]封包到要測試的電腦，測試彼此間的網路是否通暢。不過 fping 比起 ping 在執行上更有效率，原因在於若要測試 N 台主機時，fping 同一時間會送出 N 個 ICMP 的封包到等待測試的 N 台主機。對於沒有回應的電腦，則 fping 會對它們再發出一次 ICMP 封包；在設定的時間內仍然沒有回應，則直接判斷為不可送達的(unreachable)。

其中 fping 裡有一個我們會用的選項為-f，它可以直接偵測一個 txt 檔判斷裡面的 IP 位置是否存活的(alive)。只要將待檢測的 IP 位置依序加入其中，就可以一次偵測上百台電腦；相較於 ping 需要一個一個 IP 位置去測試，顯得有效率的多。

3.2 時間同步

在 D-SIPp 的運作過程中，我們必須使得 N 台 Slave 電腦具備相同的時間，以便同步與 Receiver 建立通話，因此我們運用 OpenNTPD 這套軟體作為對時工具。

我們將在每一台電腦中安裝 OpenNTPD，OpenNTPD 是由 OpenBSD 的開發者自行研發的一套軟體，它能提供一個安全的 NTP(Network Time Protocol)[3]服務。由於 NTP 伺服器以階層式架構形成時間追溯體系，當同一個 LAN 內有多台機器需要時間的同步，若每一台分別去與外界的 stratum 1 伺服器(位於階層最頂層的伺服器，直接追溯到國家標準時間)或是 stratum 2 伺服器(透過 Stratum 1 伺服器間接追溯到國家標準時間)對時，這不僅是一種浪費網路資源的作法，同時也容易因遠距離的通訊造成對時上的誤差。因此我們選擇 LAN 裡的一台機器當作 NTP 伺服器的角色，負責與外面的 stratum 1/2 對時；而 LAN 裡的其他機器就扮演 NTP 客戶端的角色，與這台 NTP 伺服器對時。因此每一台 NTP 客戶端的電腦均為同一個 stratum，因此適合做為時間同步的校正工具。

3.3 Generate Traffic

當我們能確定 Slave 的電腦已啟動且彼此之間的時鐘同步後，這些 Slave 主機就要與 Receiver 同步建立通話，因此我們運用了 SIPp 這一套系統作為通話建立的方式。

首先介紹 SIPp[4]這套軟體。SIPp 是一套開放原碼的測試工具，它針對 SIP protocol 來產生大量流量的工具。它包含了一些基本的 SipStone user agent scenarios，像是扮演 User Agent Client (UAC)

跟 User Agent Client (UAS)的角色，使用 INVITE 跟 BYE 建立和結束通話。SIPp 也可以讀取我們所制定的 XML(Extensible Markup Language) 情境檔案。最具特別的一點，它可以動態的顯示測試過程中的統計數字(call rate, round trip delay, and message statistics)，如圖三與圖四分別為顯示為 UAC 與 UAS 兩端的統計。SIPp 提供眾多參數給使用者自行設定，像是建立通話速率(Call-rate)，可以動態的調整每秒建立通話的數目等相關訊息。

```
---------------------------- Scenario Screen -------- [1-9]: Change Screen --
 Call-rate(length)   Port   Total-time  Total-calls  Remote-host
 10.0(0 ms)/1.000s   5061       5.00 s           49  127.0.0.1:5060(UDP)

 10 new calls during 1.000 s period     1 ms scheduler resolution
 0 calls (limit 30)                     Peak was 1 calls, after 0 s
 0 Running, 49 Paused, 0 Woken up
 0 dead call msg (discarded)            0 out-of-call msg (discarded)
 3 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
     INVITE ---------->           49        0         0
        100 <----------           0         0         0         0
        180 <----------           49        0         0         0
        183 <----------           0         0         0         0
        200 <----------  E-RTD1   49        0         0         0
        ACK ---------->           49        0
      Pause [      0ms]           49                            0
        BYE ---------->           49        0         0
        200 <----------           49        0         0         0

------ [+|-|*|/]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause traffic -----
```

圖三 SIPp for UAC

```
---------------------------- Scenario Screen -------- [1-9]: Change Screen --
 Port   Total-time  Total-calls  Transport
 5060      62.03 s           62  UDP

 0 new calls during 1.000 s period      1 ms scheduler resolution
 0 calls                                Peak was 41 calls, after 5 s
 0 Running, 0 Paused, 0 Woken up
 0 dead call msg (discarded)
 3 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
 ----------> INVITE              62        0         0         0

 <---------- 180                 62        0
 <---------- 200                 62        0         0
 ----------> ACK          E-RTD1 62        0         0         0

 ----------> BYE                 62        0         0         0
 <---------- 200                 62        0
 [   4000ms] Pause               62                            0
---------------------------- Sipp Server Mode -----------------------------
```

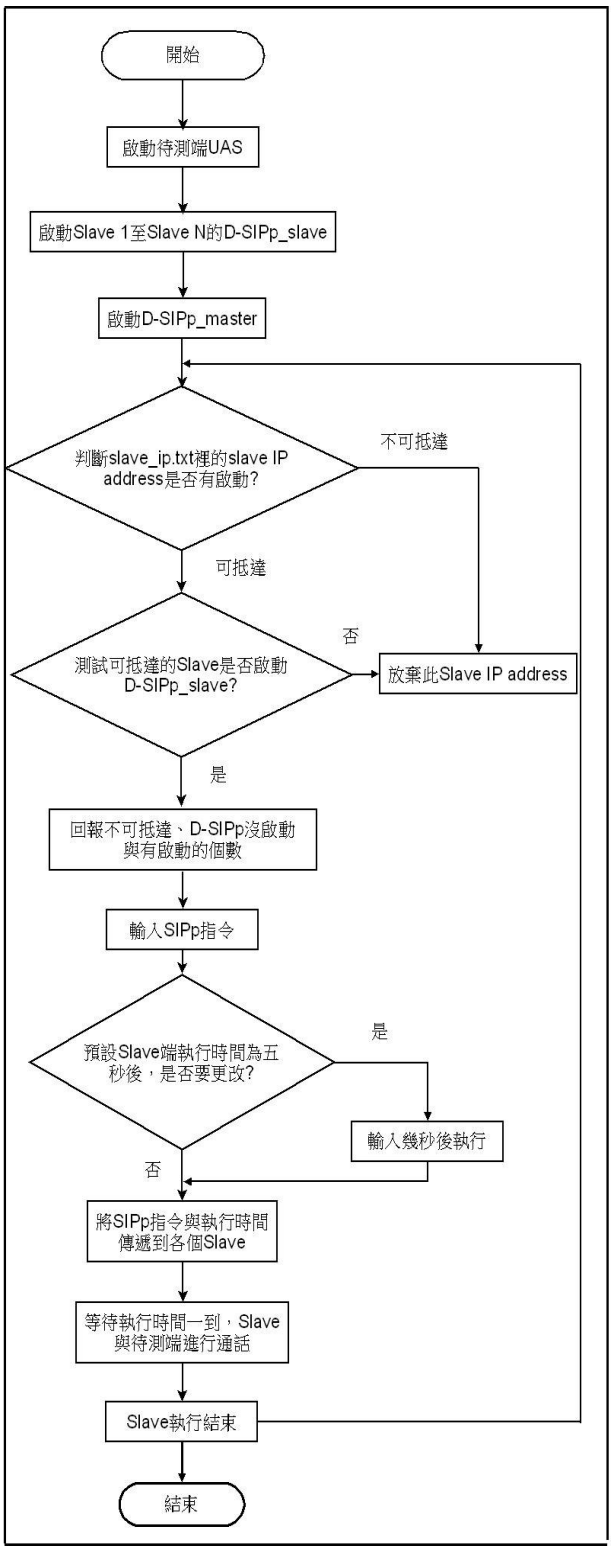
圖四 SIPp for UAS

在 D-SIPp 實作環境中，我們會運用到 SIPp 裡的一個情境，它除了在 UAC 與 UAS 間建立通話之外，還同時傳送內容為 G.711 格式與 DTMF(Dual Tone Multi-Frequency)[5]格式的 RTP[6]語音封包，其中 DTMF 是以特別的訊息(Event)夾帶方式，透過 RTP 封包將 DTMF 等傳統電話的數字鍵音頻訊號遞送至對方接受端。需要注意的是，由於 SIPp 這個工具只會由 UAC 傳送 RTP 封包給 UAS，UAS 並不會傳送 RTP 封包給 UAC，這是與實際的 VoIP 通話稍微不同的地方。

4. D-SIPp (Distributed-SIPp)

在之前的章節已經介紹了 D-SIPp 這套軟體的架構以及一些功能，接下來將介紹 D-SIPp 的流程圖與實驗環境。

圖五為 D-SIPp 整支程式的運作流程，在執行之前，需先將 Slave 1 至 Slave N 的 IP address 依序填入 slave_ip.txt，fping 將會判斷裡面的 Slave 端的主機是否有啟動。



圖五 D-SIPp 運作流程

圖六跟圖七分別為 Master 端的 D-SIPp_master 的執行畫面與其中一個 Slave 端的 D-SIPp_slave 的執行畫面。

D-SIPp_master 的執行步驟如下：

(1) 開始執行 D-SIPp_master 程式，藉由 fping 判斷 10.10.59.144 至 10.10.59.148 的 Slave 是否有啟動，並且回報沒啟動的 IP address。

(2) 判斷已經啟動的 Slave 是否有執行 D-SIPp_slave 程式。

(3) 分別回報不在線上的 Slave IP 位置與沒有執行 D-SIPp_slave 程式的 Slave IP 位置，與已在線上並啟動 D-SIPp_slave 程式準備接受指令的 Slave IP 位置。

(4) 輸入 Slave 要與待測端建立通話的 SIPp 指令

(5) 選擇預設五秒後 Slave 執行指令或是自行輸入時間；要自行輸入請按 y，並輸入自行預定的時間。

(6) 鍵入時間後，顯示當下的時間及 Slave 群預定執行 SIPp 指令的時間。

(7) 分別顯示各個 Slave 執行的指令，並結束 D-SIPp_master 的程式。

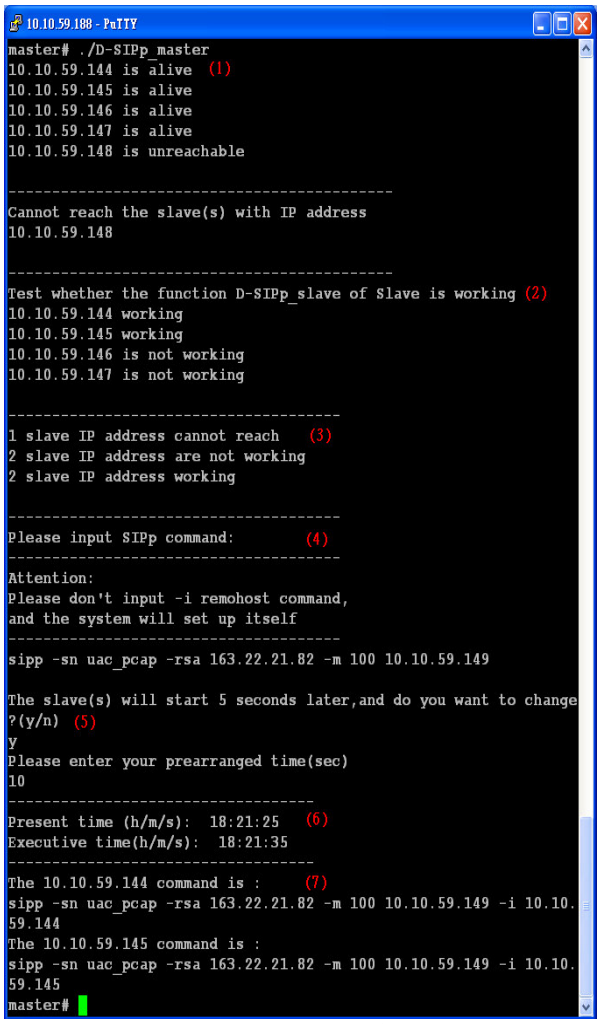
D-SIPp_slave 的執行步驟如下：

(1) 首先執行 D-SIPp_slave 程式，Slave 等待與 Master 端建立連線。

(2) 當 Master 將 SIPp 指令送達後，就會顯示 Master 的 IP address、SIPp 指令、及預定執行的時間。

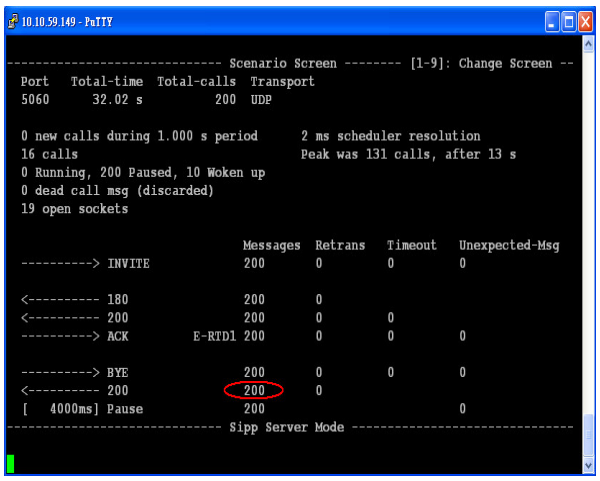
執行時間一到，各個 Slave 端的 UAC 同步與待測端的 UAS 進行通話；全部的通話建立完畢後，Slave 端則會回到(1)重新等待 Master 傳送新的 SIPp 指令。

圖八為待測端的 UAS 的執行結果，其中 10.10.59.144 與 10.10.59.145 兩個 IP 位置，以每秒 10 通的通話速率分別與 UAS 建立 100 通通話；由於 UAS 還能順利的和兩個 UAC 建立通話，因此可以看到同意結束通話 200 OK 的訊息是兩百通。假若 Slave 端每秒傳送的通話數逐漸提升，就會從 UAS 看到有遺失通話數的情形，藉此進行待測端的壓力測試。



圖六 D-SIPp_master 實作介面

圖七 D-SIPp_slave 實作介面



圖八 Receiver 執行結果

5. Conclusions and Future work

在本篇論文中我們提出一個分散式的架構，在 Master 執行 D-SIPp_master 的程式，發出指令給上百台正在執行 D-SIPp_slave 的 Slave 端。等到預定執行的時間一到，各個 Slave 端的 UAC 同時與 UAS 進行通話建立，並且模擬實際通話所發出的 RTP 封包，藉此進行伺服器端的壓力測試。

目前實作的部分僅依據通話成功建立的數目作為 Sever 端的壓力測試，除了通話數目之外，另一個值得參考的數值則是有關語音品質 MOS (Mean Opinion Score)值。在未來的研究主題上，將會對已經建立通話的通話，進行封包遺失與單向延遲的測量，套用到 E-model [7]並且轉換成 MOS 值，以供管理者更精確地評估 VoIP 通話的品質。另外可以考慮建置以 GPS 對時的 NTP 伺服器，由於 GPS 是一個可信賴的標準時間並且為 stratum 1，因此可以提升 LAN 裡主機，也就是各個 Slave 端提升到 stratum 2，使得同步的時間更加準確，也間接提升 Slave 端同步執行 SIPp 時的一致性。

參考文獻

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, “SIP：Session Initiation Protocol”, Network Working Group, RFC3261, IETF, June 2002.

[2] J. Postel, “Internet Control Message Protocol” Network Working Group, RFC 792, IETF, September 1981

[3] David L. Mills, “Network Time Protocol (Version 3) Specification, Implementation and Analysis” Network Working Group, RFC 1305, IETF, March 1992

[4] SIPp [http://sipp.sourceforge.net/]

[5] H. Schulzrinne, S. Petrack, “RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals”, RFC 2833, IETF May 2000

[6] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “ RTP: A Transport Protocol for Real-Time Applications”, Network WORKING Group, RFC 1889, IETF, January 1996.

[7] ITU-T “The E-model, a computational model for use in transmission planning” ,ITU-T Recommendation G107, March 2005