NS-2 下的 802.3 乙太網路模擬

程榮祥
http://teachers.ksu.edu.tw/rscheng/

TCL Script 的寫法

在寫模擬程式時，基本上都會先用" new Simulator "這個 Key Word 產生一個模擬環境，接著產生 Node 並建立 Topology，然後就可以在這個模擬環境下產生 Traffic，開始進行模擬了。一個簡單的 TCL 程式架構如下：

```
# 產生模擬物件
set ns [new Simulator]

# 定義結束處理程序
proc finish {} {
}
# 主程式開始

# 主程式結束
# 設定"$ns"這個物件在 at 後所指定的"$end"時間執行 finish 這個副程式
$ns at $end "finish"
$ns run
```

在這個例子中，我們用 set ns 去產生一個名為 ns 的變數，並將 new Simulator 所產生的物件指向 ns 這個變數。在 NS-2 中，變數在使用前都要加上"$"符號，#符號則是用來表示後面的文字是註解。

模擬架構

Figure 1 是我們所使用的網路架構圖，這個網路中有兩個 802.3 的區域網路，其中 Edr_0、Src_0、Src_1 到 Scr_n 都屬於 LAN 0，而 Edr_1 以及 Dst_0 則屬於 LAN 1。在這個例子中，我們在 $Src_{\{0, 1, ...n\}}$ 與 Dst_0 之間建一個 TCP 連線，並使用 FTP 在這些 TCP 連線上傳送資料。

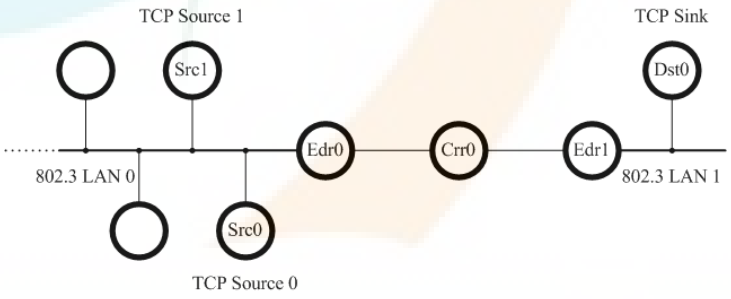

Figure 1. 模擬架構圖

TCL 程式碼

這個 TCL Script 在執行時可以接受 4 個外部參數，分別是 TCP 的版本，模擬的結束時間，LAN 0 的 Node 個數，以及從 Edr 0 到 Edr 1 之間的網路頻寬。在模擬的時間到達時，程式會列印在這段時間內的 TCP 執行的效能 (Goodput) 並結束模擬。

```
# 定義讀取使用者設定參數的子程序
proc getopt {argc argv} {
       global opt
    for {set i 0} {$i < $argc} {incr i} {
```

```
            set opt($i) [lindex $argv $i]
        }
}

# 定義結束處理的子程序
proc finish {tcp_ tstart_ tend_} {
   global ns node end
   upvar $tcp_ tcp

   # 計算 TCP 的傳輸效能
   set now [$ns now]
   set total_thr 0
   for {set i 0} {$i < $node} {incr i} {
            set thr($i) [expr [$tcp($i) set ack_]/$end/1000000.0*[$tcp($i) set packetSize_]*8]
            puts [format "goodput(%d) = %f Mbps" $i $thr($i)]
        }

   #Close the trace file
       $ns flush-trace
   puts " "          ;# 印一個空白字元並將游標移至下一行
   exit 0
}

# 讀取程式外部參數
getopt $argc $argv
set tcp_src       $opt(0)          ;# TCP 版本
set end           $opt(1)          ;# 模擬結束時間
set node          $opt(2)          ;# TCP Source 的個數
set bandwidth   $opt(3)          ;# Bottleneck 的頻寬
set psize   512                    ;# 設定封包大小為 512 bytes

# 模擬程式的執行範例
# ns 802_3.tcl reno 1 2 45

# 主程式開始
# 產生模擬物件
set ns [new Simulator]
ns-random 0

# 開啟 trace 記錄檔
ns trace-all [open out.tr w]

# 開啟 NAM 記錄檔
set nf [open out.nam w]
$ns namtrace-all $nf

set edr(0) [$ns node]              ;# Edge router 0
lappend lanlist(0) $edr(0)
for {set nid 0} {$nid < $node} {incr nid} {
       set src($nid) [$ns node]
       puts "node $nid"
       lappend lanlist(0) $src($nid)
}

set edr(1) [$ns node]              ;# Edge router 1
set dst(0) [$ns node]              ;# Destination
lappend lanlist(1) $edr(1)
lappend lanlist(1) $dst(0)
```

有關 Node 的產生方式，我個人的建議是，先產生 802.3 的 Node，接著才去產生一般的 Node，這樣比較不會出現 Node ID 和 Node IP 不一致的情形。

```
# 產生 core router 並把 core router 和 edger 連結起來
set crr(0) [$ns node]
$ns duplex-link $edr(0) $crr(0) $bandwidth+Mb 5ms DropTail
$ns queue-limit $edr(0) $crr(0) 50                                    ;# 設定 buffer size
$ns duplex-link $crr(0) $edr(1) $bandwidth+Mb 5ms DropTail
$ns queue-limit $crr(0) $edr(1) 50

# 產 802.3 的區域網路
set lan(0) [$ns newLan $lanlist(0) 100Mb 1ms -macType Mac/802_3]
set lan(1) [$ns newLan $lanlist(1) 100Mb 1ms -macType Mac/802_3]

# 產生 TCP 連線(可指定 TCP 版本)
for {set i 0} {$i < $node} {incr i} {
      if {$tcp_src == "reno"} {set tcp($i) [$ns create-connection TCP/Reno $src($i) TCPSink $dst(0) $i]
      } elseif {$tcp_src == "newreno"} {set tcp($i) [$ns create-connection TCP/Newreno $src($i) TCPSink $dst(0)
$i]
      } elseif {$tcp_src == "sack"} {set tcp($i) [$ns create-connection TCP/Sack1 $src($i) TCPSink/Sack1 $dst(0)
$i]
      } elseif {$tcp_src == "tahoe"} {set   tcp($i) [$ns create-connection TCP $src($i) TCPSink $dst(0) $i]]
      } else {puts "This tcp version does not exist."}

      # 設定 TCP 參數
      $tcp($i) set window_ 128
      $tcp($i) set packetSize_ $psize

      # 設定 FTP 連線
      set ftp($i) [ $tcp($i) attach-app FTP]
      $ns at 0          "$ftp($i) start"
      $ns at $end       "$ftp($i) stop"
}

# 設定在$end 這個參數所指定的時間執行 finish 這個副程式
$ns at $end "finish tcp "

# 開始進行模擬
$ns run
```

底下是執行模擬的範例($node = 2, 3, and 4)：

```
$ ns 802_3.tcl reno 1 2 45
node 0
node 1
goodput(0) = 1.941504 Mbps
goodput(1) = 1.974272 Mbps

$ ns 802_3.tcl reno 1 3 45
node 0
node 1
node 2
goodput(0) = 1.593344 Mbps
goodput(1) = 1.884160 Mbps
goodput(2) = 1.011712 Mbps

$ ns 802_3.tcl reno 1 4 45
goodput(0) = 1.966080 Mbps
goodput(1) = 2.002944 Mbps
```

```
goodput(2) = 2.596864 Mbps
goodput(3) = 1.110016 Mbps
```