NS-2 下的 TCP over IEEE 802.11WLAN 模擬

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

網際網路所提供的傳輸服務

目前網際網路所提供的傳輸服務，就服務的性質而言可分成兩種，一種是(1) 連接導向的可靠性傳輸服務 (Connection oriented reliable service)，這個服務主要由 TCP (Transmission Control Protocol [1]) 來提供，另一種則是(2) 非連接的不可靠性傳輸服務 (Connection-less, unreliable data transfer)，這部份則是由 UDP (User Datagram Protocol)來提供。

TCP 提供的服務

1. 可靠的資料傳送(Reliable data transfer)服務：TCP 藉由回應 (Acknowledge) 和重送的機制提供可靠性的服務。可靠的資料傳送意味著應用程式可依賴此服務，該服務可確保資料能依照順序地被傳送接收，不會有錯誤發生。
2. 流量控制(Flow control)：確保連線的兩端不會因為太快傳送過量的封包而淹沒了另一端。
3. 擁塞控制(Congestion control)：當路由器發生擁塞時，封包容易因為緩衝區溢滿而被丟棄，擁塞控制可以避免傳送端傳送太多資料到網路上造成網路擁塞。

UDP 提供的服務

TCP 與 UDP 都是透過 IP 封包來傳輸資料，在開始傳送資料之前，TCP 的 Client 行程會與 Server 行程執行建立連線的動作，而 UDP 則不需進行此一動作。就如前面所述，目前網際網路提供給應用程式的服務只有連接導向的可靠性服務與非連接的不可靠性服務，而不管是 TCP 或 UDP，基本上都沒有提供對與頻寬與時間延遲的最低保證。

網路協定和分層的概念

在瞭解了網路可能提供的服務後，接著我們就來看看目前網路所使用的分層方式，一般稱之為「網路通訊協定分層堆疊」(Internet protocol stack)。目前網際網路所使用的分層架構由上而下分別是：應用層、傳輸層、網路層、連結層以及實體層，每一層都有其各自使用的通訊協定。

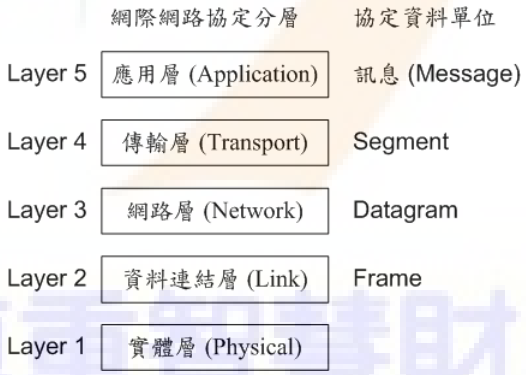

Figure 1 通訊協定堆疊與各層封包所使用的單位名稱

在習慣上，一般我們都以 Figure 1 的方式來表示，並且為了方便區別使用，各層的資料單位 (封包) 也給予不同的名稱表示。以下我們簡單地瀏覽一下各層所使用的通訊協定以及通訊協定所使用的封包格式。

傳輸層的主要功能

對於應用層而言，傳輸層主要的功能就是在應用程式之間提供邏輯的連線。所謂的邏輯連線是指在應用程式的傳送端與接收端之間其實並沒有一條實際的連線，但是透過傳輸層所提供的功能，應用程式可以假設傳送端與接收端之

間有一條連線存在，應用程式可以透過傳輸層協定將資料送出。

目前傳輸層所使用的協定主要有兩種－TCP 和 UDP，應用程式的通訊過程可用下面簡單的圖例來表示：

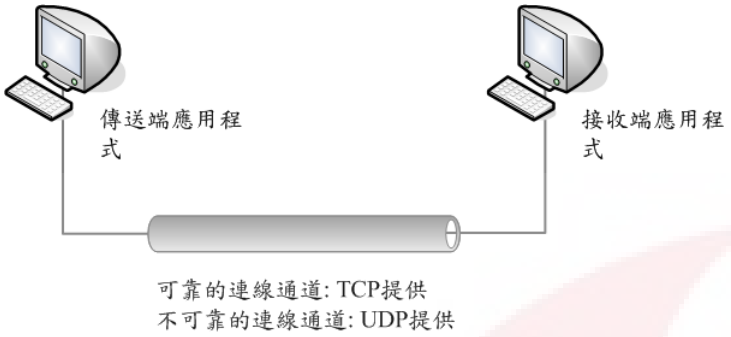

Figure 2：End-to-End 的服務模型

目前傳輸層所使用的協定主要有兩種－TCP 和 UDP，以下簡單地介紹 TCP 和 UDP 的封包格式。

TCP 的應用範圍很廣泛，如同我們在前一個章節所提到的，許多我們在日常生活中經常使用網路應用程式，底層所使用的傳輸協定都是使用 TCP，例如網頁瀏覽(Web)、電子郵件(E-mail: POP3，SMTP)、遠端終端機存取(Telnet)、檔案傳輸(FTP)等。相較於 TCP，UDP 的訊息格式較為簡單，相對地可提供的使用的功能就較少。UDP 的存在是為了滿足一些需要快速處理與反應的應用程式，因此較適合用在多媒體等對時間較為敏感或不需要擁塞控制的服務上。

Ns-2 的參數預設值存放在 ns-default.tcl 這個檔案中。這個檔案的存放位置是在 ns-allinone-2.XX/ns-2.XX/tcl/lib/ 這個目錄下，欲知 Ns-2 的預設值設定，可以將這個檔案叫出來看看：

```
$ cd ns-allinone-2.31/ns-2.31/tcl/lib/
$ cat ns-default.tcl | more
```

模擬參數設定

在 Ns-2 中，Node 可以用來表示一個主機(Host)、路由器(Router)或交換器(Switch)。Node 產生的方法如下($ns 為 Ns-2 產生的模擬物件)：

```
set router [$ns node]
```

Node 產生後，接下來就可以開始產生網路的拓撲，有關 Node 之間的 Link、Bandwidth、Propagation Delay 以及 Queue-Type 設定方法，指令格式如下：

```
$ns duplex-link <node1> <node2> <bandwidth> <delay> <queue-type>
```

網路的拓撲建好後，接下來就是指定要使用的 Agent，並告訴這個 Agent 在產生 Traffic 的時候，要使用那一種 Traffic generator，指令格式如下：：

```
set Agent_name [new Agent/<Agent-type>]
```

接下來我們用一個簡單的例子來介紹如何在 Node 與 Node 之間建一個 TCP Connection (如 Figure 3 所示)

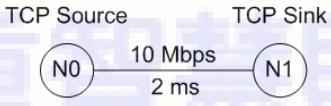

Figure 3：簡單的網路架構圖(2 個 Node 之間有一個 TCP Connection)

以下的 TCL Script 是根據 Figure 3 的設定所撰寫的：

Example 1：

```
set ns [new Simulator]      ;#Create a simulator object
set nf [open out.nam w]     ;#Open the Nam trace file
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
```

```
        global ns nf
        $ns flush-trace
        close $nf           ;#Close the trace file
        exec nam out.nam &
        exit 0
}

set node1 [$ns node]        ;#Create two nodes, then create a duplex link between the nodes
set node2 [$ns node]
$ns duplex-link $node1 $node2 10Mb 2ms DropTail

#Connect the TCP source with the TCP sink
set tcp [$ns create-connection TCP $node1 TCPSink $node2 1]
$tcp   set window_ 128     ;# Configure the TCP agent
set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns at 0.5 "$ftp start"       ;# Schedule events for the FTP agent
$ns at 4.5 "$ftp stop"
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Ns-2 是個很成熟的模擬環境，對 TCP/IP 網路通訊協定的支援也相當地豐富，幾乎所有的 TCP 實作版本 Ns-2 都有支援，底下我們列出一些 Ns-2 有支援的 TCP Agent 供參考(僅列出比較常見的版本)：

Table 1：TCP agent types in Ns-2 [1]

Agent Type	**Description**
TCP	A TCP Tahoe sender
TCP/Reno	A TCP Reno sender
TCP/Newreno	A TCP New Reno sender
TCP/Vegas	A TCP Vegas sender
TCP/SACK1	A TCP sender with selective acknowledgments
TCPSink	A TCP receiver that sends one ACK per packet
TCPSink/DelAck	A TCP receiver with a configurable delay for ever ACK
TCPSink/Sack1	A TCP receiver with selective ACKs
TCPSink/Sack1/DelAck	A TCP receiver with Sack1 and DelAck

TCP Agent 產生後，就可以針對這個 TCP Agent 作一些進一步的設定，底下我們列出在做 TCP 的模擬實驗時，比較常見的一些相關參數：

Table 2：TCP 傳送端(Source)的設定參數

參數	說明
fid_	TCP 連線的號碼 (Flow ID)。
window_	Advertised window 的上限值 (upper bond)。Advertised window 有時也稱為 receive window，其實指的就是接收端的緩衝區可以容納的封包個數。因此當 Congestion window 的值超過 Advertised window 時，TCP 的傳送端會執行流量控制以避免送的太快而導致接收端的緩衝區溢滿 (overflow)。
maxcwnd_	Congestion window 的最大值。
windowInit	Congestion window 的初始值
ecn_	True 表示使用 ECN。
packetSize_	傳送封包的資料大小。
tcpTick_	TCP 的 timeout 時間 (預設值為 0.01 秒)
maxburst_	每收到一個 ACK 觸發傳送端最多能送出的封包個數。0 表示關閉此選項功能。

Table 3：TCP 接收端(Sink)的設定參數

參數	說明
packetSize_	ACK 封包的資料大小。

模擬架構圖

下面我們用個 TCP over 802.11 Wireless LAN (WLAN)例子來介紹如何在 Ns-2 中進行 TCP 的模擬實驗。如 Figure 3 所示，在 TCP Source (W_0)和 TCP Sink (W_1)之間有一個 TCP 的 Connection，其中 W_0 和 W_1 代表網路上的 Wired Node，AP 代表 Access Point，N_i (i = 1~ n)則是 WLAN 裡面的 Wireless Station。由於 Goodput 的值直接反映 Application Layer 實際上得到的傳輸效能，所以在接下來的例子裡，我們選擇以 Goodput 作為評估 TCP 效能的一個參考依據。

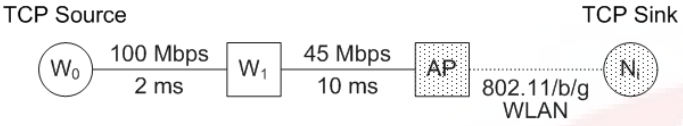

Figure 3. WLAN 模擬架構圖

模擬劇本(TCL Script)

以下的 TCL Script 是根據 Figure 3 的設定所撰寫的：

Example 2：

```
# ======================================================================
# Define options
# ======================================================================
set opt(chan)    [new Channel/WirelessChannel]          ;# Channel type
set opt(prop)          Propagation/TwoRayGround         ;# Radio-propagation model
set opt(netif)         Phy/WirelessPhy                  ;# Network interface type
set opt(mac)           Mac/802_11                       ;# MAC type
set opt(ifq)           Queue/DropTail/PriQueue          ;# Interface queue type
set opt(ll)            LL                               ;# Link layer type
set opt(ant)           Antenna/OmniAntenna              ;# Antenna model
set opt(ifqlen)        50                               ;# Max packet in ifq
set opt(nn)            1                                ;# Number of mobilenodes
set opt(adhocRouting)    DSDV                           ;# Routing protocol

set opt(x)             500                              ;# x coordinate of topology
set opt(y)             500                              ;# y coordinate of topology
set opt(seed)          0                                ;# Seed for random number gen.
set opt(stop)          60                               ;# 設定結束時間
set opt(ftp0-start)    0.0                              ;# 設定 FTP 的開始時間

set num_wired_nodes      2                              ;# 設定 Wireless node 的個數
set num_ap_nodes         1                              ;# 設定 AP (Access Point)的個數

#Suggestion by http://web.syr.edu/~dchen02/FAQ.txt
Phy/WirelessPhy set freq_ 2.4e+9                        ;# frequency is 2.4 GHz
Phy/WirelessPhy set Pt_ 3.3962527e-2                    ;# transmit power
Phy/WirelessPhy set RXThresh_ 2.1003e-09                ;# Receive sensitivity, 40 meters.
Phy/WirelessPhy set CSThresh_ 4.14873e-10               ;# 90 meters
Mac/802_11 set RTSThreshold_ 3000                       ;# < 256 bytes, no RTS/CTS

set datarate 11
if {$datarate == "2"} { puts "FHSS (IEEE802.11)"
      Mac/802_11 set SlotTime_            0.000050      ;# 50 us
      Mac/802_11 set SIFS_                0.000028      ;# 28 us
      Mac/802_11 set PreambleLength_      0             ;# no preamble
      Mac/802_11 set PLCPHeaderLength_ 128              ;# 128 bits
      Mac/802_11 set PLCPDataRate_        1.0e6         ;# 1 Mbps
      Mac/802_11 set dataRate_            2.0e6         ;# 2 Mbps
      Mac/802_11 set basicRate_           1.0e6         ;# 1 Mbps
} elseif {$datarate == "11"} { puts "DSSS (IEEE802.11b)"
      Mac/802_11 set SlotTime_            0.000020      ;# 20 us
      Mac/802_11 set SIFS_                0.000010      ;# 10 us
      Mac/802_11 set PreambleLength_      144           ;# 144 bit
```

```
	Mac/802_11 set PLCPHeaderLength_ 48		;# 48 bits
	Mac/802_11 set PLCPDataRate_	1.0e6		;# 1 Mbps
	Mac/802_11 set dataRate_		11.0e6		;# 11 Mbps
	Mac/802_11 set basicRate_		1.0e6		;# 1 Mbps
} elseif {$datarate == "54"} { puts "DSSS (IEEE802.11g)"
	Mac/802_11 set SlotTime_		0.000009	;# 9 us
	Mac/802_11 set SIFS_			0.000016	;# 16 us
	Mac/802_11 set PreambleLength_	96		;# 96 bit
	Mac/802_11 set PLCPHeaderLength_ 40		;# 40 bits
	Mac/802_11 set PLCPDataRate_	6.0e6		;# 6 Mbps
	Mac/802_11 set dataRate_		54.0e6		;# 54 Mbps
	Mac/802_11 set basicRate_		1.0e6		;# 1 Mbps
} else {
	puts "Error datarate configuration."
}
# ============================================================================
# 副程式
proc finish {} {
	global ns_ tracefd namtrace tcp0 datarate

	set now [$ns_ now]
	set goodput [expr [$tcp0 set ack_]*[$tcp0 set packetSize_]*8/$now/1000000.0] #; 計算 TCP 的 Goodput
	puts [format "goodput = %.2f Mbps (%.2f %s)" $goodput [expr $goodput/$datarate*100] %]
	puts ""
	close $tracefd
	close $namtrace
}
# 讀入外部參數
proc getopt {argc argv} {
	global opt
	lappend optlist nn
		for {set i 0} {$i < $argc} {incr i} {
		set opt($i) [lindex $argv $i]
	}
}
getopt $argc $argv

#Example: ns tcp80211-wlan.tcl 54
set datarate	$opt(0)
# ============================================================================

# 產生模擬物件
set ns_	[new Simulator]
puts "Seeding Random number generator with $opt(seed), Random [ns-random 0]"

# 使用 Hierarchial addressing 的方式定址 (2 個 domain，第 1 個 domain 有 2 個 cluster，第 2 個 domain 有 1 個
# cluster。每個 cluster 中的 node 個數分別為 1、1、$num_ap_nodes +$opt(nn))
$ns_ node-config -addressType hierarchical
AddrParams set domain_num_ 2				;# number of domains
lappend cluster_num 2 1					;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 [expr $num_ap_nodes +$opt(nn)]	;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain

set tracefd	[open wireless-out.tr w]
set namtrace [open wireless-out.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo	[new Topography]
```

```
# Define topology
$topo load_flatgrid $opt(x) $opt(y)

# Create God
create-god [expr $opt(nn) + $num_ap_nodes]

# 建立 Wired node (先設定 Wired node，接著再設定 Wireless node，一定要依此步驟進行)
set temp {0.0.0 0.1.0}                                  ;# hierarchical addresses for wired domain
for {set i 0} {$i < $num_wired_nodes} {incr i} {
     set W($i) [$ns_ node [lindex $temp $i]]
}

# Configure for base-station node
$ns_ node-config      -adhocRouting $opt(adhocRouting) \
                      -llType $opt(ll) \
                      -macType $opt(mac) \
                      -ifqType $opt(ifq) \
                      -ifqLen $opt(ifqlen) \
                      -antType $opt(ant) \
                      -propType $opt(prop) \
                      -phyType $opt(netif) \
                      -channel $opt(chan) \
                      -topoInstance $topo \
                      -wiredRouting ON \
                      -agentTrace ON \
                      -mobileIP ON \
                      -routerTrace OFF \
                      -macTrace OFF

# 設定 AP
set temp {1.0.0 1.0.1}                                  ;# hier address to be used for wireless domain
set AP(0) [$ns_ node [lindex $temp 0]]
$AP(0) random-motion 0                                  ;# disable random motion

# 設定 AP 的位置
$AP(0) set X_ 0.0
$AP(0) set Y_ 0.0
$AP(0) set Z_ 0.0

# 設定 mobile node
$ns_ node-config -wiredRouting OFF

   for {set j 0} {$j < $opt(nn)} {incr j} {
       set node_($j) [ $ns_ node [lindex $temp [expr $j+1]] ]
       $node_($j) base-station [AddrParams addr2id [$AP(0) node-addr]]

       # 設定 Mobile node 的位置
       $node_($j) set X_ [expr 30.0 + $j * 30.0]
       $node_($j) set Y_ 0.0
       $node_($j) set Z_ 0.0
}

# 設定 Wired node 和 AP 之間的連線
$ns_ duplex-link $W(0) $W(1) 100Mb 2ms DropTail
$ns_ duplex-link $W(1) $AP(0) 45Mb 10ms DropTail
$ns_ queue-limit $W(0) $W(1) 64
$ns_ queue-limit $W(1) $AP(0) 64

# 設定 TCP 連線
set tcp0 [new Agent/TCP/Reno]             ;# 指定 TCP 版本為 TCP Reno
set sink0 [new Agent/TCPSink]
$ns_ attach-agent $W(0) $tcp0
```

```
$ns_ attach-agent $node_(0) $sink0
$ns_ connect $tcp0 $sink0                    ;# 建立 End-to-End 的連線
$tcp0 set window_ 128                        ;# Initial window size 為 128 packets
$tcp0 set packetSize_ 1400                   ;# Packet size 為 1400 bytes
$tcp0 set maxburst_ 2                        ;# 指定收到 1 個 ACK 最多能連續送出的封包個數是 2 個 packet

set ftp0 [new Application/FTP]               ;# 指定 application layer traffic generator
$ftp0 attach-agent $tcp0
$ns_ at $opt(ftp0-start) "$ftp0 start"       ;# 排定產生 FTP traffic 的開始時間

# Define initial node position in nam
for {set i 0} {$i < $opt(nn)} {incr i} {
      # 20 defines the node size in nam, must adjust it according to your scenario
      # The function must be called after mobility model is defined
      $ns_ initial_node_pos $node_($i) 20
}

# 排定執行的時間
# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
      $ns_ at $opt(stop).0 "$node_($i) reset";
}
$ns_ at $opt(stop).0 "$AP(0) reset";
$ns_ at $opt(stop).0 "finish"
$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"

puts "Starting Simulation..."
$ns_ run
```

一般而言，評估 TCP 效能的方法就是去觀察 FTP 的 Goodput(應用層實際接收到的資料傳輸速率)、Sequence number，或者是觀察 TCP Congestion window 的變化。在這個例子中，我們使用 Goodput 作為評估 TCP 的效能的依據。在這個例子中，每次模擬結束時，NS-2 會執行 finish {}這個自訂程序並將 TCP 的 Goodput 列印在螢幕上，如下所示：

```
proc finish {} {
      …
      set goodput [expr [$tcp0 set ack_]*[$tcp0 set packetSize_]*8/$now/1000000.0] #; 計算 TCP 的 Goodput
      puts [format "goodput = %.2f Mbps (%.2f %s)" $goodput [expr $goodput/$datarate*100] %]
      …
}
```

```
[chengrs@nsda wlan]$ ns wireless2.tcl 11 20
DSSS (IEEE802.11b)
Seeding Random number generator with 0, Random -1544719163
num_nodes is set 2
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,   distCST_ = 142.7
SORTING LISTS ...DONE!
goodput = 3.83 Mbps / 34.83 %

NS EXITING...
```


底下列出幾個常見的 TCP 狀態參數供參考：

Table 5：TCP 傳送端的狀態參數

參數	說明
window_	The upper bound on the advertised window for the TCP connection.
Maxcwnd_	The upper bound on the congestion window for the TCP connection. Set to zero to ignore. (This is the default)
packetSize_	The size in bytes to use for all packets from this source.

t_seqno_	目前傳送的封包序號。
ssthresh_	Slow-start threshold 的值。
cwnd_	Congestion window 的值。
ack_	傳送端最後收到的 ACK 序號。
maxseq_	傳送端最後送出的封包序號。

如果說要重複進行這個模擬程序，在 Linux 的環境中，我們可以將想要重複執行的工作寫在一個 Script 檔中，接著將這個 Script 檔交給 Linux 去執行 (科技始終來自於人性…^_^)；底下是我們撰寫的一個 Script 檔的範例(example2.tcl 是 TCL Script 的檔案名稱)：

```
rm -f out*
rm -f rawdata*
# 設定 for 迴圈的執行次數
for jth in 1 2 3 4 5
do
	echo -run $jth-
	ns wireless2.tcl 54 60 >> rawdata
done
# 將 simulation 得到的結果交由 awk 分析(計算平均的 goodput)
awk 'BEGIN {goodput_ = 0; achieved_ = 0; ig = 0; ia = 0;}
{
	if ($1 == "goodput" && $3 > 0) {
		goodput_ += $3;
		ig++;
		achieved_ += $6
	}
}
END { m_goodput_ = goodput_/ig;	m_achieved_ = achieved_/ig; printf("Average(%d): good = %.2f Mbps,
achieve %.2f \n", ig, m_goodput_, m_achieved_); printf "";}' rawdata
```

根據我們的設定，這個 Script 每次會執行 TCL script 程式 5 次，隨後將平均的 Goodput (Mbps)計算出來，執行結果如下：

```
[chengrs@nsda wlan]$ ./run_tcp
-run 1-
-run 2-
warning: Route to base_stn not known: dropping pkt
-run 3-
-run 4-
-run 5-
warning: Route to base_stn not known: dropping pkt
Average(5): good = 11.29 Mbps, achieve 20.90 %
[chengrs@nsda wlan]$
```

在 NS-2 中，Wireless Node 的傳輸距離可藉由調整 Pt_與 RXThresh_的值來估算，若想變更這兩個參數的設定，在 NS-2 中已經有現成的範例，只要將範例程式編譯過後就可以直接使用了，請依下面介紹的方法進行：(1)請開啟終端機視窗(命令列文字模式)，將工作目錄切換至~ns/indep-utils/propagation/目錄下，接著執行下列指令($為 linux 提示符號):

```
$ cd ns/indep-utils/propagation/
$ g++ -lm threshold.cc -o threshold
```

假設我們想將 Wireless node 的有效傳輸距離設為 40 公尺，若無線網路卡的傳送頻率為 2.4 GHz，Transmit Power 為 0.0339625，(2)下面的例子示範如何取得 Receiving threshold (RXThresh_)的設定值($為 linux 提示符號)：

```
$ threshold -m TwoRayGround -Pt 3.3962527e-2 -fr 2.4e+9 40
distance = 40
propagation model: TwoRayGround

Selected parameters:
transmit power: 0.0339625
frequency: 2.4e+09
transmit antenna gain: 1
```

```
receive antenna gain: 1
system loss: 1
transmit antenna height: 1.5
receive antenna height: 1.5

Receiving threshold RXThresh_ is: 2.1003e-09
```

接下來只要在 TCL Script 檔的開頭，也就是在主程式開始之前，將 RXThresh_的值設定好就行了，例如：

```
Phy/WirelessPhy set RXThresh_ 2.1003e-09                ;# Receive sensitivity, 40 meters.
```

Table 6 列出 802.11 的相關參數及其設定供參考：

Table 6. Parameters for MAC and PHY Layer.

	802.11	**802.11b**	**802.11g**
SLOT	50 μsec	20 μsec	9 μsec
SIFS	28 μsec	10 μsec	10 μsec
DIFS	128 μsec	50 μsec	28 μsec
PHY_{hdr}	128 bits	192 bits	192 bits
CW_{min}	32	32	32
CW_{max}	1024	1024	1024
RTS	160 bits	160 bits	160 bits
CTS	112 bits	112 bits	112 bits
ACK	112 bits	112 bits	112 bits

在 Ns-2 中，RTS、CTS 以及 ACK 的大小分為別 20、14、14 byte (不含 PHYhdr，PHYhdr = PLCP preamble + PLCP header)；DIFS = 2 SLOT time + SIFS。

最後，Table 7 則是有關 Trace 檔的欄位格式說明。

Table 7：Tracing format in NS-2

欄位	說明
1	Type of operation performed on the packet: enqueue (+), dequeue (-), drop (d), receive (r)
2	Time of the operation
3	Source node of the trace
4	Destination node of the trace
5	Packet type
6	IP packet size
7	Flags
8	IP flow identifier
9	Source IP address
10	Destination IP address
11	Sequence number
12	Unique packet identifier

[1] High Performance TCP/IP Networking M. Hassan, R. Jain