電腦網路實驗課程
NS2(Network Simulation, version 2)
2011-03-30 蕭惠陽
Environment set up
  ◦ Virtual Box
  ◦ NS2
TCL
awk
Simulation
Virtual Box
VirtualBox is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop and embedded use.

Presently, VirtualBox runs on Windows, Linux, Macintosh and OpenSolaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7), DOS/Windows 3.x, Linux (2.4 and 2.6), Solaris and OpenSolaris, and OpenBSD.

Reference: http://www.virtualbox.org/
歡迎使用新增虛擬機器精靈！

這個精靈將引導您完成必要的步驟來新建一個 VirtualBox 的虛擬機器。

使用 [下一步] 按鈕前往精靈的下一頁及 [上一步] 按鈕退回上一頁。如果您要取消這個精靈的執行，您也可以按下 [取消]。

VM 名稱和作業系統類型

輸入新虛擬機器的名稱並選取您計劃安裝到虛擬機器的客體作業系統之類型。

虛擬機器的名稱通常指示它的軟體和硬體組態。所有 VirtualBox 元件將使用它來識別您的虛擬機器。

名稱 (A)

作業系統類型 (I)

作業系統 (C): Microsoft Windows
版本 (V): Windows 2012

記憶體

選取要配置到虛擬機器的基本記憶體 (RAM) 總數，以 MB 為單位。

建議的基本記憶體大小為 512 MB。基本記憶體大小 (M):

512 MB

4 MB

取消
虛擬硬碟
選取用來作為虛擬機器開機硬碟的虛擬硬碟。您可以建立新的硬碟或從拉清單選取現有的。選取後按下的相應的按鈕（調用 [開機磁碟] 視窗）。

如果您需要更複雜的硬碟設定，您也可以跳過這個步驟並稍後使用 VM 設定對話方塊附加硬碟。

開機硬碟的建議大小為 8.00 GB。

硬碟存放類型
選取您想要建立的虛擬硬碟類型。

動態延伸存放：最初在您的實體硬碟佔有非常小的空間量。它將會動態成長（最大到指定的大小）作為客觀作業系統的存放磁碟空間。

固定大小存放：不會成長，它存放在一個與虛擬硬碟大小相同的檔案中。建立一個固定大小存放時可能需要很長的時間，取決於存放大小和您硬碟的寫入效能。

存放類型

動態延伸存放

固定大小存放

虛擬磁碟大小和位置

按 下 [選取] 按鈕以選取存放硬碟資料的檔案位置或在項目欄位輸入檔案名稱。

位置(1)：

fedora14

選取虛擬硬碟的大小，以 MB 為單位。這個大小將在客觀作業系統中作為這個硬碟的最大容量。

大小(3)：

4.00 MB

2.00 TB

摘要

您即將使用以下參數建立新的虛擬硬碟：

類型：動態延伸存放

位置：C:UsersferoxVirtualBoxVMFedora14Fedora14.vdi

大小：10.00 GB (10737418240 B)

如果以上設定正確，請按下 [完成] 按鈕。一旦按下，將會建立一個新的硬碟。
從手邊的清單中選取設定類別並移動滑鼠到設定項目的上方取得更多資訊。
待系統安裝完畢後，要再回來點選按右鍵移除附件
NS2
(Network Simulation, version2)
網路模擬的優點
  ◦ 模擬時間少、成本低
  ◦ 提供網路研究學者驗證自己提出的方法是否有效

NS2
  ◦ 由UC Berkeley所開發
  ◦ 由C++和OTcl為開發語言
Install NS2

- $ tar zxvf ns-allinone-2.34.tar.gz
- $ cd ns-allinone-2.34
- $ ./install
Please put /home/brady/ns2/ns-allinone-2.34/bin:/home/brady/ns2/ns-allinone-2.34/tcl8.4.18/unix:/home/brady/ns2/ns-allinone-2.34/tk8.4.18/unix into your PATH environment, so that you'll be able to run itm/tclsh/wish/xgraph.

IMPORTANT NOTICES:

(1) You MUST put /home/brady/ns2/ns-allinone-2.34/otcl-1.13. /home/brady/ns2/ns-allinone-2.34/lib, into your LD_LIBRARY_PATH environment variable.
   If it complains about X libraries, add path to your X libraries into LD_LIBRARY_PATH.
   If you are using csh, you can set it like:
      setenv LD_LIBRARY_PATH <paths>
   If you are using sh, you can set it like:
      export LD_LIBRARY_PATH=<paths>

(2) You MUST put /home/brady/ns2/ns-allinone-2.34/tcl8.4.18/library into your TCL_LIBRARY environment variable. Otherwise ns/nam will complain during startup.

After these steps, you can now run the ns validation suite with
cd ns-2.34; ./validate

For trouble shooting, please first read ns problems page
http://www.isi.edu/nsnam/ns/ns-problems.html. Also search the ns mailing list archive for related posts.
Set up Path

- `$ vim ~/.bashrc` (增加自己家目錄的export)
- `$ export PATH=`

```
brady@localhost:/home/brady/ns2/ns-allinone-2.34
# .bashrc
#
# Source global definitions
if [ -f /etc/bashrc ]; then
  . /etc/bashrc
fi
#
# User specific aliases and functions
export PATH=/home/brady/ns2/ns-allinone-2.34/bin:/home/brady/ns2/ns-allinone-2.34/tcl8.4.18/unix:/home/brady/ns2/ns-allinone-2.34/tk8.4.18/unix$PATH
export LD_LIBRARY_PATH=/home/brady/ns2/ns-allinone-2.34/otcl-1.13:/home/brady/ns2/ns-allinone-2.34/libSLD_LIBRARY_PATH
export TCL_LIBRARY=/home/brady/ns2/ns-allinone-2.34/tcl8.4.18/library:$TCL_LIBRARY
```

- `$ source .bashrc`
Environment Testing

$ ns

[brady@localhost ns-allinone-2.34]$ ns
%
TCL
(Tool Command Language)
TCL
(Tool Command Language)

- 在NS2扮演的角色為描述腳本
- 描述網路環境和參數設定
Example

```tcl
set month 3
set day 30
set date "$month - $day"
pos $date
```

```bash
# ns date.tcl
# 3 - 30
```
expr

TCL包含很多種表示式，如數學表示式、關係表示式…等，統常會用”expr”去判斷真假或求值

Example

```tcl
set value [expr 2>=1]
puts $value

# ns expr1.tcl
# 1
```
set value [expr 5+2]
puts $value

# ns expr2.tcl
# 7
把執行的結果取代tcl script

Example

puts " I am \[expr 5+2\] years old, and my I.Q. is \[expr 55+20\]. "

# ns expr3.tcl
# I am 7 years old, and my I.Q. is 75.

set my_height 6.0
puts " If I was 2 inches taller, I would be \[expr $my_height+(2.0/12.0)\] feet tall. "

# ns expr4.tcl
# If I was 2 inches taller, I would be 6.166666666666667 feet tall.
流程控制
Control Flow

TCL也有提供控制程式运作流程的方法，包含if–else, switch, while, for, foreach…等

Example

```tcl
set pass 69
if { $pass >= 60 }
    puts “ Congratulations pass. ”
} else {
    puts “ Fail. ”
}
```

```
# ns control1.tcl
# Congratulations pass.
```
set num_class 80
switch $num_class {
    100 {puts “Perfect.”}
    80 {puts “Good.”}
    60 {puts “Bad.”}
}

# ns  control2.tcl
# Good

for {set i 0} {$i<10} {incr i 2} {
    puts “ For loop number $i.”
}

# ns  control3.tcl
# For loop number 0
# For loop number 2
# For loop number 4
# For loop number 6
# For loop number 8
set i 1
while {$i < 10} {
    puts "While loop number $i"
    incr i 2
}

# ns  control4.tcl
# While loop number 1
# While loop number 3
# While loop number 5
# While loop number 7
# While loop number 9
讓使用者自定程序

proc *name* *params* *body*

- name 為程序的名稱
- params 為參數列表
- body 為程序主體

```
proc  *name*  {  *params*  }  {
   *body*...
}
```
Example

```tcl
proc sum_proc { a b }
    { return [expr $a + $b] };

set value1 12
set value 8
Set sum [sum_proc $value1 $value2]
puts "sum_proc value = $sum"

# ns proc1.tcl
# sum_proc value = 20
```
Example

```tcl
for { set i 0} { $i < 0 } {incr i 1} {
    set myarray($i) $i
    puts “ myarray $i value = $myarray($i) ”
}
```

```tcl
# ns arrays1.tcl
# myarray 0 value = 0
# myarray 1 value = 1
# myarray 2 value = 2
# myarray 3 value = 3
    .     .
    .     .
# myarray 9 value = 9
```
輸出
(Output)

- `puts`未指定輸出裝置，內定輸出裝置是螢幕；若有指定輸出裝置為檔案，會把秀出來的內容寫入檔案

- **Example**

```tcl
set f [open "/tmp/myfile" "w"]
for { set i 0} { $i < 0 } {incr i 1} {
    set myarray($i) $i
    puts $f " myarray $i value = $myarray($i) "
}
close $f
```

# ns output1.tcl
awk
awk

- 使用直譯器不須先行編譯
- **awk**擅長處理如
  - 資料列
    ```
    +  0.1  1  2  cbr  1000  -----  2  1.0  3.1  0  0
    ```
  - 欄位
    ```
    | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 十一 | 十二 |
    |----|----|----|----|----|----|----|----|----|----|------|------|
    | +  | 0.1| 1  | 2  | cbr| 1000| -----| 2  | 1.0| 3.1| 0    | 0     |
    ```
- **awk**分析

<table>
<thead>
<tr>
<th>欄位變數</th>
<th>欄位資料內容</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0</td>
<td>為一字串，目前<strong>awk</strong>所讀入的資料列</td>
</tr>
<tr>
<td>$1</td>
<td>代表$0資料列的第一個欄位資料</td>
</tr>
<tr>
<td>$2</td>
<td>代表$0資料列的第二個欄位資料</td>
</tr>
</tbody>
</table>
程式結構

Pattern 1  { Actions 1 }
Pattern 2  { Actions 2 }

Pattern n  { Actions n }

- 一般常用關係判斷式來當成Pattern
  - >, <, >=, <=, ==, &&, != …等

- Actions 為awk由語言所構成，awk語言與c語言很類似
  - printf、if (...) {...} else {...}、while (...) {...}
工作流程

① 從指定的資料檔中讀取一筆資料列
② 更新相關變數之值
③ 逐次執行程式中所有 `pattern { actions }`
④ 反覆執行 ① – ③，一直到資料讀取完畢，`awk` 會自動執行上述步驟，所以使用者不用在程式中寫這個迴圈
此模擬為TCP和UDP傳送封包的過程，並設有DropTail佇列；先使用CBR封包傳送一段時間後再開始送FTP封包，因FTP與CBR互搶頻寬，所以就會發生封包停在佇列的情況，等有限的佇列滿的時候就會看到開始丟棄封包
傳輸節點 S1、S2、路由器 r、資料接收端 d

s1→r
  ◦ TCP
  ◦ 2Mbps 頻寬，10ms 延遲時間
  ◦ Start 1.0s，End 4.0s

s2→r
  ◦ UDP
  ◦ 2Mbps 頻寬，10ms 延遲時間
  ◦ Start 0.1s，End 4.5s

r→d
  ◦ 1.7mbps 頻寬，20ms 延遲時間
  ◦ Droptail 佇列機制，最大佇列 10 個封包長度
TCP Simulation

FTP

TCP

s1,ID=0

2Mbps,10ms

r, ID=1

1.7Mbps,20ms

d, ID=2

sink
# “#”之後為註解，不會被NS2所執行
# 產生一個模擬物件
set ns [new Simulator]
# 定義一個結束的程序
proc finish {} {
    exit 0
}
# 開始新增一些網路架構的程式碼
# 結束模擬的時間(如5秒)
$ns at 5.0 " finish "
# 開始執行模擬
$ns run
產生一個新的模擬物件
set ns [new Simulator]

設定封包顏色給NAM用
$ns color 1 Blue

開一個NAM紀錄檔
set nf [open out.nam w]
$ns namtrace-all $nf

開一個模擬過程的紀錄檔, 紀錄封包傳送過程
set nd [open out.tr w]
$ns trace-all $nd

定義一個結束的程序
proc finish {} {
    global ns nf nd
    $ns flush-trace
    close $nf
    close $nd

    exec nam out.nam &
    exit 0
}
設定傳輸節點,s1的id是0,r的id是1,d的id是2(id會按照所建順序依序產生)
set s1 [$ns node]
set r [$ns node]
set d [$ns node]

設定各連線的傳輸頻寬、延遲時間、佇列機制
$ns duplex-link $s1 $r 2Mb 10ms DropTail
$ns duplex-link $r $d 1.7Mb 20ms DropTail

設定佇列大小
$ns queue-limit $r $d 10

設定模擬畫面節點位置
$ns duplex-link-op $s1 $r orient right-down
$ns duplex-link-op $r $d orient right

設定來觀察r-d佇列的變化(NAM)
$ns duplex-link-op $r $d queuePos 0.5
# 開始建立一個TCP的連線
set tcp [new Agent/TCP]

# 設定節點的通訊協定
$ns attach-agent $s1 $tcp

# 設定TCP的接收端，此TCPSink agent收到tcp封包會回ack
set sink [new Agent/TCPSink]
$ns attach-agent $d $sink
$ns connect $tcp $sink

# 設定此TCP連線為藍色(NAM)
$tcp set fid_1

# 建立一個FTP應用程式在TCP連線上
set ftp [new Application/FTP]

# 設定ftp為tcp協定
$ftp attach-agent $tcp
$ftp set type_ FTP
# 開始建立一個UDP的連線
# 設定此UDP連線為NAM
# 建立一個cbr應用程式在UDP連線上

# 設定FTP和cbr傳送的起始時間和結束時間
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"

# 結束TCP的連線
$ns at 4.5 "$ns detach-agent $s1 $tcp"
$ns at 4.5 "$ns detach-agent $d $sink"

# 5秒後結束此模擬
$ns at 5.0 "finish"

# 開始執行模擬
$ns run
TCP Simulation

TCP & UDP Simulation
Out.tr Analysis

<table>
<thead>
<tr>
<th>欄位一</th>
<th>欄位二</th>
<th>欄位三</th>
<th>欄位四</th>
<th>欄位五</th>
<th>欄位六</th>
<th>欄位七</th>
<th>欄位八</th>
<th>欄位九</th>
<th>欄位十</th>
<th>欄位十一</th>
<th>欄位十二</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>time</td>
<td>From node</td>
<td>To node</td>
<td>Packet type</td>
<td>Packet size</td>
<td>flags</td>
<td>fid</td>
<td>Src addr</td>
<td>Dst addr</td>
<td>Seq num</td>
<td>Packet Id</td>
</tr>
</tbody>
</table>

- 欄位一：封包事件
- 欄位二：事件發生時間
- 欄位三、欄位四：from node 到 to node
- 欄位五：封包型態
- 欄位六：封包大小
- 欄位七：旗標
- 欄位八：表示封包屬於的資料流
- 欄位九、欄位十：封包來源端和目的端
  - x.y -> x:節點編號 , y:port number
- 欄位十一：封包序號
- 欄位十二：封包 id
Event -> r: receive, +: enqueue, -: dequeue, d: drop
BEGIN {

#變數設定，drops (遺失封包數)，packets (傳送總封包數)
drops = 0;
packets = 0;

}

{

action = $1;
time = $2;
from = $3;
to = $4;
type = $5;
pktsize = $6;
flow_id = $8;
src = $9;
dst = $10;
seq_no = $11;
packet_id = $12;
}
統計從n1送出多少packets

判斷共送出多少封包...

統計flow_id為2，且被drop的封包

統計遺失封包個數...

END {

printf("number of packets sent:%d lost:%d\n", packets, drops);

}
1. 安裝NS2，並且可執行任一範例
   ◦ 安裝及模擬的操作過程請以圖片及文字說明
2. 實做TCP與UDP模擬實驗
   ◦ 將實做部分程式碼貼上即可，並繳交完整程式碼及記錄檔
3. 使用awk語言分析記錄檔
   ◦ 將實做部分程式碼貼上即可，並繳交完整程式碼
Download NS2 all-in-one
  ◦ http://sourceforge.net/projects/nsnam/files/allinone/

Download Virtualbox
  ◦ http://www.virtualbox.org/wiki/Downloads

柯志亨, 計算機網路實驗以NS2模擬工具實做, 學貫行銷, 2005.