NETWORK SIMULATIONS

จุดประสงค์

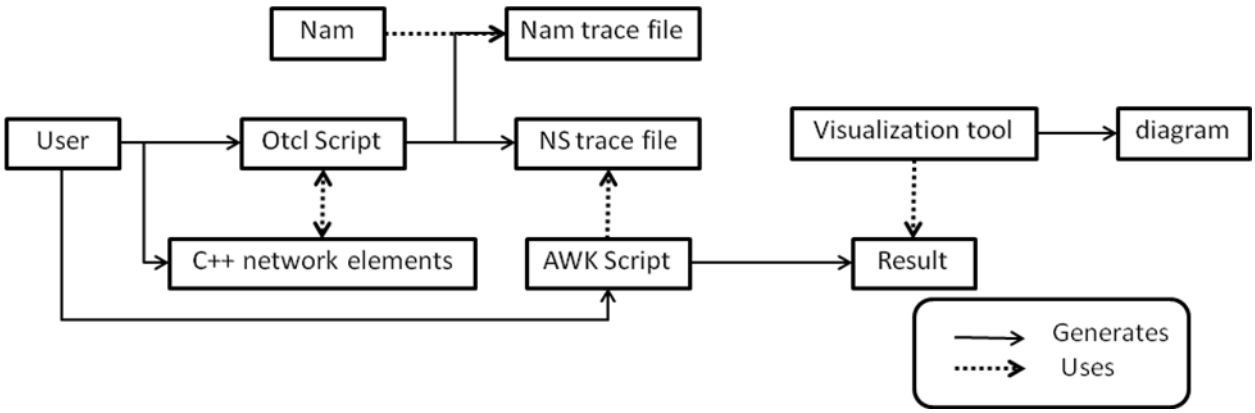
1. สามารถใช้โปรแกรม Network Simulator (NS2)
2. สามารถจำลองการทำงานในเครือข่ายแบบไร้สาย รวมถึงการวิเคราะห์ผลการทดสอบ
3. นำสิ่งที่เรียนไปประยุกต์ใช้ในการทำงานวิจัย หรือเพื่อการเรียนในระดับปริญญาโท-เอกต่อไป

บทนำ

NS2 เป็น open-source ที่สามารถทำงานได้ทั้งบน Linux FreeBSD SunOS และ Window ซึ่งถูกพัฒนาขึ้นโดย Information Sciences Institute (ISI) สำหรับการจำลองการทำงานสามารถวางระบบเครือข่ายได้ทั้งแบบมีสาย และแบบไร้สายได้ ดังนั้นโปรแกรมตัวนี้จึงเป็นที่นิยมในกลุ่มนักวิจัย เพื่อเข้ามาใช้ทดสอบหาทฤษฎีใหม่ๆ หรือแก้ปัญหาระบบเครือข่าย รวมถึงจำลองระบบเครือข่ายก่อนที่จะไปสร้างระบบเครือข่ายจริง

สำหรับเนื้อหาของ Lab จะเป็นการเรียนรู้การใช้โปรแกรมแบบง่าย และสามารถจำลองการทำงานในเครือข่ายแบบไร้สาย โดยใช้มาตรฐานของ IEEE 802.11

หลักการทำงานของ NS2



เริ่มต้นจากผู้ใช้สร้าง Otcl Script แล้วทำการจำลองการทำงาน ผลที่ได้จะเป็น NS trace file และ Nam trace file สำหรับผลที่ได้นี้จะใช้ภาษาที่เรียกว่า AWK Script หรือ Perl Script ช่วยในการวิเคราะห์การทำสอบ และจะใช้ Xgraph หรือ Excel แสดงออกมาเป็นรูปกราฟฟิก

C++ Network elements จะถูกใช้เมื่อต้องการแก้ไขอัลกอริทึม หรือวิเคราะห์ทฤษฎีใหม่ๆ ส่วน Nam trace file จะนำไปประมวลผลแล้วแสดงผลออกมาในรูปของกราฟฟิก

หลักการเขียน Otcl Script

1. Create Simulator Object

set ns [new Simulator] # เป็นการสร้าง Object ของ Simulator

2. Tracing

การเปิด **Nam trace file**

set <variable name> [open <ชื่อ file.nam> w]

$ns namtrace-all file-descriptor: เป็นการบอกให้ simulator บันทึก simulation trace ลงใน file-descriptor ตามรูปแบบของ NAM

w คือ การบอกว่าเป็นการเขียนข้อมูลลงใน file

set nf [open out.nam w]

$ns namtrace-all $nf

การเปิด **trace file**

set <variable name>[open <ชื่อ file.tr> w]

$ns trace-all $<variable name>

w คือ การบอกว่าเป็นการเขียนข้อมูลลงใน file

set tf [open out.tr w]

$ns trace-all $tf

$ns flush-trace: เป็น member function ของ trace-all ซึ่งเป็นคำสั่งที่ใช้ในการบันทึก simulation

3. Create Nodes (Physical layer)

set <variable name> [$ns node]

set n0 [$ns node]

set n1 [$ns node]

4. Creating TCP connection (Transport layer)

set tcp [new Agent/TCP]

set tcpsink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp

$ns attach-agent $n1 $tcpsink

$ns connect $tcp $tcpsink

5. Creating Traffic (application layer)

FTP

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

Telnet

```
set telnet [new Application/Telnet]
$telnet attach-agent $tcp
```

6. การเขียน Procedure

proc ชื่อprocedure {}

{ body }

```
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
       }
```

7. Call Procedure

$ns at 5.0 "finish"

8. Schedule event

$ns at <time> <event>: เป็นคำสั่งที่จะทำการประมวลผล event ตามเวลาที่กำหนด

<event>: คำสั่งที่ได้กำหนดไว้ใน ns/tcl

$ns at 1.0 “Start”

9. Start NS (เริ่มการทำงานของ Script)

$ns run

10. Stop NS

Exit 0

11. การ Run โปรแกรม NS

การ Run แบบ interactive mode โดยการพิมพ์ ns แล้วจึงพิมพ์ command ต่าง ๆ ทีละบรรทัด

```
bash-shell$ ns
% set ns [new Simulator]
% $ns at 1 “puts \“Hello World!\””
% $ns at 1.5 “exit”
% $ns run
Output :
        Hello World!
```

การ Run แบบ Batch mode เป็นการที่ run จาก file ที่มีการกำหนด Script ไว้เรียบร้อยแล้ว

- ใช้คำสั่ง ns <Otcl script file>

12. กระบวนการของการประมวลผลของ NS

- อ่าน Otcl file
- Run simulation program
- สร้าง trace file
- แสดงผลลัพธ์โดยใช้ NAM (network animator)
- ผลทางสถิติ

นักศึกษาสามารถเรียนรู้การทำงานของโปรแกรมดังนี้

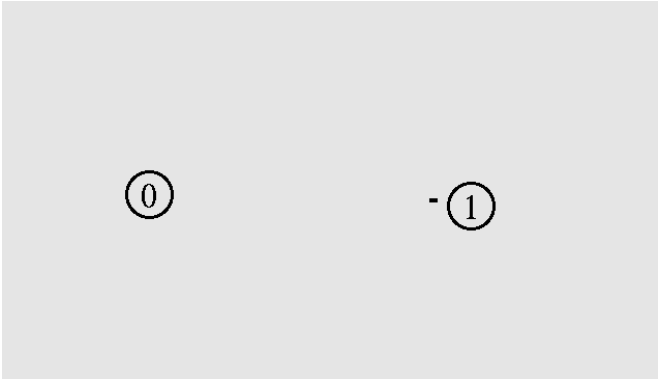
ตอนที่ 1 แนะนำการเขียนไฟล์ Otcl และดูผลลัพธ์ของการจำลองเครือข่ายแบบไร้สาย
ตอนที่ 2 สามารถเข้าใจไฟล์ NS trace เพื่อนำไปใช้ในการวิเคราะห์สมรรถนะในการทำงานของเครือข่าย
ตอนที่ 3 แก้ไขไฟล์ Otcl เพื่อสามารถออกแบบการจำลองด้วยตัวเองได้

ตอนที่ 1 แนะนำการเขียนไฟล์ Otcl

ในการกำหนดรูปแบบการทำงานของระบบเครือข่ายจะต้องทำการสร้างไฟล์ Otcl เพื่อทำการเลือกโพรโทคอล และรูปแบบของระบบเครือข่ายที่ต้องการจำลองการทำงาน และกำหนดค่าพารามิเตอร์ที่ใช้ใน

การจำลอง เพื่อให้การทำงาน ตรงกับรูปแบบของระบบเครือข่ายที่ได้ทำการออกแบบไว้ โดยมีรายละเอียดการออกแบบดังนี้

ออกแบบการส่งข้อมูลในเครือข่ายไร้สาย โดยใช้มาตรฐาน IEEE 802.11 โดยการสื่อสารระหว่างอุปกรณ์เป็นแบบ Ad-hoc ซึ่งลักษณะการทำงานแบบนี้เป็นรูปแบบหนึ่งของเครือข่ายไร้สาย ที่ไม่ต้องการสถานีฐาน โดยที่โหนดแต่ละตัวสามารถเคลื่อนที่แบบอิสระดังแสดงในรูปที่ 1



รูปที่ 1 การสื่อสารแบบไร้สาย

จากรูปที่ 1 สามารถเขียนไฟล์ Otcl ดังนี้

```
# A simple example for wireless simulation or  simple.tr

# ===================================================================

# Define options

# ===================================================================

setval(chan)        Channel/WirelessChannel     ;# ชนิดของช่องสัญญาณ
setval(prop)        Propagation/TwoRayGround    ;# รูปแบบของคลื่นวิทยุ
setval(netif)       Phy/WirelessPhy             ;# ชนิดของ network interface
setval(mac)         Mac/802_11                  ;#โพรโตคอลในระดับชั้น MAC
setval(ifq)         Queue/DropTail/PriQueue     ;# รูปแบบของคิว
setval(ll)          LL                          ;# link layer
setval(ant)         Antenna/OmniAntenna         ;# antenna model
setval(ifqlen)      50                          ;# ขนาดของบัฟเฟอร์(จำนวน Packet)
```

```
setval(nn)          2                    ;# จำนวนของโหนด
setval(rp)          AODV                 ;# โพรโตคอลในระดับชั้น Routing
setval(x)           2000                 ;# ความยาวของแกน x
setval(y)           2000                 ;# ความยาวของแกน y
setval(time)        500.0                ;# ระยะเวลาในการจำลอง
setstart_time       60.0                 ; # เวลาที่เริ่มจำลอง

proc finish {} {
global ns_ tracefilenamfile
$ns_ flush-trace
close $tracefile
close $namfile
exit 0
}
# Main Program
# Initialize Global Variables
set ns_ [new Simulator]

# Tracde Filesกำหนดรูปแบบของไฟล์ที่บันทึกเหตุการณ์ที่เกิดขึ้นในการจำลอง
settracefile [open simple.tr w]
$ns_ use-newtrace
$ns_ trace-all $tracefile

# Nam Files กำหนดฟลลัพธ์ของการจำลองในรูปแบบของกราฟฟิก
setnamfile [open simple.nam w]
$ns_ namtrace-all-wireless $namfile $val(x) $val(y)
# set up topography object
settopo       [new Topography]
$topoload_flatgrid $val(x) $val(y)

# Create God
create-god $val(nn)
setchan [new $val(chan)]
```

```
# configure node กำหนดพารามิเตอร์ให้กับโหนดที่ใช้ในการจำลอง
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
                 -channel $chan
        for {set i 0} {$i< $val(nn) } {incri} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0 ;# disable random motion
        $ns_ initial_node_pos $node_($i) 30
        }

#กำหนดค่าพิกัดของโหนดเริ่มต้น
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
# and produce some simple node movements

        $node_(0) set X_ 0.0
        $node_(0) set Y_ 200.0
        $node_(0) set Z_ 100.0
        $node_(1) set X_ 350.0
        $node_(1) set Y_ 200.0
        $node_(1) set Z_ 100.0
        $ns_ at 180.0 "$node_(1) setdest 350.0 400.0 100.0"
```

#ในวินาทีที่ 180 ของการจำลอง กำหนดให้โหนดหมายเลข 1 เคลื่อนที่ไปที่พิกัด 350 200

Initial Movementกำหนดพิกัดตั้งต้น

```
for {set i 0} {$i< $val(nn) } {incri} {
$ns_ at 0.0 "$node_($i) setdest 10.0 10.0 0.0"
}
```

#กำหนดการูปแบบของการส่งข้อมูลในระบบเครือข่าย

#การส่งข้อมูลแบบ Constant Bit Rate มีโหนดต้นทาง คือ โหนด 0 และโหนดปลายทางคือ โหนด 5

Data Source , Nodesโหนดต้นทาง คือ โหนด 0

```
setudp_(0) [new Agent/UDP]
$udp_(0) set fid_ 0
$ns_ attach-agent $node_(0) $udp_(0)
```

Null Agent to receive Packets for Node 0

กำหนดโหนดปลายทาง คือ โหนด 5

```
set null_(0) [new Agent/Null]
$null_(0) set fid_ 1
$ns_ attach-agent $node_(1) $null_(0)
```

Constant Bit rate traffic generator

กำหนดขนาด และ อัตราในการส่งข้อมูล

```
setcbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.1
$cbr_(0) set random_ 0
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
```

```
$ns_ connect $udp_(0) $null_(0)
$ns_ at $start_time "$cbr_(0) start"

# Tell nodes when the simulation ends
# สิ้นสุดการจำลอง

for {set i 0} {$i< $val(nn) } {incri} {
    $ns_ at $val(time) "$node_($i) reset";
}

$ns_ at $val(time) "finish"
$ns_ at [expr $val(time) + 0.01] "puts \"NS EXITING...\"; $ns_ halt"
puts "Starting Simulation..."
$ns_ run
```

การ run โปรแกรม NS

การทดสอบการทำงานของ file (.tcl)

ผู้ใช้จะต้อง cd เข้าไปยัง Directory ที่เก็บ file(.tcl) แล้วพิมพ์คำสั่ง ns ตามด้วยชื่อfile(.tcl) ที่ต้องการจะประมวลผล

ns simple.tcl

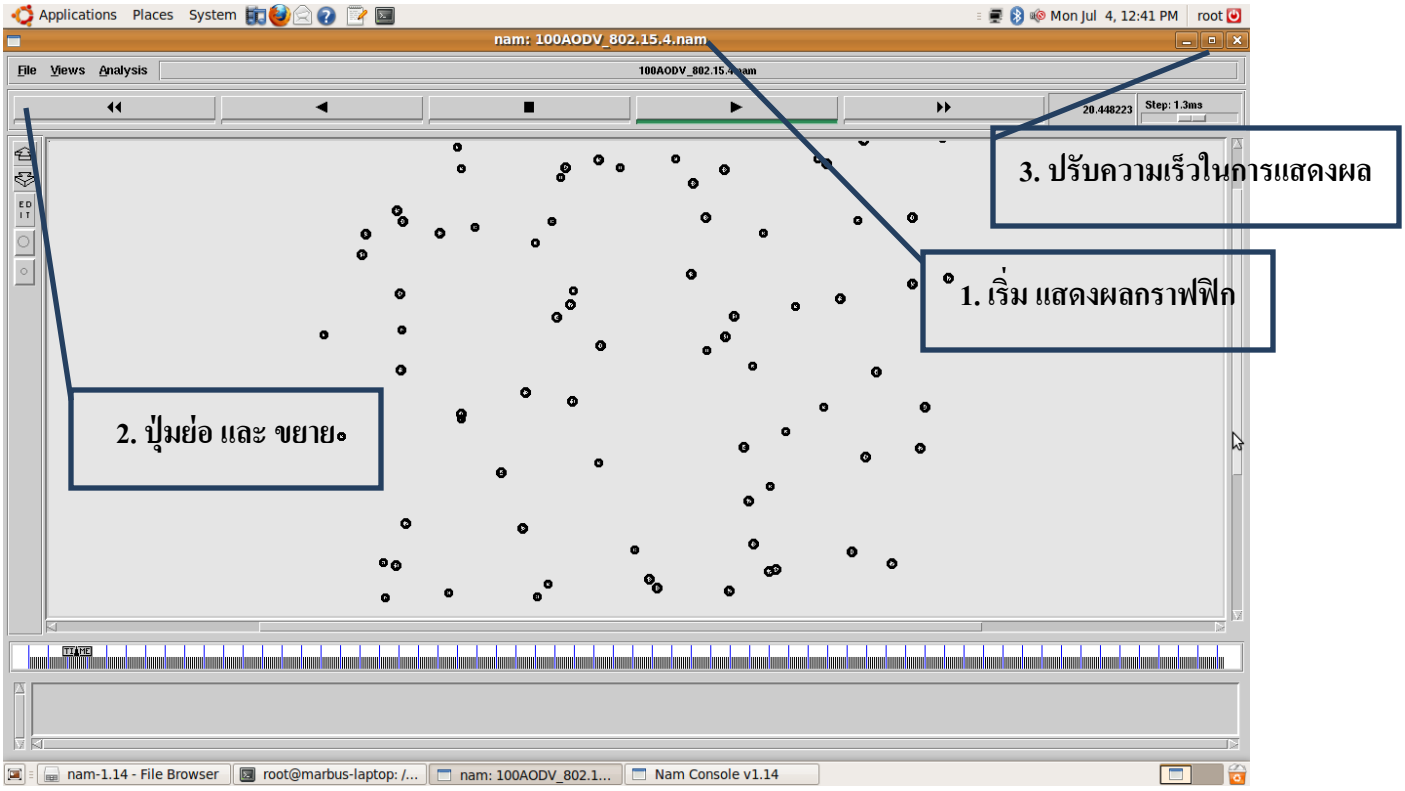
หลังจากพิมพ์คำสั่ง NS-2 จะทำการประมวลผลการจำลองการทำงานและสร้างผลลัพธ์ของการจำลองเก็บไว้ใน Directory เดียวกับที่เก็บ file(.tcl) โดยมีผลลัพธ์ที่ได้จากการจำลองสองไฟล์ คือ Trace file(.tr) และ Network AniMator file (.nam)

ในส่วนของ nam เป็นส่วนของการแสดงผลออกมาในรูปกราฟฟิก โดยต้องเข้าไปใน Directory ที่เก็บ file (.nam) และพิมพ์คำสั่ง

nam simple.nam

โปรแกรม NAM สามารถแสดงผลกราฟฟิก

ดังนี้



#checkpoint 1

ผลลัพธ์ที่ได้เมื่อพิมพ์คำสั่ง ns simple.tcl

.....

.....

.....

ผลลัพธ์ที่ได้เมื่อพิมพ์คำสั่ง nam simple.nam

.....

.....

.....

โหนดใดบ้างมีการเคลื่อนที่ และเคลื่อนที่ ที่วินาทีที่เท่าไหร่ของการจำลอง

.....

.....

.....

<u>ตอนที่ 2</u> การวิเคราะห์ผลการจำลอง

ผลลัพธ์ที่ได้จากการจำลอง file (.tr) เป็น log file ที่เก็บค่ารายละเอียดการจำลองการทำงานเป็นข้อมูลดิบที่ต้องผ่านกระบวนการวิเคราะห์ผลจึงสามารดูผลลัพธ์ของการจำลองได้ โดยสามารถอ่านรายละเอียดของเหตุการณ์ที่เกิดขึ้นในช่วงเวลาต่างๆได้ สามารถดูโดยใช้คำสั่ง

gedit simple.tr

ความหมายของเหตุการณ์ที่ถูกบันทึกใน trace file
ข้อมูล field แรก อธิบายเกี่ยวกับชนิดของเหตุการณ์ มี 4 ชนิด เหตุการณ์ คือ

s send
r receive
d drop
f forward

ข้อมูล field ถัดไป
-t time

Node property
-Ni: node id
-Nx: node's x-coordinate
-Ny: node's y-coordinate
-Nz: node's z-coordinate
-Ne: node energy level
-Nl: trace level, such as AGT, RTR, MAC
-Nw: reason for the event. The different reasons for dropping a packet are given below:
Packet information at IP level
-Is: source address.source port number

-Id: destaddress.dest port number

-It: packet type

-Il: packet size

-If: flow id

-Ii: unique id

-Iv: ttl value

Next hop info

This field provides next hop info and the tag starts with a leading “-H”.

-Hs: id for this node

-Hd: id for next hop towards the destination.

#checkpoint 2

จงบอกความหมายของเหตุการณ์ใน .tr ที่ยกมาดังนี้

s -t 60.002992333 -Hs0 -Hd -2 -Ni 0 -Nx0.00 -Ny200.00 -Nz100.00 -Ne -1.000000 -Nl MAC -Nw --- -Ma 4fe -Md1 -Ms0 -Mt 0

……

……

……

……

……

r -t 60.003345500 -Hs 1 -Hd -2 -Ni 1 -Nx 350.00 -Ny 200.00 -Nz 100.00 -Ne -1.000000 -Nl MAC -Nw --- -Ma 4fe -Md 1 -Ms 0 -Mt 0

……

……

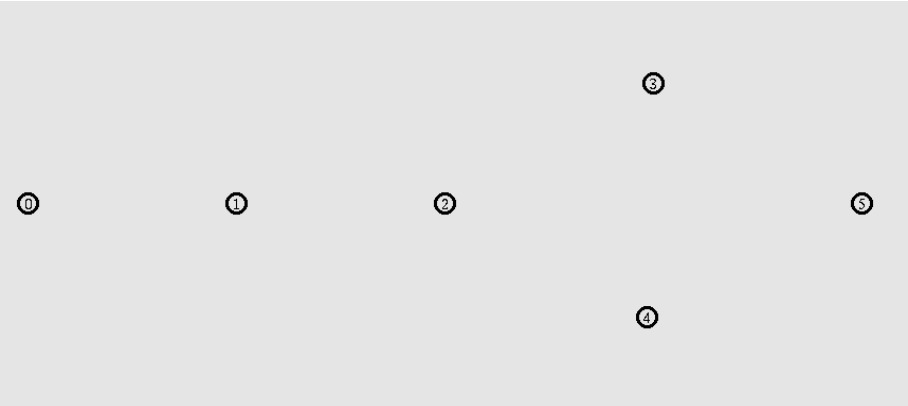
……

……

……

ตอนที่ 3

ให้นักศึกษาทำการออกแบบการจำลองการทำงานของเครือข่ายไร้สายดังแสดงในรูปที่ 2



รูปที่ 2 รูปแบบของการจำลอง

โดยกำหนดให้ทำการแก้ไขไฟล์ simple.tcl กำหนดพิกัดเริ่มต้นของแต่ละโหนดดังนี้

$node_(0) set X_ 0.0

$node_(0) set Y_ 200.0

$node_(0) set Z_ 0.0

$node_(1) set X_ 350.0

$node_(1) set Y_ 200.0

$node_(1) set Z_ 0.0

$node_(2) set X_ 700.0

$node_(2) set Y_ 200.0

$node_(2) set Z_ 0.0

$node_(3) set X_ 1040.0

$node_(3) set Y_ 390.0

$node_(3) set Z_ 0.0

$node_(4) set X_ 1040.0

$node_(4) set Y_ 0.0

$node_(4) set Z_ 0.0

$node_(5) set X_ 1400.0

$node_(5) set Y_ 200.0

$node_(5) set Z_ 0.0

และเมื่อถึงวินาทีที่ 180 ของการจำลอง ให้โหนดที่ 3 เคลื่อนที่ไปพิกัด 1050.0 900.0 0.0
การส่งข้อมูลเป็นแบบ Constant Bit rate (CBR) โดยมีโหนด 0 เป็น โหนดต้นทาง และโหนด 5 เป็นโหนดปลายทาง
เมื่อทำการแก้ไขไฟล์ simple.tcl แล้ว ทำการบันทึกและจำลองโดยการพิมพ์คำสั่งใน terminal

ns simple.tcl

เมื่อได้ผลลัพธ์ของการจำลองสามารถตรวจสอบผลลัพธ์ของการจำลองโดยพิมพ์คำสั่ง

nam simple.nam

เพื่อดูผลลัพธ์ของการจำลองว่าถูกต้องตามคำสั่งหรือไม่

#checkpoint 3 สามารถแก้ไขไฟล์ simple.tcl ให้จำลองการทำงานได้อย่างถูกต้องตามคำสั่ง

เมื่อดูจากกราฟฟิกของผลลัพธ์เหตุใดจึงมีการเปลี่ยนเส้นทางในการส่งข้อมูล

.....

.....

.....

.....

.....

มีการดรอปของข้อมูลเกิดขึ้นครั้งแรกวินาทีที่เท่าไหร่ของการจำลอง

.....

.....

.....

.....

.....

Hint สามารถดูได้จาก trace file (.tr)