

# ***Sommaire***

- ***Introduction***

La simulation permet de tester à moindre coût les nouveaux protocoles et d'anticiper les problèmes qui pourront se poser dans le futur afin d'implémenter la technologie la mieux adaptée aux besoins. NS [75] est un simulateur à événements discrets disponible gratuitement sur le site <http://www.isi.edu/nsnam/>. Il permet à l'utilisateur de définir un réseau et de simuler des communications entre les nœuds de ce réseau. NS v2 utilise le langage OTCL (Object Tools Command Language), dérivé objet de TCL. À travers ce langage, l'utilisateur décrit les conditions de la simulation : topologie du réseau, caractéristiques des liens physiques, protocoles utilisés, communications... La simulation doit d'abord être saisie sous forme de fichier texte que NS utilise pour produire un fichier trace contenant les résultats. NS est fourni avec différents utilitaires dont des générateurs aléatoires et un programme de visualisation : NAM.

- ***Présentation de NS2***

Le NS2 est un simulateur de réseaux orienté objet, écrit sur la base du langage C++, avec au devant un interpréteur OTcl. La question qui se pose c'est pourquoi deux langages ? Tout simplement par ce que le NS fonctionne suivant deux raisonnements différents. D'une part, la manipulation des bits, des entêtes de paquet, mais aussi de pouvoir implémenter des algorithmes capables de parcourir plusieurs types de données donc la création rapide et efficace des objets et variables manipulés lors de la simulation. Pour cette tâche une rapidité d'exécution est requise, et est importante (*la découverte des erreurs la correction recompilation et en fin réexécution est moins importante*), ceci est offert par le C++. D'autre part la configuration des objets et la gestion des événements, ou autre l'exploration d'un grand nombre de scénarios (*Changement du modèle et réexécution*), donc le temps d'itération est plus important par rapport à la rapidité d'exécution, ceci par contre est offert par l'OTcl, qui le permet d'une manière interactive. Notons aussi que le tclcl permet aux objets et variables d'apparaître et d'être utilisés par les deux langages.

- ***Présentation de Network Animator (NAM)***

Le Nam est un outil d'animation basé sur Tcl/Tk, utilisé dans NS afin de visualiser le tracé de simulation des réseaux, ainsi que les tracés de données. Le modèle théorique du Nam a été non seulement créé pour lire un large ensemble de données d'animation, mais aussi suffisamment extensible pour être utilisé quelque soit le type de réseau simulé (*fixe ou mobile ou mixte*). Ce qui permet de visualiser tout type de situation possible.

- ***Exercice 0***

Dans cet exercice on a installé le simulateur NS, pour ce faire on a suivi les étapes suivantes :

**Etape 1 :**

Télécharger le package « ns-allinone-2.32.tar.gz ».

**Etape 2 :**

Copier le fichier « ns-allinone-2.32.tar.gz » dans le répertoire «/home/» en utilisant la commande « cp ».

**Etape 3 :**

Extraire ce fichier dans « /home/ » :

```
#tar -zxvf ns-allinone-2.32.tar.gz
```

#### Etape 4 :

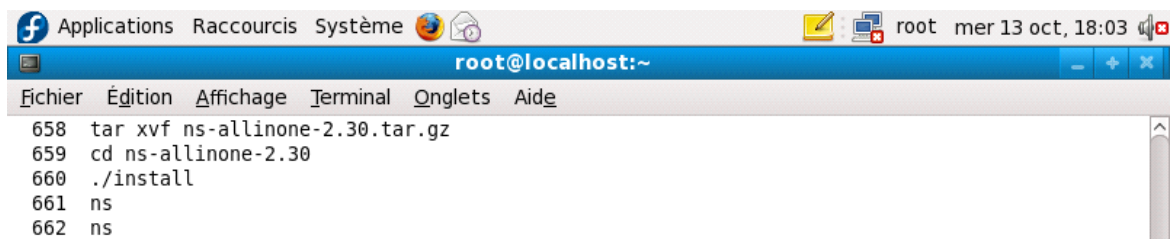
Positionner sur le dossier extrait pour pouvoir installer le package.

```
#cd ns-allinone-2.32
```

#### Etape 5 :

Installer le package et cela par exécuter le script d'installation comme suit :

```
#./install
```



The screenshot shows a terminal window titled 'root@localhost:~' with a menu bar containing 'Fichier', 'Édition', 'Affichage', 'Terminal', 'Onglets', and 'Aide'. The terminal output shows the following commands and their results:

```
658 tar xvf ns-allinone-2.30.tar.gz
659 cd ns-allinone-2.30
660 ./install
661 ns
662 ns
```

#### Etape 6 :

```
775 export PATH=$PATH:/root/Bureau/ns-allinone-2.30/bin:/root/Bureau/ns-allinone-2.30/tcl8.4.13/unix:/root/Bureau/ns-allinone-2.30/tk8.4.13/unix
776 export PATH=$PATH:/root/Bureau/ns-allinone-2.30/otcl-1.12:/root/Bureau/ns-allinone-2.30/lib
```

Paramétrer les variables d'environnement :

- **Exercice I :**

Dans cet exercice, simuler deux nœuds tels que le premier est un nœud stable et l'autre est un nœud qui s'approche et s'éloigne du premier.

L'environnement de simulation :

- Espace de travail : 600m \* 600 m

· Durée de la simulation : 30s

## V.1-Le programme

```
set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
model
set val(netif)          Phy/WirelessPhy           ;# network interface
type
set val(mac)            Mac/802_11                ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)            LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                       ;# max packet in ifq
set val(nn)             1                         ;# number of mobilenodes
set val(rp)            DSDV                       ;# routing protocol
#set val(rp)            DSR                       ;# routing protocol
set val(x)              600
set val(y)              600

# Initialize Global Variables
set ns_                  [new Simulator]
set tracefd              [open /home/user01/Bureau/res.tr w]
$ns_ trace-all $tracefd

set namtrace [open /home/user01/Bureau/res.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo                 [new Topography]

$topo load_flatgrid $val(x) $val(y)
# Create God
create-god $val(nn)
# New API to config node:
# 1. Create channel (or multiple-channels);
# 2. Specify channel in node-config (instead of channelType);
# 3. Create nodes for simulations.

# Create channel #1 and #2
set chan_1_ [new $val(chan)]

# Create node(0) "attached" to channel #1

# configure node, please note the change below.
$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 OFF \
        -channel $chan_1_

set node_(0) [$ns_ node]
set node_(1) [$ns_ node]
$node_(0) random-motion 0
$node_(1) random-motion 0

for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 20
}

#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
#
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

# Now produce some simple node movements
# Node_(1) starts to move towards node_(0)
#
$ns_ at 3.0 "$node_(1) setdest 50.0 40.0 25.0"
$ns_ at 3.0 "$node_(0) setdest 48.0 38.0 5.0"

# Setup traffic flow between nodes
# TCP connections between node_(0) and node_(1)

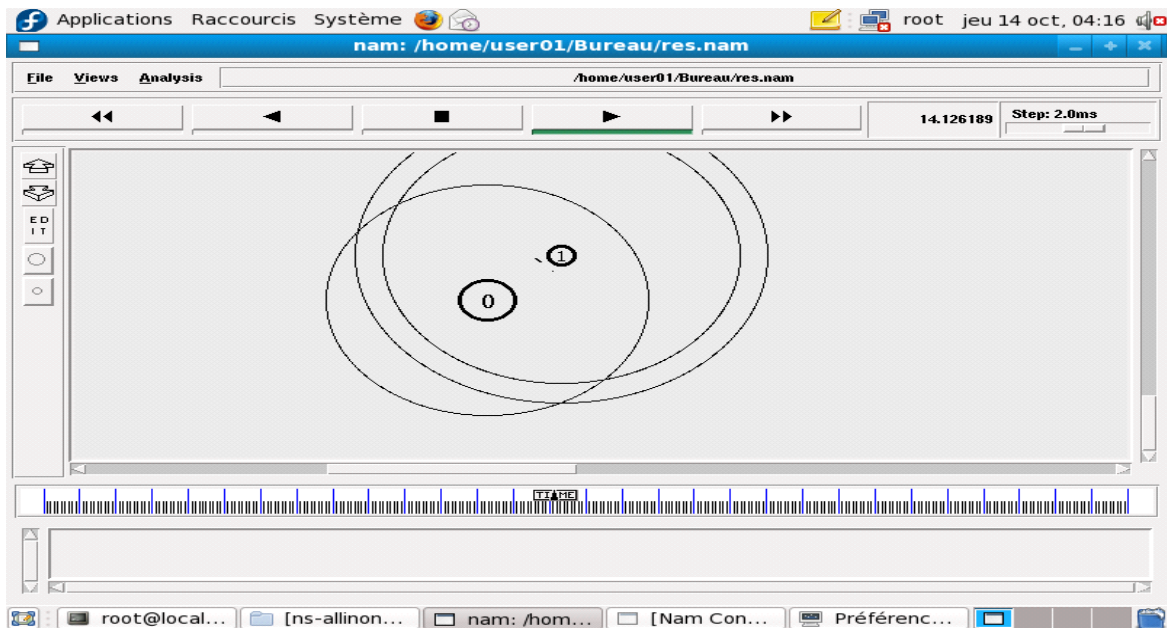
set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 3.0 "$ftp start"

#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 30.0 "$node_($i) reset";
}
$ns_ at 30.0 "stop"
$ns_ at 30.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

puts "Starting Simulation..."
$ns_ run

```

## V.2- L'exécution du programme



- **Exercice II**

On a considéré la topologie représentée sur la Figure 1. Les nœuds n0 et n1 génèrent du trafic vers les nœuds n5 et n6 respectivement. Leurs trafics passent par les nœuds intermédiaires n2, n3, n4. Le protocole utilisé est TCPNewReno, la portée de chaque nœud est de 250 m, Les applications lancées au niveau des deux nœuds sources sont de types ftp. Dans l'expérience à simuler, le nœud n1 émet du trafic entre  $t=2$  et  $t=300s$  tandis que n0 lui est actif pendant  $t=4$  et  $t=400s$  dans un espace de travail de  $1000m \times 1000m$ .

## VI.1- Le programme

```
set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
model
set val(netif)          Phy/WirelessPhy           ;# network interface
type
set val(mac)            Mac/802_11                ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                        ;# link layer type
set val(ant)            Antenna/OmniAntenna       ;# antenna model
set val(ifqlen)         50                        ;# max packet in ifq
set val(nn)             6                        ;# number of mobilenodes
set val(rp)             DSDV                      ;# routing protocol
#set val(rp)            DSR                      ;# routing protocol
set val(x)              1000
set val(y)              1000

# Initialize Global Variables
set ns_                 [new Simulator]

set tracefd             [open /home/user01/Bureau/tp_2.tr w]
$ns_ trace-all $tracefd
set namtrace             [open /home/user01/Bureau/tp_2.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo                 [new Topography]

$topo load_flatgrid $val(x) $val(y)
# Create God
create-god $val(nn)
# Create channel #1 and #2
set chan_1_             [new $val(chan)]
# configure node, please note the change below.
$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 OFF \
        -channel $chan_1_

set node_(0) [$ns_ node]
set node_(1) [$ns_ node]
set node_(2) [$ns_ node]
set node_(3) [$ns_ node]
set node_(4) [$ns_ node]
set node_(5) [$ns_ node]
set node_(6) [$ns_ node]
$node_(0) random-motion 0
$node_(1) random-motion 0
$node_(2) random-motion 0
$node_(3) random-motion 0
$node_(4) random-motion 0
$node_(5) random-motion 0
$node_(6) random-motion 0
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 20
}

#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
#
$node_(0) set X_ 100.0
$node_(0) set Y_ 100.0
$node_(0) set Z_ 0.0
$node_(2) set X_ 200.0
$node_(2) set Y_ 200.0
$node_(2) set Z_ 0.0
$node_(3) set X_ 350.0
$node_(3) set Y_ 200.0
$node_(3) set Z_ 0.0
$node_(4) set X_ 500.0
$node_(4) set Y_ 200.0
$node_(4) set Z_ 0.0
$node_(5) set X_ 600.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0
$node_(1) set X_ 100.0
$node_(1) set Y_ 300.0
$node_(1) set Z_ 0.0
$node_(6) set X_ 600.0
$node_(6) set Y_ 300.0
$node_(6) set Z_ 0.0

$ns_ at 1.0 "$node_(1) setdest 100.0 300.0 0.0"
$ns_ at 0.0 "$node_(0) setdest 100.0 100.0 0.0"
$ns_ at 2.0 "$node_(2) setdest 200.0 200.0 0.0"
$ns_ at 3.0 "$node_(3) setdest 350.0 200.0 0.0"
$ns_ at 4.0 "$node_(4) setdest 500.0 200.0 0.0"
$ns_ at 5.0 "$node_(5) setdest 600.0 100.0 0.0"
$ns_ at 6.0 "$node_(6) setdest 600.0 300.0 0.0"

# Setup traffic flow between nodes
# TCP connections between node_(0) and node_(1)
set tcpl [new Agent/TCP/Vegas]
$tcpl set class_ 2

```



```

$ns_ color 2 Blue
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp1
$ns_ attach-agent $node_(5) $sink1
$ns_ connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at 2.0 "$ftp1 start"

set tcp2 [new Agent/TCP/Vegas]
$tcp2 set class_ 1
$ns_ color 1 Red
set sink2 [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $tcp2
$ns_ attach-agent $node_(6) $sink2
$ns_ connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns_ at 4.0 "$ftp2 start"
# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 30.0 "$node_($i) reset";
}
$ns_ at 300.0 "stop"
$ns_ at 300.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

puts "Starting Simulation

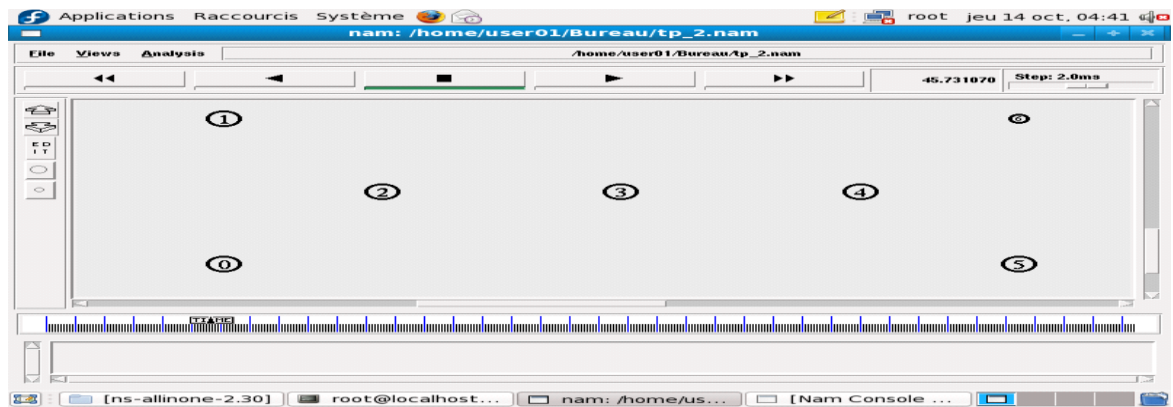
```

## VI.2-L'exécution du programme

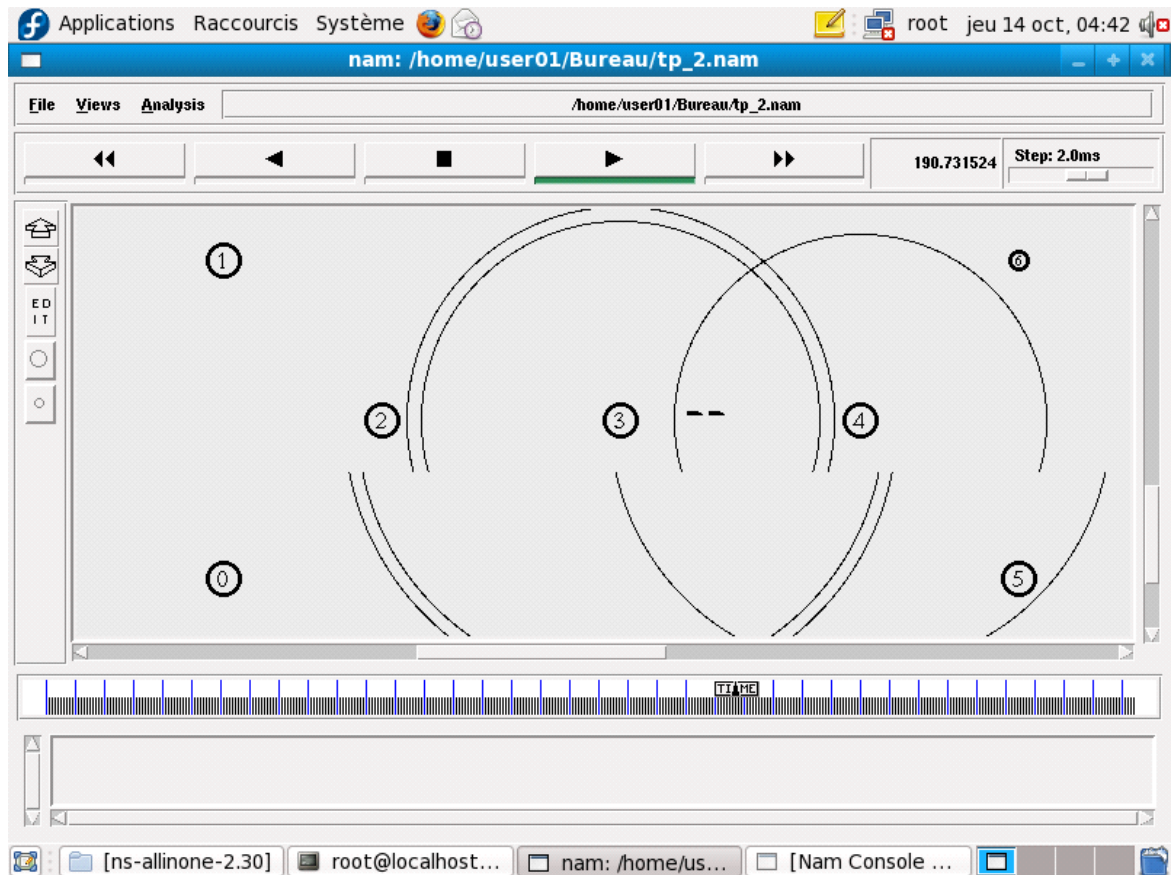
```

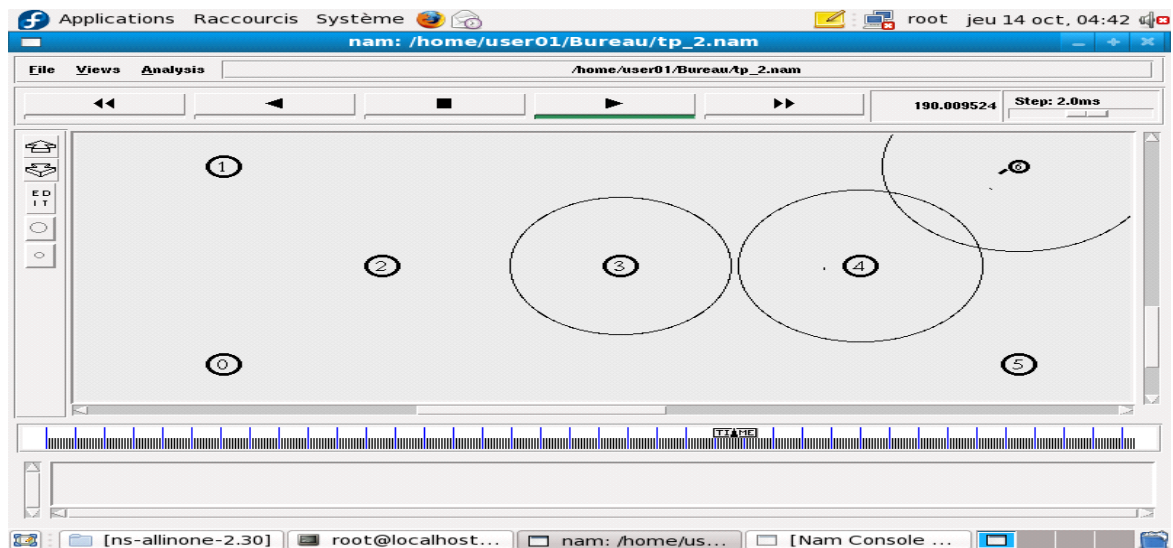
root@localhost:~/Bureau/ns-allinone-2.30
[root@localhost ~]# export PATH=$PATH:/root/Bureau/ns-allinone-2.30/otcl-1.12:/root/Bureau/ns-allinone-2.30/lib
[root@localhost ~]# cd Bureau
[root@localhost Bureau]# cd ns-allinone-2.30
[root@localhost ns-allinone-2.30]# ns tp_2.tcl
num nodes is set 6
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
SORTING LISTS ...DONE!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
MAC 802_11: accessing MAC cache_ array out of range (src 6, dst 4, size 6)!
[suppressing additional MAC cache_ warnings]
NS EXITING...
[root@localhost ns-allinone-2.30]#

```

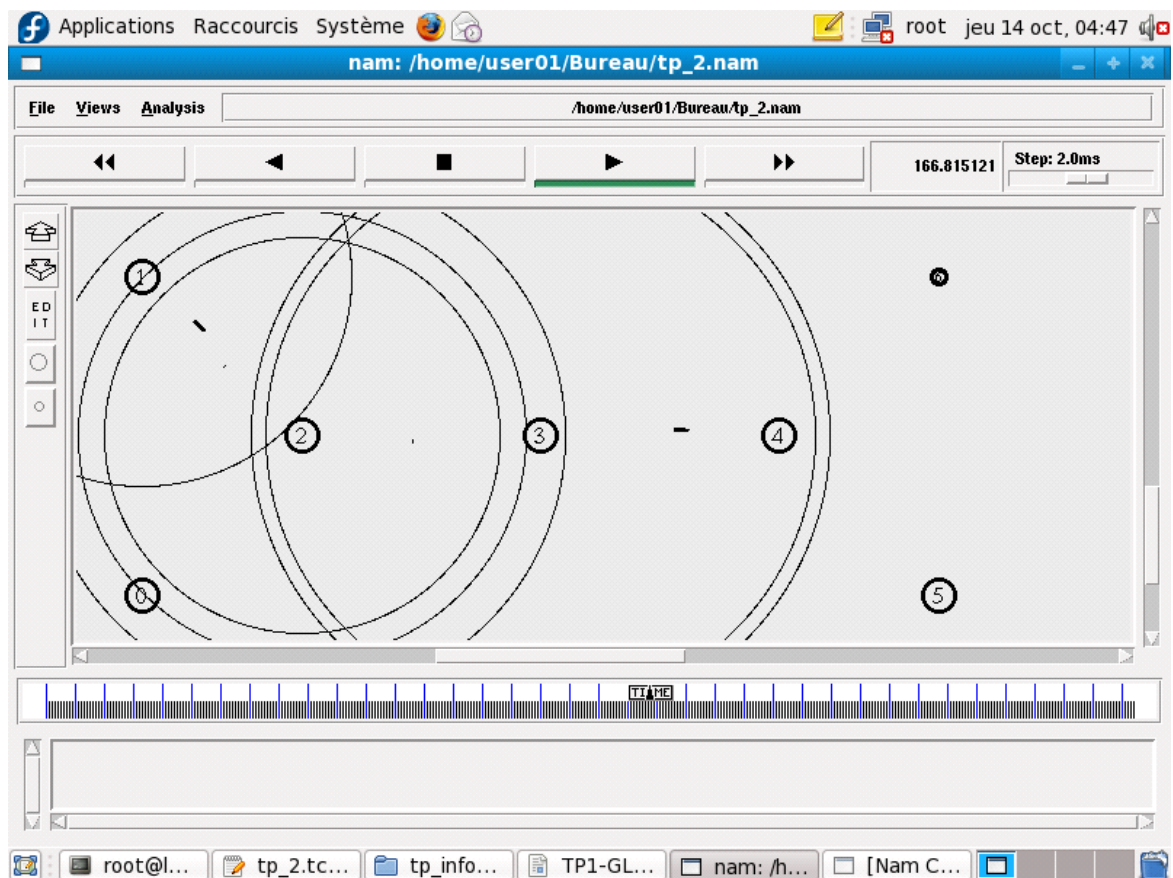


### Le résultat en utilisant le protocole TCP/Vegas





## Le résultat en utilisant le protocole TCP/Reno



### **Comparaison entre TCP/Vegas et TCP/Reno :**

TCP Vegas est une extension de TCP Reno. Dans TCP Vegas, l'émetteur enregistre les temps auxquels un segment est émis ainsi que les temps auxquels on reçoit l'accusé de réception correspondant. Quand un acquittement arrive à l'émetteur, TCP Vegas calcule le RTT en se basant sur l'horloge du système. Il utilise ensuite cette valeur pour décider s'il doit retransmettre le dernier segment émis ou non. Par exemple, lorsque le RTT estimé est plus grand que la valeur du RTO, il retransmet le paquet suivant directement sans avoir à attendre les 3 acquittements dupliqués. Autrement dit, TCP Vegas traite la réception d'un acquittement comme un indice pour voir s' il est nécessaire de déclencher le RTO pour les paquets suivants ou non. Ainsi, TCP Vegas détecte la perte de paquets de manière plus rapide que les autres variantes. L'objectif mis en évidence dans l'algorithme de Vegas, est d'essayer d'obtenir un taux de transmission plus élevé avec moins de retransmissions. Donc le TCP/Vegas est plus performant que TCP/Reno.