Wireless and Mobile Networks: Simulation d'une architecture SANET avec NS2

M'hamed SEFIANE EL OUADGHIRI
2e année Génie Logiciel - ENSIAS
mhamed_sefianeelouadghiri@um5.ac.ma
https://github.com/mhamedouadghiri/wmn-sanet

***Abstract*—Les réseaux ANETs (Ad hoc NETWORKs), consistent en une grande population, relativement dense, d'unités sans fil dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée. Une caractéristique particulière du réseau ad hoc est l'absence de toute installation fixe. Ceci lui permet d'être rapide et facile à déployer. Pour cela, le réseau ad hoc est utilisé pour les applications tactiques comme les opérations de secours, militaires ou d'exploration. Notre but est de comparer les performances des protocoles TCP New Reno et TCP Vegas dans un environnement SANET.**

***Index Terms*—TCP, New Reno, Vegas, SANET, NS2, NAM**

I. Introduction

Nous commençons par présenter la démarche d'installation de l'outil NS2. Nous présentons ensuite l'architecture SANET à déployer et le code TCL réalisé. Finalement nous analysons les résultats obtenus suite l'aboutissement des simulations, afin de sortir avec des conclusions par rapport aux performances des protocoles TCP New Reno et TCP Vegas.

NB: *Le code utilisé est disponible sur https://github.com/mhamedouadghiri/wmn-sanet*

II. Installation de ns, nam et ns-allinone

A. Installation de NS2.35 et de NAM

L'installation de NS2.35 est réalisée sous Ubuntu 20.04 en suivant les étapes ci-après: [1]

1) sudo apt update
2) sudo apt install gcc build-essential autoconf automake libxmu-dev -y
3) sudo apt install ns2 -y
4) sudo apt install nam -y
5) telecharger *ns-allinone-2.35* depuis ce lien https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/

L'apparition de **%** et de **Nam Console**, en tappant *ns* et *nam* respectivement, nous prouve que l'installation s'est bien passée. (fig. 1)

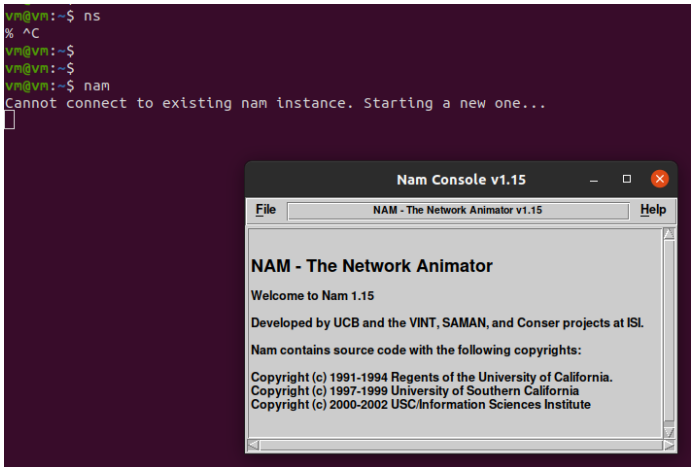

Fig. 1. Installation réussie

B. Création du fichier sanet.tcl

On commence par dé-zipper le tar téléchargé dans l'étape précédente puis on se met au niveau du répertoire *tcl/ex*. Le fichier qui nous intéresse ici est *wireless-mitf.tcl*. On le copie puis on le simule à l'aide de *ns*. La figure 2 présente ces étapes.

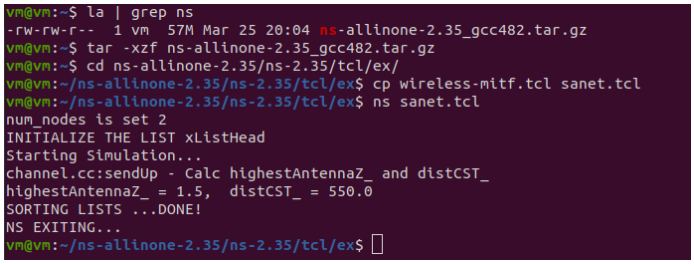

Fig. 2. Création et Simulation de sanet.tcl

III. Déploiement d'une architecture SANET

A. Mise en place de l'architecture

L'architecture SANET qui nous est demandée est la suivante: [2]

- 2 sources (0 et 1)
- 3 gateways (2, 3 et 4)
- 2 destinations (5 et 6)
- 6 cellules de portée max 1000 (chacun des ovales de couleurs différentes)

Nous représentons ceci avec le schéma figure 3.

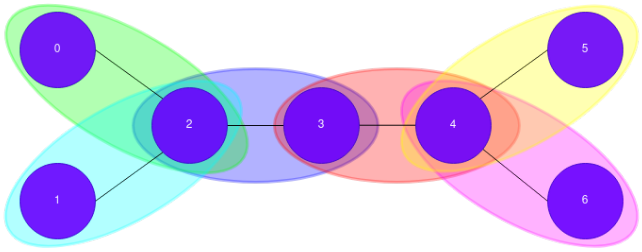

Fig. 3. Architecture SANET

Ensuite on code cette architecture avec le code TCL figures 4, 5 et 6 [3]

```
set val(nn)                     7
set val(rp)                     DSDV
#set val(rp)                    DSR
set val(x)                      1000
set val(y)                      1000
```

Fig. 4. SANET setup: nœuds et portée

```
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]

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

# define some coordinate constants
set x1 100.0
set x2 200.0
set x3 300.0
set x4 400.0
set x5 500.0
set y1 100.0
set y2 200.0
set y3 300.0

$node_(0) set X_ $x1
$node_(0) set Y_ $y3
$node_(0) set Z_ 0.0

$node_(1) set X_ $x1
$node_(1) set Y_ $y1
$node_(1) set Z_ 0.0

$node_(2) set X_ $x2
$node_(2) set Y_ $y2
$node_(2) set Z_ 0.0

$node_(3) set X_ $x3
$node_(3) set Y_ $y2
$node_(3) set Z_ 0.0

$node_(4) set X_ $x4
$node_(4) set Y_ $y2
$node_(4) set Z_ 0.0

$node_(5) set X_ $x5
$node_(5) set Y_ $y3
$node_(5) set Z_ 0.0

$node_(6) set X_ $x5
$node_(6) set Y_ $y1
$node_(6) set Z_ 0.0
```

Fig. 5. SANET setup: positions

B. Simulations des scénarios avec TCP New Reno et TCP Vegas

L'objectif ici est de simuler un transfert FTP grâce aux deux agents: TCP New Reno et TCP Vegas.

```
$ns_ at 0.0 "$node_(0) setdest $x1 $y3 0.0"
$ns_ at 0.0 "$node_(1) setdest $x1 $y1 0.0"
$ns_ at 0.0 "$node_(2) setdest $x2 $y2 0.0"
$ns_ at 0.0 "$node_(3) setdest $x3 $y2 0.0"
$ns_ at 0.0 "$node_(4) setdest $x4 $y2 0.0"
$ns_ at 0.0 "$node_(5) setdest $x5 $y3 0.0"
$ns_ at 0.0 "$node_(6) setdest $x5 $y1 0.0"

# node labels
$ns_ at 0.0 "$node_(0) label Source1"
$ns_ at 0.0 "$node_(1) label Source2"
$ns_ at 0.0 "$node_(2) label Gateway1"
$ns_ at 0.0 "$node_(3) label Gateway2"
$ns_ at 0.0 "$node_(4) label Gateway3"
$ns_ at 0.0 "$node_(5) label Destination1"
$ns_ at 0.0 "$node_(6) label Destination2"
```

Fig. 6. SANET setup: positions(suite) et labels

On fera alors communiquer les nœuds 1 et 6 avec New Reno, et les nœuds 0 et 5 via Vegas.

On distingue 2 scénarios:

- Dans le 1er, on simule les deux communications (Vegas et New Reno) en même temps ie. les nœuds 1/6 et 0/5 émettent et reçoivent des paquets en même temps.
- Dans le 2eme scénario, on effectue les simulations une par une, d'abord New Reno (1/6) et ensuite Vegas (0/5)

1) Réalisation du code: On présente ici, comme exemple, certaines captures du code TCL utilisé. [3]

La figure 7 est un exemple du code New Reno du scénario 1.

```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $tcp
$ns_ attach-agent $node_(6) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at $start_at "$ftp start"
```

Fig. 7. Code New Reno - Scénario 1

La figure 8 est un exemple du code Vegas du scénario 2.

```
set tcp1 [new Agent/TCP/Vegas]
$tcp1 set class_ 2
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 $start_at_1 "$ftp1 start"
$ns_ at $end_at_1 "$ftp1 stop"
```

Fig. 8. Code Vegas - Scénario 2

La différence majeure est qu'au scénario 2 on oblige New Reno à s'arrêter avant que Vegas ne commence, grâce à **$ns_**

at $end_at_1 "$ftp stop".

2) Exécution de la simulation: Une fois le code TCL achevé, on simule à l'aide de *ns*, et on emploie *nam* pour un rendu graphique: **ns sanet.tcl 600 && nam sanet.nam**

Une capture du résultat est présentée à la figure 9.

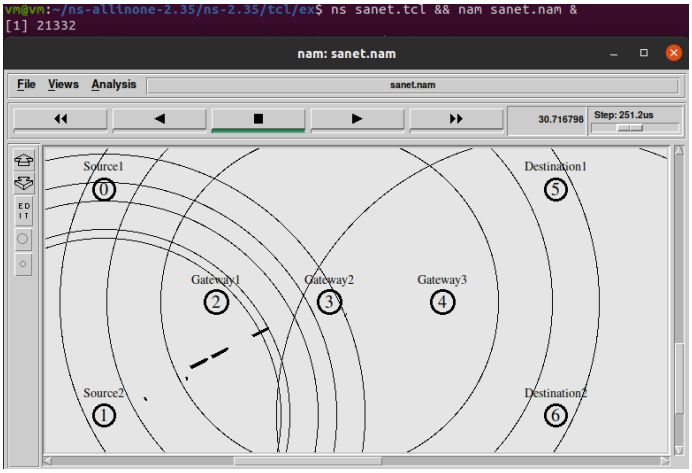

Fig. 9. Capture de l'exécution de la simulation sous NAM

IV. Analyse des résultats

A. Choix d'un bon critère de comparaison

Nous étudions le cas d'une application FTP, l'évaluation de l'efficacité de l'un de l'autre variante de TCP revient à comparer leurs taux de perte. Cette métrique est définie comme suit: **taux de perte = 100 * (1 - reçu / envoyé)**.

B. Fichier trace et script awk

Pour ce faire, on extrait les valeurs *'reçu'* (*received 'r'*) et *'envoyé'* (*sent 's'*) depuis le fichier trace généré par *ns* (*sanet.tr*) en utilisant un script *awk*.

La figure 10 présente un extrait du fichier trace, en particulier les lignes qui nous intéressent (TCP). [4] [5]

```
vm@vm:~/ns-allinone-2.35/ns-2.35/tcl/ex$ grep tcp sanet.tr | head
s 3.000000000 _1_ AGT  --- 7 tcp 40 [0 0 0 0] ------- [1:0 6:0 32 0] [0 0] 0 0
r 3.000000000 _1_ RTR  --- 7 tcp 40 [0 0 0 0] ------- [1:0 6:0 32 0] [0 0] 0 0
s 3.000000000 _0_ AGT  --- 8 tcp 1000 [0 0 0 0] ------- [0:0 5:0 32 0] [0 0] 0 0
r 3.000000000 _0_ RTR  --- 8 tcp 1000 [0 0 0 0] ------- [0:0 5:0 32 0] [0 0] 0 0
s 6.000000000 _1_ AGT  --- 9 tcp 40 [0 0 0 0] ------- [1:0 6:0 32 0] [0 0] 0 0
r 6.000000000 _1_ RTR  --- 9 tcp 40 [0 0 0 0] ------- [1:0 6:0 32 0] [0 0] 0 0
s 6.000000000 _0_ AGT  --- 10 tcp 1000 [0 0 0 0] ------- [0:0 5:0 32 0] [0 0] 0 0
r 6.000000000 _0_ RTR  --- 10 tcp 1000 [0 0 0 0] ------- [0:0 5:0 32 0] [0 0] 0 0
s 6.000000000 _0_ AGT  --- 11 tcp 1000 [0 0 0 0] ------- [0:0 5:0 32 0] [1 0] 0 0
r 6.000000000 _0_ RTR  --- 11 tcp 1000 [0 0 0 0] ------- [0:0 5:0 32 0] [1 0] 0 0
```

Fig. 10. sanet.tr excerpt

C. Script awk pour le calcul des taux

La figure 11 représente le script awk [6], qui permet de compter les paquets envoyés et reçus par New Reno et par Vegas, concernant le protocole **TCP** uniquement, et ayant comme *trace level* **AGT** (*MAC, RTR, IFQ... ne nous intéressent pas*).

Un exemple d'exécution (ns puis awk) est présenté figure 12.

```
BEGIN {
  rcv_16_reno = 0
  sent_16_reno = 0
  rcv_05_vegas = 0
  sent_05_vegas = 0
}
{
  if ($7 == "tcp" && $4 == "AGT") {
    if ($3 == "_0_" && $1 == "s") sent_05_vegas++
    if ($3 == "_1_" && $1 == "s") sent_16_reno++
    if ($3 == "_5_" && $1 == "r") rcv_05_vegas++
    if ($3 == "_6_" && $1 == "r") rcv_16_reno++
  }
}
END {
  loss_rate_reno = 100 * (1 - rcv_16_reno / sent_16_reno)
  loss_rate_vegas = 100 * (1 - rcv_05_vegas / sent_05_vegas)

  printf("NewReno sent = %d, rcv  = %d, loss rate = %f\n",
         sent_16_reno, rcv_16_reno, loss_rate_reno)
  printf("Vegas sent = %d, rcv = %d, loss rate = %f\n",
         sent_05_vegas, rcv_05_vegas, loss_rate_vegas)
}
```

Fig. 11. Script awk

```
vm@vm:~/sanet$ ns sanet.tcl 600 && awk -f awk_pp.awk sanet.tr
num_nodes is set 7
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
NS EXITING...
NewReno sent = 20761, rcv  = 20745, loss rate = 0.077068
Vegas sent = 2619, rcv = 2611, loss rate = 0.305460
```

Fig. 12. Exécution ns puis awk

D. Simulations multiples

Afin d'obtenir des résultats exploitables, on effectue la simulation à plusieurs reprises en changeant la durée d'exécution au niveau du code TCL. Ensuite nous prenons note des taux de perte calculés, et nous utilisons matplotlib avec python (fig 13) pour de tracer des courbes représentant le taux de perte de chacun des deux variantes de TCP en fonction de la durée de la simulation. Et ce pour chaque scénario.

```
plt.plot(time, newreno, 'bo-', label="NewReno S1")
plt.plot(time, newrenoObO, 'bo:', label="NewReno S2")
plt.plot(time, vegas, 'ro-', label="Vegas S1")
plt.plot(time, vegasObO, 'ro:', label="Vegas S2")
```

Fig. 13. Python script excerpt

Changer de durée à chaque simulation et pour chaque scénario est une tâche fastidieuse, nous avons alors écrit un script shell **simMul.sh** pour automatiser la tâche (extrait fig. 14) en tapant simplement **./simMul.sh sanet 10 600 200** et **./simMul.sh sanetObO 10 600 200**

Les résultats finaux obtenus sont présentés figure 15.

NB:

- S1 == Setup 1 == Scénario 1 == en même temps
- S2 == Setup 2 == Scénario 2 == un par un

```
for i in `seq 0 $nb_simul`
do
  current=$(($base_duration + $i * $interval))
  echo "------------------------------ start $

  ns $tcl $current &> /dev/null

  awkOut=$(awk -f awk.awk $tr)

  nr=$(sed -n 1p <<< $awkOut)
  v=$(sed -n 2p <<< $awkOut)
```

Fig. 14. Shell script excerpt

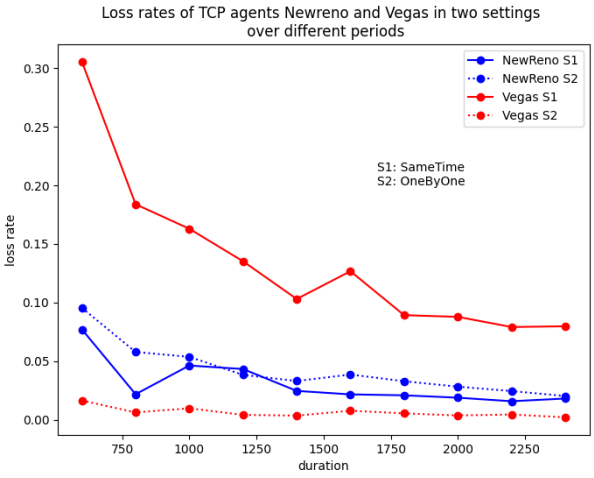

Fig. 15. Tracé des courbes des taux d'erreur

E. Analyse des résultats

On soulève plusieurs remarques à travers les courbes tracées:

- Les valeurs de S1 sont généralement supérieures à celles de S2.
- Dans la config S1, Vegas donne des taux de perte nettement supérieurs à ceux de New Reno.
- Dans la config S2, les taux de pertes de New Reno sont légèrement supérieurs à ceux de Vegas.

A partir de cela, on déduit que dans un environnement où les deux agents sont utilisés en même temps (S1), et donc conséquemment où les gateways 2, 3 et 4 (selon notre architecture fig. 3) sont partagés en même temps pour la communication 1-6 (New Reno) et 0-5 (Vegas); nous trouvons que New Reno donne de meilleurs résultats avec des taux de perte nettement inférieurs à ceux de Vegas, ceci peut indiquer que New Reno s'empare des gateways communs pour effectuer sa communication entre 1 et 6 et bloque Vegas qui n'arrive pas à prendre la main sur les gateways aussi agressivement que New Reno. D'un autre côté, dans le scénario 2, lors de l'évaluation de chaque variante TCP seule de son côté, Vegas prend le dessus avec des taux de perte inférieurs à New Reno, mais plus encore, Vegas exploite l'architecture SANET proposée plus efficacement que New Reno comme le montre le nombre de paquets envoyés sur la figure 16 (97640 pour Vegas contre seulement 93884 pour New Reno).

```
vm@vm:~/sanet$ ns sanetOb0.tcl 2400 && awk -f awk_pp.awk sanetOb0.tr
num_nodes is set 7
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
NS EXITING...
NewReno sent = 93884, rcv  = 93865, loss rate = 0.020238
Vegas sent = 97640, rcv = 97638, loss rate = 0.002048
```

Fig. 16. Meilleure exploitation de Vegas que New Reno dans une environnement séparée

V. Conclusion

La variante New Reno de TCP est plus agressive que Vegas, unfair [7], dans le sens où elle prend le dessus dans un environnement partagé en occupant les gateways communs (bottleneck links [7]) plus que son homologue ce qui lui permet d'assurer un taux de perte inférieur et donc meilleur. Mais dans des environnements séparés, Vegas marche mieux en exploitant les nœuds plus que New Reno (quantité envoyée supérieure) tout en assurant un meilleur envoi des paquets [8] ce qui résulte en un taux de perte inférieur et donc meilleur pour Vegas.

Vegas est plus efficace si elle est déployée seule; New Reno est plus efficace dans un environnement mixte.

Ces conclusions sont conformes aux résultats trouvés dans la littérature. [7] [8] [9]

Annexe

Tout le code utilisé est disponible sur https://github.com/mhamedouadghiri/wmn-sanet

References

[1] D. R. Sahana K. Bhosale, "Simulation of vertical handover between wifi and wimax and its performance analysis – an installation perspective."

[2] Simulation d'une architecture sanet en utilisant ns2. prof. amine berquia. [Online]. Available: https://moodle-ensias.um5.ac.ma/pluginfile.php/84066/mod_resource/content/1/TCPoverSANET.pdf

[3] Network simulator (ns) prof. nelson l. s. da fonseca. [Online]. Available: https://ic.unicamp.br/ nfonseca/MO648/doc/curso_ns_comentarios.pdf

[4] Trace files and description. [Online]. Available: https://ns2blogger.blogspot.com/p/the-file-written-by-application-or-by.html

[5] Ns-2 trace formats. [Online]. Available: http://nsnam.sourceforge.net/wiki/index.php/NS-2_Trace_Formats

[6] The gnu awk user's guide. [Online]. Available: https://www.gnu.org/software/gawk/manual/gawk.html

[7] Fairness comparisons between tcp reno and tcp vegas for future deployment of tcp vegas. [Online]. Available: http://www.isoc.org/inet2000/cdproceedings/2d/2d_2.htm

[8] Y. Been Seok, B. Ong, and R. Ahmad, "Performance evaluation of tcp vegas versus different tcp variants in homogeneous and heterogeneous wired networks," *World Academy of Science, Engineering and Technology*, vol. 50, pp. 175–181, 02 2011.

[9] A comparative analysis of tcp tahoe, reno, new-reno, sack and vegas. [Online]. Available: https://inst.eecs.berkeley.edu/ ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf