TP1 - Simulation de réseaux simples avec ns-2

But du TP

Le simulateur de réseaux ns-2 permet d'analyser les réseaux et d'évaluer leurs performances. Ce type de simulateur est très utilisé en recherche et en R&D afin d'évaluer de nouvelles solutions avant de les intégrer dans des réseaux réels.

Le but de ce TP est de se familiariser avec l'outil de simulation de réseaux ns-2 :

(1) en composant quelques scripts TCL simples,

(2) en analysant les traces produites par les expériences simulées sur des réseaux simples,

ainsi que d'étudier certaines performances réseaux.

Utiliser ns-2 en salle de TP

Il faut un minimum de 3 étapes pour réaliser une simulation avec ns-2 sur votre poste :

1. Composer votre script ns avec votre éditeur de texte favori et sauvegarder le dans un fichier portant l'extension `.tcl`
2. Exécuter le. Pour cela, il suffit de saisir dans un terminal la commande suivante : `ns nomDuPrograme.tcl` (ou `ns < nomDuPrograme.tcl`)
3. Accéder aux fichiers contenant les traces issus de la simulation (`out.tr` et `out.nam`). Ils sont situés dans le répertoire courant. Consulter les de préférence depuis un terminal avec les commandes `more` ou `less`.

Assurez vous que la variable d'environnement `LC_NUMERIC` n'est pas fixée à ',' (si elle existe elle doit être fixée à '.'). Si ce n'est pas le cas, exécutez la commande `export LC_ALL=POSIX` pour forcer POSIX comme standard pour votre terminal.

Tous les délais demandés dans ce TP seront exprimés en ms.

1 Un seul flux UDP (1 heure)

Nous considérons une topologie très simple comprenant deux nœuds reliés par un lien direct. Un trafic UDP circule du premier nœud vers le second. Nous considérons également que :

- le lien est full-duplex de capacité 1 Mb/s avec un délai de propagation de 10 ms et un buffer de type "DropTail" ;

- le trafic UDP est de type CBR ("Constant Bit Rate") avec les paramètres suivants : taille des paquets (packetSize_) = 1000 octets et durée entre deux émissions (transmissions) de paquets (interval_) = 5 ms ;
- le flux UDP démarre à t=0.5 sec et s'arrête à t=4.5 sec.

La figure 1 récapitule le scénario à simuler.

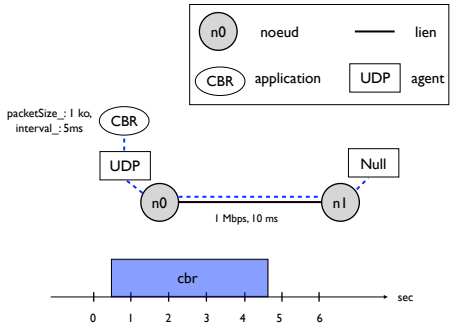

FIGURE 1 – Scénario à simuler pour l'exercice "Un seul flux UDP".

Travail demandé :

1. Exécuter le script ns proposé en annexe vous permettant de réaliser cette expérience.
2. Visualiser l'expérience avec l'outil NAM. *(si NAM ne fonctionne pas, cela ne vous empêche pas de faire la suite du TP)*
3. Rappeler ce qu'est un lien full-duplex. Le lien de communication est configuré pour appliquer une politique Drop Tail. Qu'est-ce qu'une politique Drop Tail ?

Réponse :

4. Calculer "à la main" le délai de bout-en-bout pour un paquet du trafic UDP et le débit applicatif du flux UDP.

Réponse :

5. Explorer et interpréter le contenu du fichier de traces *out.tr*. Vous pouvez vous reporter à la documentation de ns (`http://www.isi.edu/nsnam/ns/doc/node289.html`) pour comprendre le format de ce fichier. Vous trouverez également en annexe un descriptif de chacun de ces champs ainsi qu'un exemple de fichier de traces (cf. figures 3 et 4).

6. D'après le fichier de traces, déterminer l'instant de génération du paquet du trafic UDP ayant le numéro de séquence 10, puis l'instant de son début d'émission sur le lien.

Réponse :

7. Toujours pour le 10ème paquet UDP, déterminer également son temps d'attente dans le nœud $n0$, puis la somme de son temps d'émission (transmission) sur le lien et de son délai de propagation [1].

Réponse :

8. En déduire le délai de bout-en-bout du 10ème paquet. Cette valeur correspond-elle à celle calculée à la question 3 ? Si non pourquoi ?

Réponse :

2 Cohabitation d'un flux UDP avec un flux TCP (1 heure)

On considère le scénario représenté sur la figure 2. Tous les liens de communication du scénario sont full-duplex et utilisent un buffer de type Drop Tail. Les nœuds $n0$ et $n1$ génèrent du trafic vers le nœud n3. Leur trafic passent par le nœud $n2$ qui le relaie vers $n3$. Le trafic émis par le nœud $n0$ est généré par une source FTP qui utilise le protocole de transport TCP (se reporter à l'Annexe pour déclarer un flux TCP). Le trafic émis par le nœud $n1$ est généré par une source CBR qui utilise lui le protocole de transport UDP. Les paramètres des trafics UDP et TCP sont indiqués sur la figure 2. Dans cette expérience, le flux UDP est actif entre t=1 et t=5 sec tandis que le flux TCP lui est actif entre t=2 et t=4 sec. **Attention, notez bien que c'est l'agent TCP qui définit la taille des segments (et seulement la taille), tandis que pour UDP, c'est l'application qui définit la taille des segments et le débit auquel ils seront émis. Par ailleurs, le traffic CBR est caractérisé ici par les paramètres packetSize_ et rate_ (et non plus interval_).**
Travail demandé :

1. Composer le script ns pour simuler cette expérience, puis exécuter le.
2. Vérifier que votre expérience est correcte en vous assurant que l'instant de génération du paquet UDP ayant le numéro de séquence 211 a lieu à l'instant 2.688s.
3. Visualiser votre simulation avec NAM. *(si NAM ne fonctionne pas, cela ne vous empêche pas de faire la suite du TP)*

À présent tout le travail se fera à partir du fichier de traces `out.tr`.

4. Pourquoi un paquet a-t-il deux numéros dans la trace ?

1. Dans le simulateur ns le temps de traitement d'un paquet à chaque nœud est toujours nul.

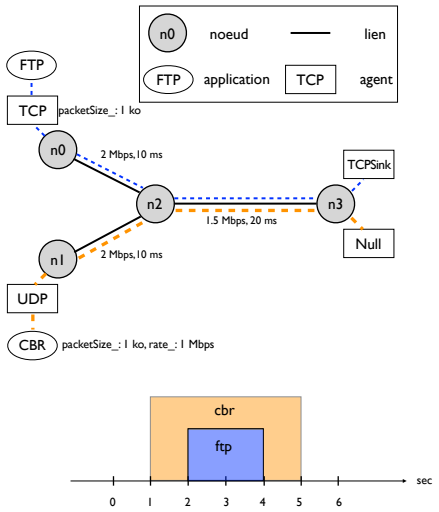

FIGURE 2 – Scénario à simuler pour l'exercice "Un flux UDP avec un flux TCP".

Réponse :

5. Est-ce que la taille des paquets de données TCP indiquée dans la trace est cohérente avec la taille que vous avez fixée dans le script ? Justifier votre analyse.

Réponse :

6. Est-il possible de "lire" le délai d'émission d'un paquet TCP sur un lien uniquement à partir de la trace ? Même question pour un paquet UDP ?

Réponse :

7. Trouver une commande simple sur votre terminal permettant d'afficher tous les événements d'un paquet donné (par exemple le paquet dont l'identifiant est 100).

Réponse :

8. Pour le premier paquet TCP généré par $n0$ après t= 3 sec, déterminer son délai d'attente dans chacun des nœuds qu'il traverse.

Réponse :

9. Ces délais d'attente seront-ils les mêmes pour tous les paquets TCP ? Comment l'expliquer ?

Réponse :

10. Est-ce que les paquets UDP auront aussi un délai d'attente sur le nœud $n2$?

Réponse :

3 Performances d'un flux (1 heure)

Le but de cet exercice est de comprendre comment exploiter les traces d'une simulation afin d'en extraire des mesures sur les performances moyennes d'un flux. Pour cela, nous considérons de nouveau le scénario de l'exercice 2 (cf. figure 2). Sur le lien reliant le nœud n2 au nœud n3, on positionne un buffer de type "DropTail" et de longueur égale à 10 (commande : `$ns queue-limit $n2 $n3 10`) [2]. Il est possible de visualiser avec NAM l'état d'occupation du buffer sur le lien entre n2 et n3 en incluant dans votre script ns la commande `$ns duplex-link-op $n2 $n3 queuePos 0.5`.

Sur cette topologie, le lien entre n2 et n3 représente le goulet d'étranglement, c'est-à-dire l'endroit où les paquets seront le plus "ralentis". Enfin, puisque le buffer sur le lien entre n2 et n3 est de taille finie, des pertes pourront se produire si des paquets tentent d'entrer dans le buffer alors qu'il est plein.

Le premier objectif de cet exercice sera de calculer le taux de perte des paquets du flux UDP sur la période de temps de 1 à 2 sec et puis de comparer cette valeur à celle obtenue sur la période de 2 à 4 sec.

Le deuxième objectif sera d'évaluer le temps que prend en moyenne un paquet du flux UDP pour aller de n1 à n3 sur la période de temps de 1 à 2 sec et de comparer cette valeur à celle obtenue sur la période de 2 à 4 sec.

Travail demandé :

2. Par défaut, la taille des buffers sur les liens est limitée à 50. Attention, cette commande ne fixe que la taille du buffer de n2 vers n3 (et pas de n3 vers n2 !)

1. Composer le script ns pour simuler cette expérience, puis exécuter le.
2. Visualiser votre simulation avec NAM. *(si NAM ne fonctionne pas, cela ne vous empêche pas de faire la suite du TP)*
3. Concevoir une procédure théorique permettant de calculer le taux de paquets UDP perdus à partir du fichier de traces généré pendant le temps compris entre t1 et t2.

Réponse :

4. Implémenter cette procédure dans un script Awk (une courte introduction à Awk est donnée en annexe) ou Perl et noter le résultat obtenu pour {t1,t2}={1s, 2s}, {t1,t2}={2s, 3s} et {t1,t2}={3s, 4s}.

Réponse :

5. Concevoir une procédure permettant de calculer le délai moyen de bout-en-bout des paquets UDP (en prenant en compte le temps d'attente sur le premier nœud) pendant le temps compris entre t1 et t2 à partir du fichier de traces généré. Aide : faire 2 tableaux séparés indiquant les instants de départs et d'arrivées des paquets et indexés par leur numéro de séquence. Attention aux paquets émis avant t1 mais reçus après t1.

Réponse :

6. Implémenter cette procédure dans un script Awk ou Perl et noter le résultat obtenu pour {t1,t2}={1s, 2s}, {t1,t2}={2s, 3s} et {t1,t2}={3s, 4s}. Ces résultats vous paraissent-ils cohérents avec le délai de bout-en-bout estimé théoriquement ? Comment expliquer les écarts ?

Réponse :

7. Recommencer l'exercice mais cette fois-ci avec un buffer de taille 100. Qu'observez-vous ? Comment expliquer l'évolution des résultats ?

Réponse :

Annexes

Script proposé pour le premier exercice

```
# instantiation du simulateur
set ns [new Simulator]

# associer les couleurs pour NAM
$ns color 1 green
$ns color 2 red
$ns color 3 blue

# instantiation du fichier de traces
set file1 [open out.tr w]
$ns trace-all $file1

# instantiation du fichier de traces pour NAM
set file2 [open out.nam w]
$ns namtrace-all $file2

# procédure pour lancer automatiquement le visualisateur NAM à la fin de la simulation
proc finish {} {
     global ns file1 file2
     $ns flush-trace
     close $file1
     close $file2
     exec nam out.nam &
     exit 0
}

# instantiation de deux noeuds
set n0 [$ns node]
set n1 [$ns node]

# instantiation d'une ligne de communication full duplex entre les noeuds n0 et n1
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

# instantiation d'une connexion UDP
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n1 $null
$ns connect $udp $null
$udp set fid_ 1  # permet d'associer un identifiant a ce flux
```



```
# instantiation d'un trafic CBR composé de paquets de 1000 octets, générés toutes les 5 ms
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp
$cbr0 set packetSize_ 1000
$cbr0 set interval_ 0.005

# scénario de début et de fin de génération des paquets par cbr0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

# la simulation va durer 5 secondes de temps simulé
$ns at 5.0 "finish"

# début de la simulation
$ns run
```

Déclarer un flux TCP

On utilisera les commandes suivantes pour déclarer un flux TCP ici entre les noeuds n0 et n3.

```
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
```

Exemple d'un fichier trace

Le fichier contenant les traces d'une simulation ns est un fichier ascii organisé en ligne de 12 champs comme le montre la figure 3. La signification de chacun de ces champs est la suivante :

1. Le premier champ indique le type de l'événement. 5 symboles sont possibles `r`, `+`, `-`, `d`, `e`. Ils signifient respectivement que :

pour `r`	le paquet a été reçu à la sortie du lien (au noeud suivant)
pour `+`	le paquet a été mis en attente dans un buffer (celui de l'interface de sortie vers le noeud suivant)
pour `-`	le paquet a quitté le buffer (correspond au début de l'émission sur le lien)
pour `d`	le paquet a été rejeté du buffer (correspond à une perte)
pour `e`	le paquet a subi une erreur (correspond à une erreur binaire)
pour `c`	le paquet entre en collision au niveau couche MAC
pour `h`	dans le contexte d'un LAN, le paquet a atteint le noeud "virtuel"

   Ainsi, le délai d'attente se mesure entre un `+` et `-`, le délai d'émission (transmission) combiné au délai de propagation entre `-` et `r` et enfin le délai de traitement s'il existait entre `r` et `+`.
2. Ce champ indique l'instant auquel l'événement s'est produit.
3. Ce champ indique le noeud source sur le lien (adresse couche 2)
4. Ce champ indique le noeud destination sur le lien (adresse couche 2)
5. Ce champ indique le type du paquet (par exemple CBR ou TCP)

6. Ce champ indique la taille du paquet (avec les en-têtes des couches 3 et 4)
7. Ce champ indique quelques drapeaux (“flags”)
8. Ce champ indique l’identifiant du flux. Cette identifiant peut être fixé dans le script ns et permet notamment de colorer les fluxs avec NAM
9. Ce champ indique l’adresse source globale désignée dans le format “noeud.port” (adresse couche 3)
10. Ce champ indique l’adresse destination globale (même format) (adresse couche 3)
11. Ce champ indique le numéro de séquence du paquet tel qu’il est donné par le protocole de transport.
12. Ce dernier champ indique un identifiant unique du paquet pour toute la simulation.

Evénement	Temps	Du noeud	Vers le noeud	Type paquet	Taille paquet	Flags	FID	Adresse SRC	Adresse DST	Numéro séquence	ID paquet

FIGURE 3 – Format d’une trace ns.

La figure 4 montre les premières lignes d’un fichier de traces produit par un script ns.

+	0.1	1	2	cbr	1000	-------	2	1.0	5.0	0	0
-	0.1	1	2	cbr	1000	-------	2	1.0	5.0	0	0
r	0.114	1	2	cbr	1000	-------	2	1.0	5.0	0	0
+	0.114	2	3	cbr	1000	-------	2	1.0	5.0	0	0
-	0.114	2	3	cbr	1000	-------	2	1.0	5.0	0	0
r	0.240	2	3	cbr	1000	-------	2	1.0	5.0	0	0
+	0.240	3	5	cbr	1000	-------	2	1.0	5.0	0	0
-	0.240	3	5	cbr	1000	-------	2	1.0	5.0	0	0
r	0.286	3	5	cbr	1000	-------	2	1.0	5.0	0	0
+	0.9	1	2	cbr	1000	-------	2	1.0	5.0	1	1
-	0.9	1	2	cbr	1000	-------	2	1.0	5.0	1	1
r	0.914	1	2	cbr	1000	-------	2	1.0	5.0	1	1
+	0.914	2	3	cbr	1000	-------	2	1.0	5.0	1	1
-	0.914	2	3	cbr	1000	-------	2	1.0	5.0	1	1
+	1	0	2	tcp	40	-------	1	0.0	4.0	0	2
-	1	0	2	tcp	40	-------	1	0.0	4.0	0	2
r	1.01	0	2	tcp	40	-------	1	0.0	4.0	0	2
+	1.01	2	3	tcp	40	-------	1	0.0	4.0	0	2
-	1.01	2	3	tcp	40	-------	1	0.0	4.0	0	2
r	1.04	2	3	cbr	1000	-------	2	1.0	5.0	1	1
+	1.04	3	5	cbr	1000	-------	2	1.0	5.0	1	1
-	1.04	3	5	cbr	1000	-------	2	1.0	5.0	1	1
r	1.08	3	5	cbr	1000	-------	2	1.0	5.0	1	1

FIGURE 4 – Exemple de trace enregistrée.

Généralités sur Awk

Awk est un programme qui permet de chercher dans des fichiers des lignes ou portion de texte contenant un ou des patrons et d’y effectuer des opérations. Il y a au moins deux façons de lancer awk. (1) Directement le lancer depuis la ligne de commande : `awk -f <script_awk> out.tr` ou (2) en mémorisant les commandes awk dans un script : `cat out.tr | awk -f <script_awk>`. Note, pour passer un argument en ligne de commande à awk, on utilise l’option `-v <nom_variable>=<valeur_variable>`.

Quand awk lit une ligne d’un fichier, il divise la ligne en champs basé sur le séparateur de champs en entrée, FS, qui est une variable awk. Cette variable est prédéfinie avec un ou plusieurs espaces et tabulations. Les variables $1, $2, $3, ..., $N stockent les valeurs du premier, second, troisième jusqu’au dernier champs de la ligne traitée. La variable $0 stocke la valeur de la ligne entière.

Un programme awk se compose comme une suite de blocs de code compris entre {} qui sont appliqués aux lignes de données du fichier si un patron est vérifié. Voici quelques types de patron que l’on peut avoir :

BEGIN	Le bloc qui suit est exécuté au début du programme, une seule fois.
END	Le bloc qui suit est exécuté à la fin du programme, une seule fois.
expression relationnelle	Le bloc qui suit est exécuté si l’expression est vraie. Par exemple, ma-Variable==5
modèle && modèle	Le bloc qui suit est exécuté si les deux modèles sont vérifiés.
modèle || modèle	Le bloc qui suit est exécuté si au moins un des modèles est vérifié.
! modèle	Le bloc qui suit est exécuté si le modèle n’est pas vérifié.

Voici un exemple `incomplet` d’un script awk qui peut vous servir de modèle :

```
# les commentaires dans un script awk se trouvent après un caractère #
# attention, la variable d'environnement LC_NUMERIC doit être fixée à '.'
 BEGIN {
}

{
    event = $1;
    time = $2; # en secondes
    src_H2 = $3;
    dst_H2= $4;
    type = $5;
    size = $6; # en octets
    idflow = $8;
    src_H3=$9;
    dst_H3=$10;
    idseq=$11;
    idpaquet = $12;

######## VOTRE CODE ICI ########
    if ( event  == "d" ) {
        loss++;
    }
################################
}

END {
     printf("le taux de perte vaut %g, en effet %d paquets ont ete perdus
 parmi les %d envoyes\n", loss/pkt_send, loss, pkt_send);
}
```