Internet Nouvelle Génération

Module de M1 de la mention Informatique
Spécialité Réseaux, 2e semestre 2007 - 2008
Université Pierre et Marie Curie

Supports de TME

TME 1
TME 2

Bénédicte LE GRAND

TME ING

NS : Network Simulator

Copiez le répertoire /users/Enseignants/blegrand/TPing sur votre compte.

NS est un outil logiciel de simulation de réseaux informatiques. Il a été conçu autour des idées de conception par objets, de réutilisabilité du code et de modularité. Il est devenu aujourd'hui un standard de référence en ce domaine. C'est un logiciel dans le domaine public disponible sur l'Internet. Son utilisation est gratuite. Le logiciel est exécutable tant sous Unix que sous Windows.

NS est bâti autour d'un langage de programmation appelé Tcl dont il est une extension. Du point de vue de l'utilisateur, la mise en œuvre de ce simulateur se fait via une étape de programmation qui décrit la topologie du réseau et le comportement de ses composants, puis vient l'étape de simulation proprement dite et enfin l'interprétation des résultats. Cette dernière étape peut être prise en charge par un outil annexe, appelé nam qui permet une visualisation et une analyse des éléments simulés. Au cours de ce TME, nous allons montrer comment utiliser le simulateur avec les composants disponibles dans la distribution.

NS est en réalité un programme relativement complexe écrit en C++ et interfacé via Tcl. Pour modifier le comportement d'objets existants, il est donc nécessaire de modifier le code C++ qui en réalise l'implantation.

Dans le cadre de ce TME, nous nous en tiendrons au lancement et à la modification de scripts tcl, sans modification du code C++.

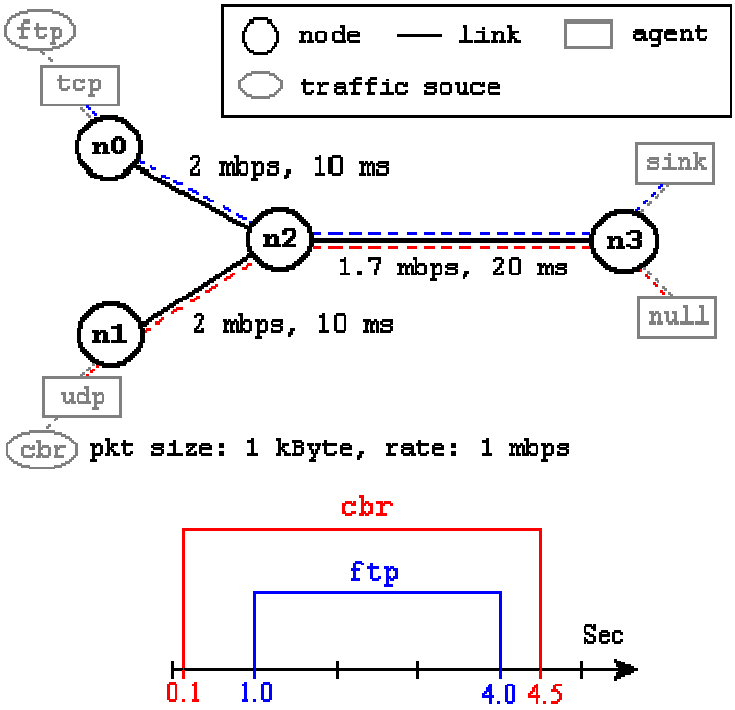
Introduction à NS : premier exemple de simulation

Cette section montre un script de simulation NS simple (ns-simple.tcl) et explique à quoi chaque ligne correspond. Il s'agit d'un script OTcl qui crée une configuration de réseau simple et lance le scénario de simulation de la figure ci-dessous. Pour lancer la simulation, tapez la commande :

```
ns ns-simple.tcl
```

Dans la terminologie NS, ce que nous appelons machine s'appelle un nœud. Un nœud peut contenir des agents qui représentent des comportements, par exemple des applications. Une bibliothèque assez complète de composants existe de façon standard. Une propriété intéressante de ce système est son extensibilité. En effet, il est assez facile d'étendre la bibliothèque des comportements, des types de liens, ou de tout autre élément du système en programmant ses propres extensions qui deviennent alors intégrées au système.

Lors de cette simulation, les résultats sont consignés dans un fichier de trace que l'outil de visualisation nam va permettre de traiter. Dans notre programme, l'outil de visualisation est appelé directement à la fin de la simulation. Deux éléments intéressants sont proposés à la visualisation : un dessin de la topologie du réseau étudié et une visualisation dynamique du déroulement du programme dans le temps.



Ce réseau contient 4 noeuds (n0, n1, n2, n3), comme le montre la figure ci-dessus.

Les liens bi-directionnels entre n0 et n2 et entre n1 et n2 ont une bande passante de 2 Mbps et un délai de propagation de 10 ms. Le lien bi-directionnel entre n2 et n3 a une bande passante de 1.7 Mbps et un délai de propagation de 20 ms. Chaque nœud utilise une file d'attente DropTail, dont la taille maximale est 10.

Un agent "tcp" est attaché au noeud n0 et une connexion est établie vers un agent tcp "sink" attaché à n3. Par défaut, la taille maximale d'un paquet généré par un agent tcp est de 1000 octets. Un agent tcp "sink" génère et envoie les paquets d'acquittement (ACK) à l'émetteur (l'agent tcp) et libère les paquets reçus.

Un agent udp attaché à n1 est connecté à un agent "null" attaché à n3. Un agent "null" libère simplement les paquets reçus.

Des générateurs de trafic "ftp" et "cbr" sont attachés respectivement aux agents "tcp" et "udp". Le "cbr" est configuré pour générer des paquets de 1000 octets au débit de 1 Mbps. Le "cbr" commence à l'instant 0.1 s et s'arrête à 4.5 s, et le "ftp" commence à 1.0 s et s'arrête à 4.0 s.

Dans le script ns-simple.tcl, les commentaires (introduits par le caractère #) expliquent le rôle de chaque instruction ou partie de programme.

```
#ns-simple.tcl

# création d'un simulateur
set ns [new Simulator]

# définition de différentes couleurs pour les flots de données #(pour
NAM)
$ns color 1 Blue
$ns color 2 Red

# création du fichier de trace utilisé par le visualiseur
# et indication à ns de l'utiliser
set nf [open out.nam w]
$ns namtrace-all $nf

# lorsque la simulation est terminée, cette procédure est appelée
# pour lancer automatiquement le visualiseur
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

# création de quatre noeuds
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# création de liens de communication full duplex entre les noeuds
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

# taille de la file d'attente du lien (n2-n3) fixée à 10
$ns queue-limit $n2 $n3 10

# position des noeuds (pour NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

```
# monitoring de la file d’attente pour le lien (n2-n3) (pour NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

# mise en place d’une connexion TCP
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

# Mise en place d’un FTP sur la connexion TCP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

# Mise en place d’une connexion UDP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

# Mise en place d’un CBR sur la connexion UDP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

# définition des événements pour les agents CBR et FTP
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

# détachement des agents tcp et sink (pas vraiment nécessaire)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

# appel de la procédure finish après 5 secondes de temps de #simulation
$ns at 5.0 "finish"

# affichage de la taille des paquets CBR et de l’intervalle
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

# lancement de la simulation
$ns run
```

1. Utilisation de composants standards

Le script ex1.tcl permet de simuler un réseau très simple. Pour lancer la simulation, tapez la commande :

```
ns ex1.tcl
```

Question 1 : Quelle est la topologie du réseau simulé ?
Question 2 : Quelles sont les caractéristiques du lien de communication ?
Question 3 : Quel agent est attaché à chaque nœud ? Justifiez le choix de l'agent attaché au nœud 1.
Question 4 : De quel type d'application s'agit-il ?
Question 5 : Quelle est la taille des paquets envoyés ?
Question 6 : A quel débit les paquets sont-ils envoyés ?
Question 7 : Combien de temps dure la simulation ?
Question 8 : Pendant combien de temps du trafic est-il envoyé sur le lien ?
Question 9 : Décrivez l'interface nam. Quels paramètres pouvez-vous faire varier ?

Considérons à présent le script ex2DropTail.tcl.

Question 10 : Quelle est la topologie du réseau (nœuds, liens) ? Vous pouvez faire un schéma.
Question 11 : Combien y a-t-il de sources de trafic ? Décrivez chacune des applications (source, destination, type, débit, instant de début et de fin)
Question 12 : Observez la file d'attente de messages qui se forme au nœud 2 (puisque les trafics en entrée sont assez proches de la saturation). Que constatez-vous entre qui concerne la gestion de la file d'attente à l'entrée du lien 2 vers 3 ?

Considérons à présent le script ex2SFQ.tcl.

Question 13 : La différence avec le script précédent réside dans la gestion poilitique de la file d'attente à l'entrée du lien 2 vers 3. Quelle est la ligne qui a été modifiée ?
Question 14 : Quelle est la conséquence de l'utilisation de la politique SFQ (Stochastic Fair Queuing) ?

Nous nous intéressons à présent au script ex3.tcl, qui crée une topologie circulaire de réseau, génère un trafic et fait apparaître des pannes sur l'un des liens.

Question 15 : Quelle est la topologie du réseau (nœuds, liens) ? Vous pouvez faire un schéma.
Question 16 : Combien y a-t-il de sources de trafic ? Décrivez chacune des applications (source, destination, type, débit, instant de début et de fin)
Question 17 : Quel est le type de routage utilisé ?
Question 18 : Quel est le lien qui tombe en panne ? Pendant quelle durée ce lien est-il défaillant ?
Question 19 : Quelle est la conséquence de la panne du lien ?

2. Connexions TCP et CBR

Considérons le script ex4.tcl.

Question 20 : Quelle est la topologie du réseau (nœuds, liens) ? Vous pouvez faire un schéma.
Question 21 : Combien y a-t-il de sources de trafic ? Décrivez chacune des applications (source, destination, type, débit, instant de début et de fin)
Question 22 : Quels sont les fichiers générés par cette simulation ?

Nous allons à présent étudier comment les traces peuvent être analysées afin d'en extraire des données visualisables. Le format des fichiers de trace est le suivant :

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)              src_addr : node.port (3.0)
- : dequeue (at queue)              dst_addr : node.port (0.0)
d : drop    (at queue)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

Le fichier WinFile, généré lors de la simulation, représente la taille de la fenêtre pour la connexion TCP en fonction du temps. Pour visualiser ces données, on peut utiliser gnuplot, de la manière suivante :

```
gnuplot
set size 0.4,0.4
plot 'WinFile' w lines 1
```

Question 23 : Dessinez l'allure de la courbe obtenue en visualisant le fichier WinFile avec gnuplot.
Question 24 : Comment expliquez-vous les variations de la taille de la fenêtre ?
Question 25 : A quels instants a-t-on de la congestion durant le slow-start ?

Il est également possible d'exécuter des scripts sur le fichier out.tr. Le programme perl throuput.pl calcule le débit moyen de TCP en fonction du temps, au niveau d'un nœud précis, et écrit le résultat dans le fichier thp, si on lance la commande :

```
perl throughput.pl out.tr 4 1 > thp
```

Question 26 : A quoi correspondent les arguments de la commande précédente (out.tr, 4, 1) ?
Question 27 : Dessinez l'allure de la courbe obtenue en visualisant le ficher thp avec gnuplot.

Question 28 : Cette courbe vous paraît-elle cohérente avec la précédente ?

3. Routage unicast

Considérons le script ex5.tcl.

Question 28 : Quelle est la topologie du réseau (nœuds, liens) ? Vous pouvez faire un schéma.
Question 29 : Combien y a-t-il de sources de trafic ? Décrivez chacune des applications (source, destination, type, débit, instant de début et de fin)
Question 30 : Quel est le type de routage utilisé ?
Question 31 : Quel est le lien qui tombe en panne ? Pendant quelle durée ce lien est-il défaillant ?
Question 32 : Quelle est la conséquence de la panne du lien ?

Considérons à présent le script ex6.tcl.

Question 33 : Quelle est la différence entre les scripts ex5.tcl et ex6.tcl ? Comment cette différence se traduit-elle en cas de panne ?

4. Routage multicast

Question 34 : Décrivez la simulation obtenue avec le script ex7.tcl, qui utilise le protocole PIM-DM.
Question 35: Décrivez la simulation obtenue avec le script ex8.tcl, qui utilise un protocole de routage multicast avec un point de rendez-vous central.
Question 35 : Quelles sont les différences entre ces deux protocoles ?

TME 2 ING - NS : Network Simulator

Copiez le répertoire /users/Enseignants/blegrand/TP2ing sur votre compte.

1. Comparaison de DropTail et RED

La gestion de buffer RED (Random Early Discard) a été introduite en 1993 par Floyd et Jacobson. L'idée de base est que l'on ne doit pas attendre que le buffer soit plein pour détecter la congestion (jeter des paquets) ; il faut détecter la congestion avant que le buffer déborde.

RED contrôle la taille moyenne de la file d'attente (avg) et vérifie qu'elle se trouve entre un seuil minimum (min_{th}) et un seuil maximum (max_{th}). Si c'est le cas, un paquet qui arrive est jeté avec la probabilité p=p(avg), qui est une fonction croissante de la taille moyenne de la file d'attente. Tous les paquets qui arrivent alors que avg>max_{th} sont jetés.

La taille moyenne de la file d'attente est calculée comme suit : le paramètre avg est initialisé à 0. Ensuite, à chaque arrivée de paquet, avg prend la valeur :

$$(1 - w_q)avg + w_q q$$

où q est la taille réelle de la file d'attente et w_q est une constante de petite valeur.

Dans le script drptail.tcl, la gestion du buffer est un simple mécanisme de drop tail. Pour lancer la simulation, tapez la commande :

```
ns drptail.tcl
```

Question 1 : Quelle est la topologie du réseau simulé ? Vous pouvez faire un schéma.
Question 2 : Quelles sont les caractéristiques des liens de communication ? Quel est le goulot d'étranglement ?
Question 3 : Quel agent est attaché à chaque nœud ? Combien y a-t-il de connexions ?
Question 4 : De quel type d'application s'agit-il ?
Question 5 : Quelle est la taille des paquets TCP ?
Question 6 : Quelle est la taille maximale des fenêtres ?
Question 7 : Combien de temps dure la simulation ?
Question 8 : A quels instants les applications démarrent-elles ?
Question 9 : Quels sont les fichiers générés par cette simulation ?

Observez avec gnuplot l'évolution de la taille de la file d'attente (fichier queue.tr). Pour cela, il faut choisir la première colonne de ce fichier comme axe des abscisses et la colonne 5 comme axe des ordonnées. Cela se traduit de la manière suivante :
gnuplot
plot 'queue.tr' using 1:5 w lines 1

Observez également (sur le même graphe) les courbes représentant les variations des fenêtres pour chaque connexion. Il vous faut pour cela utiliser le fichier win en prenant la première colonne comme axe des asbcisses et les diverses colonnes suivantes comme axe des ordonnées.
Par exemple :
gnuplot
plot 'win1' using 1:2 w lines 1, 'win1' using 1:3 w lines 2, etc.

Question 10 : Représentez succinctement et commentez l'évolution de la taille de la file d'attente Constate-t-on des oscillations ? Le buffer déborde-t-il parfois ?
Question 11 : Quelle est la taille moyenne de la file d'attente ? Déduisez-en le délai moyen des connexions.
Question 12 : Que constatez-vous en ce qui concerne l'évolution relative de la taille des fenêtres ? D'après vous, sont-elles synchronisées ? Est-ce un avantage ou un inconvénient ?

Considérons à présent le script red.tcl, pour lequel on garde les mêmes paramètres, mais où l'on utilise une gestion de buffer RED, avec une configuration de paramètres automatique.

Question 13 : Quelle est la valeur des paramètres RED (seuils) ?
Question 14 : Observe-t-on plus ou moins d'oscillations de la taille de la file d'attente que dans le cas du drop tail ? (Affichez pour cela, sur le même graphe, les courbes ave.tr et cur.tr).
Question 15 : Quelle est la taille moyenne de la file d'attente ? Déduisez-en le délai moyen des connexions.
Question 16 : La file d'attente déborde-t-elle parfois ?
Question 17 : Que pouvez-vous dire de l'évolution relative de la taille des fenêtres pour les diverses connexions ?

Faites varier les paramètres thresh_ (60), maxthresh_ (80) et q_weight_ (0.002).

Question 18 : Quelle est la taille moyenne de la file d'attente ?
Question 19 : Commentez l'évolution de la taille des fenêtres.

On s'intéresse maintenant au script shortRed.tcl, pour lequel on a des sources qui génèrent de nombreuses connexions TCP (à la différence des cas précédents, où l'on avait des connexions de plus longue durée).

Question 20 : Quelle est la topologie du réseau simulé ? Décrivez les caractéristiques du lien bottleneck.
Question 21 : Combien y a-t-il de sources TCP ?
Question 22 : Décrivez et commentez l'allure de l'évolution de la taille de la file d'attente (affichez pour cela ave.tr et cur.tr sur le même graphe). Comparez ces courbes avec les courbes obtenues dans le cas précédent (script red.tcl). A-t-on un trafic plus ou moins sporadique ?
Question 23 : Affichez la courbe représentant le nombre de connexions actives en fonction du temps (Conn.tr).

2. Réseaux ad hoc

Il existe plusieurs protocoles de routage dans les réseaux ad hoc implémentés dans NS : DSDV (Destination Sequenced Distance Vector), DSR (Dynamic Source Routing), TORA (Temporally Ordered Routing Algorithm) et AODV (Ad-hoc On Demand Distance Vector).

Nous nous intéressons au protocole DSDV. Considérons le script wrls-dsdv.tcl.

Question 24 : Quelle est la topologie du réseau ?
Question 25 : Décrivez le scénario de simulation. A quel instant le transfert ftp démarre-t-il ?
Question 26 : Observez la courbe win.tr. Commentez-la.

Faites démarrer le tranfert ftp à l'instant 12.
Question 27 : Comparez la nouvelle courbe win.tr avec la précédente. Comment expliquez-vous les différences ?