TP MIF27 N°2
Fichier *trace*, Files d'attente, Modèles d'erreurs

M. Haddad

14 avril 2011

1 Analyse du fichier trace (réseau filaire)

Dans cette section, nous allons étudier les événements se produisant dans un réseau filaire à partir du fichier trace (.tr) résultant de la simulation.

– Construire un réseau à 6 nœuds ayant pour topologie celle illustrée par la figure ci-dessous. Tous les liens doivent offrir exactement 1 mb/s de débit et 50 ms de latence.

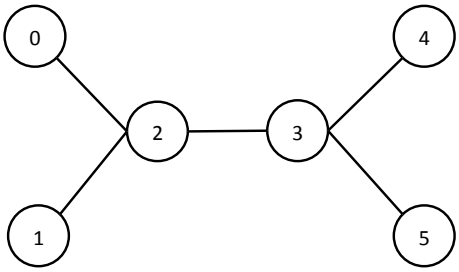

FIGURE 1 – Topologie du réseau.

– Établir un flux constant (CBR) entre 0 et 4 et un autre entre 1 et 5. Donner la couleur bleue au premier et la couleur rouge au second.

– Régler les paramètres des flux de manière à ne pas avoir de congestion (pas de perte de paquets).

Assurez-vous d'avoir un fichier trace (fichier log) en insérant les lignes suivantes après la création d'une instance du simulateur :

```
# Le fichier trace
set f [open out.tr w]
$ns trace-all $f
```

Passons maintenant à l'analyse du fichier trace. Le format d'un ligne du fichier trace, pour un réseau filaire, est la suivante :

<%c %g %d %d %s %d %s %d %d.%d %d.%d %d %d>

```
char : Event
double : Time
int : (Link-layer) Source Node
int : (Link-layer) Destination Node
string : Packet Name
int : Packet Size
string : Flags
int : Flow ID
int : (Network-layer) Source Address
int : Source Port
int : (Network-layer) Destination Address
int : Destination Port
```

```
int : Sequence Number
int : Unique Packet ID
```

Où Event peut être :

```
r: Receive
d: Drop
e: Error
+: Enqueue
-: Dequeue
```

Considérons l'exemple suivant :

```
r 1.84471 2 1 cbr 210 ------- 1 3.0 1.0 195 600
r 1.84566 2 0 ack 40 ------- 2 3.2 0.1 82 602
+ 1.84566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
- 1.84566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
r 1.84609 0 2 cbr 210 ------- 0 0.0 3.1 225 610
+ 1.84609 2 3 cbr 210 ------- 0 0.0 3.1 225 610
d 1.84609 2 3 cbr 210 ------- 0 0.0 3.1 225 610
```

La première ligne signifie la réception du paquet n° 600 de type CBR par le nœud 1 et dont l'émetteur est le nœud 2. Les deux dernières lignes montrent la mise en file du paquet n° 610 puis sa suppression de cette dernière (perte du paquet). Les caractères '-' (la chaîne flag) ne nous intéressent pas dans ce TP.

Les transmission doivent durer exactement 1 seconde.

- Augmenter les débits d'émission de manière à avoir des pertes de paquets.
- Dans un langage au choix (C/C++, Java, Awk ...), extraire le nombre de paquets perdus pour chaque flux (on peut distinguer les flux par leurs émetteurs ou destinataires mais au niveau réseau et non pas lien : network-layer et non pas link-layer).
- Comparer les nombres de paquets reçus.
- Remplacer le second flux CBR par un flux FTP. Que remarquez-vous ?
- Comparer les nombres de paquets perdus puis les nombres de paquets reçus.

2 Files d'attente

Dans cette section, nous allons voir comment établir une file d'attente selon le modèle M/M/1/K sur une lien de communication filaire. Dans l'exemple suivant, nous allons créer deus nœuds et lier par un lien unidirectionnel sans latence afin de pouvoir gérer manuellement les temps inter-arrivées.

```
set ns [new Simulator]

# Le fichier trace
set tf [open out.tr w]
$ns trace-all $tf

# paramètre des lois exponentielles

set lambda   30.0
set mu       33.0

# K = taille de la file
set qsize     5

# Durée de simulation
set duration  200


# création du réseau
```

```
set n1 [$ns node]
set n2 [$ns node]
set link [$ns simplex-link $n1 $n2 100kb 0ms DropTail]
$ns queue-limit $n1 $n2 $qsize

# Génération aléatoire des intervalles de temps et des tailles de paquets.
set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr 1/$lambda]

# Puisque les tailles des paquets seront arrondies à des valeurs entières,
# nous devons avoir une tailles max importante afin de diminuer l'erreur d'arrondi.

set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ [expr 100000.0/(8*$mu)]

# création et attachement des agents de transport.

set src [new Agent/UDP]
$src set packetSize_ 100000
$ns attach-agent $n1 $src

set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $src $sink

# Procédure d'envoi des paquets

proc sendpacket {} {
    global ns src InterArrivalTime pktSize
    set time [$ns now]
    $ns at [expr $time + [$InterArrivalTime value]] "sendpacket"
    # la fonction round permet l'arrondi en valeur entière
    set bytes [expr round ([$pktSize value])]
    $src send $bytes
}

# queue monitoring : moniteur de la file d'attente
# le fichier qm.out contiendra la trace correspondante.
# le format de cette trace est :
# Time From To qSize qSize Departed Arrived Dropped Departed Arrived Dropped

set qmon [$ns monitor-queue $n1 $n2 [open qm.out w] 0.1]
$link queue-sample-timeout

proc finish {} {
    global ns tf
    $ns flush-trace
    close $tf
    exit 0
}


$ns at 0.0001 "sendpacket"
$ns at $duration "finish"

$ns run
```

A faire :

1. Décrire le code source ci-dessus : le résumer.
2. Expliquer (en détail) la fonction *sendpacket*.

3 Modèles d'erreurs

Dans cette section, nous verrons comment introduire des erreurs de transmission sur un lien de communication.

```
set ns [new Simulator]

set filetrace [open out.tr w]

proc finish {} {
    global ns flowmon fcl filetrace
    close $filetrace
    exit 0
}

proc record {} {
        global ns flowmon bitsOutOld filetrace
        set fid 1
        #Set the time after which the procedure should be called again
        set time 0.5
        #How many bytes have been received by the traffic sinks?
        set flow [[$flowmon classifier] lookup auto 0 0 $fid]
        set bitsOut [$flow set bdepartures_]
        #Get the current time
        set now [$ns now]
        #Calculate the bandwidth (in MBit/s) and write it to the files
        puts $filetrace "$now [expr ($bitsOut-$bitsOutOld)/$time*8/1000000]"

        #puts $filetrace "$now $drop"
        set bitsOutOld $bitsOut
        #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}

##Création des noeuds
set n0 [$ns node]
set n1 [$ns node]

##Init du lien duplex
$ns duplex-link $n0 $n1 50.0Mb 200.0ms DropTail
$ns queue-limit $n0 $n1 20

##Init de la connexion TCP
set tcp0 [new Agent/TCP]

#Parametre du fluc TCP
$tcp0 set packetSize_ 1500
$tcp0 set fid_ 1
$ns attach-agent $n0 $tcp0

set null0 [new Agent/TCPSink]
$ns attach-agent $n1 $null0

$ns connect $tcp0 $null0

##flow monitor
set flowmon [$ns makeflowmon Fid]
#Attache le monitor au lien
set lien [$ns link $n0 $n1]
$ns attach-fmon $lien $flowmon
```

```
##Loss Model
set lossModel [new ErrorModel]
$lossModel unit packet
$lossModel set rate_ 0.1
$lossModel ranvar [new RandomVariable/Uniform]
$lossModel drop-target [new Agent/Null]

$ns lossmodel $lossModel $n0 $n1

##Scheduler
set bitsOutOld 0
puts "On envoie le fichier de 3000Ko"
$ns at 10.0 "$tcp0 send 3000000"
$ns at 11.0 "record"
$ns at 200.0 "finish"

$ns run
```

A faire :

1. Comment est fait le calcul de la bande passante dans la fonction *record*.
2. Dessiner la courbe de l'évolution du débit durant le temps.
3. Comparer les débits obtenus avec des taux d'erreur de 0.01, 0.1 et 0.5.