La simulation et le simulateur réseaux ns-2

Illustration du cours avec le simulateur réseau ns-2

Lecture given by Emmanuel Lochin

ISAE-SUPAERO

Plan

Méthode d'évaluation des performances d'un système

- Experimentation réelle
  - Problème du coût et de la reproduction des mesures
  - Nécessité de prise en compte des instabilités du système physique (effets de bord, paramètres cachés ou inconnus)
- Simulation à événements discrets
  - Modèle simplifié du système réel
  - Pas d'effets de bord et de mesures "cachées"
  - Métriques similaires à l'expérimentation réelle mais idéalisée
  - Temps simulé (rapide)
- Méthode analytique
  - Certaines méthodes sont proches de la simulation (ex. programmation de modèles sous Matlab)
  - Plus rapide que la simulation
  - Problème : requiert parfois des hypothèses fortes (loi exponentielles pour un modèle Markovien)
- Emulation
  - Compromis entre la simulation et l'expérimentation réelle
  - Permet d'ajouter des éléments réels à un module simulé
  - Temps réel (long)

Simulation à événement discret

- Temps réel et temps simulé
- Définition du scénario (*event*)
- Graines aléatoires (*seeds*)
- Comme en réel, besoin d'un échantillon statistique cohérent pour obtenir un cas moyen cohérent
- Simulation déterministe ou stochastique
  - Événements aléatoires durant la simulation
  - Utilisation d'un canal à effacement basé sur un modèle probabiliste (e.g. Gilbert Elliot, Bernouilli)
- Simulation terminale ou interrompue
  - Ex : simulation de la durée de vie d'un système, interruption volontaire
- Les conditions initiales déterminent le résultat de la simulation
  - Dans le cas de seeds : plusieurs résultats pour des conditions identiques
  - A-t-on besoin de chercher tous les cas possibles ?
  - Simulation stationnaire ou asymptotiquement stationnaire (dépend du scénario, de la trace, ...)

Mesures obtenues

- Toutes celles relatives au système et à expérimentation
  - Max, min, moyenne, ...
  - De tout élément du système
  - Toutes métriques mesurées
    - ⋆ délai, débit, pertes, retransmissions, taux d'occupation de file, corrélation entre toutes les mesures, possibilité de disséquer un comportement, identifier les cas bloquants ...
- Comportement moyen
  - Nécessité de définir l'ensemble des scénarios à étudier
  - De vérifier l'exhaustivité des scénarios
  - De s'assurer via des outils statistiques de la pertinence de l'échantillon statistique

Le simulateur ns-2

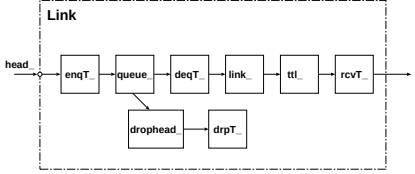
- ns-2 : Network Simulator.
- Simulateur sous licence GPL développé à Lawrence Berkeley National Laboratory (LBNL) `http://www.isi.edu/nsnam/ns/`
- Conçu principalement pour le monde de l'Internet et plus particulièrement pour le protocole TCP
- Beaucoup d'extensions sont disponibles de part sa bonne organisation hiérarchique au niveau des :
  - Nouveaux protocoles de transport et de routage
  - Nouveaux médias de transmission, satellite, WiMAX, 802.11, ...
  - Nouvelles architectures de qualité du service dans l'Internet (MPLS, DiffServ, IntServ)
- Utilisé par CNES, TAS, Thales Comm., Eurecom, INRIA, ...
- Maintenant remplacé par ns-3

Structure de ns-2

- Simulateur orienté objet :
  - Une simulation n'est autre que la circulation d'un paquet, qui est lui même un objet, entre différents objets représentant les différents éléments du réseau à simuler
- Le code du simulateur est écrit en C++. Les principales classes sont :
  - **Application** : la classe mère de toutes les applications (e.g., ftp,telnet)
  - **Agent** : La classe mère de tous les protocoles qui tournent au niveau 3 ou 4 (e.g., TCP, UDP, SRM, RIP, OSPF).
  - **Node** : Représente l'ensemble des nœuds du réseau. Chaque nœud contenant ...
  - ... un **Classifier** pour envoyer le paquet arrivant d'un agent ou d'une interface vers la bonne sortie
  - **Queue** : La classe mère de tous les buffers (e.g., Drop Tail, RED).

Structure de ns-2

- **LinkDelay** : Simule un délai de propagation. Avec la classe **Queue**, elle permet de définir la classe des liens du réseau
- **Packet** : La classe des paquets. Cette classe définit un pointeur vers l'en-tête du paquet (Header) et un pointeur vers le champ de données (Payload)
- **TimerHandler** : La classe mère de tous les timers utilisés par les différents éléments du réseau



Les objets dans ns-2

- Chaque objet du simulateur (à l'exception de l'application de destination) dispose d'un pointeur vers l'objet suivant auquel le paquet doit être fourni après être traité. Pour passer le paquet à un objet, il suffit d'appeler sa fonction `recv` avec un pointeur vers le paquet comme paramètre
- La simulation d'un réseau en ns-2 nécessite donc :
  - La définition des objets (si les classes correspondantes existent déjà)
  - La connexion des objets créés entre eux
  - Le lancement des applications sources

Le Scheduler

- Le **Scheduler** qui permet l'exécution des événements selon un scénario définit par l'expérimentateur
- Au début de la simulation, l'expérimentateur précise un certain nombre d'événements
- Pendant la simulation, les objets définissent eux-même d'autres événements
- Tous les événements envoyés au **Scheduler** sont stockés dans une même liste
- La simulation termine lorsque l'événement `exit` précisé par l'utilisateur est rencontré
- On dit que ns-2 est un simulateur basé sur les événements

L'interface OTcl

- OTcl est un langage de programmation orienté objet offrant les mêmes fonctionnalités que C++
- L'ensemble des actions spécifiques à un scénario de simulation donné est passé au simulateur sous forme d'un fichier écrit en OTcl
- Un interpréteur OTcl est ajouté au simulateur pour traduire les instructions OTcl passées par l'interface au simulateur en des instructions C++
- Lorsque le simulateur est appelé, une hiérarchie de classes similaires à celle au niveau C++ est créée au niveau OTcl. L'utilisateur programme sur l'interface comme si le simulateur avait été écrit en OTcl. En particulier, il peut instancier des classes existantes ou bien définir de nouvelles classes

La programmation en ns-2

- Aucune connaissance en C++ nécessaires pour utiliser le simulateur. Néanmoins il faut connaître :
  - OTcl (très rapide à prendre en main)
  - Un peu l'architecture au niveau C++ et la hiérarchie des classes
  - Les méthodes et les attributs de chaque objet OTcl
  - OTcl est seulement une image de ce qui existe au niveau C++
  - La fonction `main()` du simulateur est donc toute simple et ne fait qu'appeler l'interpréteur OTcl

Les instructions de base en OTcl

- Assigner une valeur à une variable : `set a 10`
- Lire le contenu d'une variable : `set b $a`
- Ouvrir un fichier : `set f [open File w]`
- Imprimer le contenu d'une variable dans un fichier : `puts $f "$b"`
- Opérations arithmétiques : `set a [expr $b (+-*/) $c]`

Entrainez-vous !

`http://www.compileonline.com/execute_tcl_online.php`

Les instructions de base en OTcl

- Structures de contrôle :
  - condition :

```
if {$a==$b} {
... }
```

  - boucle `for` :

```
for {set i 1} {$i<=10} {incr i} {
... }
```

  - pour faire une incrémentation avec un pas :

```
for {set i 1} {$i<=10} {set i [expr $i+1]} {
... }
```

  - un exemple pour créer 25 émetteurs et 25 récepteurs :

```
for {set i 1} {$i <= 25} {incr i} {
        set s_($i) [$ns node]
        set r_($i) [$ns node]
... }
```


Les instructions de base en OTcl

- Définir une fonction :

```
proc fct {par1 par2 (...)} {
   global a b
   (...)
   return [expr $a + $par1]
}
set c [fct $par1 $par2 ...]
```

- Instancier une classe : `set obj [new Class1/Class2/Class3]`
- Appeler une méthode d'un objet sans retour : `$obj method par1 par2 (...)`
- Et avec retour : `set a [$obj method par1 par2 (...)]`
- Assigner une valeur à un attribut d'un objet : `$obj set attrib 10`
- Lire le contenu de l'attribut d'un objet : `set a [$obj set attrib]`

Création d'un scénario de simulation

- On instancie la classe OTcl Simulateur : `set ns [new Simulator]`
- Création des nœuds :

```
for {set i 1} {$i<=NbNode} {incr i}
{set n_($i) [$ns node]}
```

- Les liens entre les nœuds :

```
$ns duplex-link $n(1) $n(2) Bandwidth Delay \
                            TypeOfBuffer
```

- On peut fixer la taille des buffers à l'entrée des liens :

```
$ns queue-limit $n(1) $n(2) Size
```

Création d'un scénario de simulation

- Création des agents transports et attachement aux nœuds. Pour un agent TCP :

```
set transp1 [new Agent/TCP]
$ns attach-agent $n(1) $transp1
set transp2 [new Agent/TCPSink]
$ns attach-agent $n(2) $transp2
$ns connect $transp1 $transp2
```

- Ensuite, création des applications et on les attache aux agents transports. Pour une application FTP :

```
set app [new Application/FTP]
$app attach-agent $transp1
```


Création d'un scénario de simulation

- Il ne reste plus que lancer le Scheduler :

```
$ns at 10 "$app start"
$ns at 100 "exit 0"
```

- On peut aussi changer les conditions à la volée :

```
$ns at 10 "$app start"
$ns at 25.0 "$ns bandwidth $n1 $n2 1000Kb duplex"
$ns at 75.0 "$ns bandwidth $n1 $n2 500Kb duplex"
$ns at 100 "exit 0"
```

- avec un PLR :

```
$ns at 25.0 "$ns lossmodel $plr $n1 $n2"
$ns at 75.0 "$plr set rate_ 0"
```


Paramétrage de quelques objets

- TCP :

```
set tcp [newAgent/TCP(TCP/Reno)(TCP/Newreno)(TCP/Sack1)]
set sink [newAgent/TCPSink(TCPSink/DelAck)(TCPSink/Sack1)]
$tcp set window_ X (packets)
$tcp set ssthresh_ X (packets)
$tcp set packetSize_ X (bytes)
```

- RED :

```
$ns duplex-link $n(1) $n(2) Bandwidth Delay RED
set q [[$ns link $n(1) $n(2)] queue]
$q set thresh_ Min
$q set maxthresh_ Max
$q set q weight_ AveragingWeight
$q set linterm_ InverseOfMaxProb
```


Tester de nouveaux contrôle de congestion

Voir : http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux-2.29-linux-2.6.16/tutorial/index.html

```
#setup sender side
set tcp [new Agent/TCP/Linux]
$tcp set timestamps_ true
$ns attach-agent $n0 $tcp
#set up receiver side
set sink [new Agent/TCPSink/Sack1]
$sink set ts_echo_rfc1323_ true
$ns attach-agent $n1 $sink
#logical connection
$ns connect $tcp $sink
#setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ns at 0 "$tcp select_ca {reno, cubic, westwood, ...}"
```


Paramétrage de quelques objets

- Une source CBR :

```
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ X (bytes)
$cbr set rate_ XKB (Kb) (MB) (Mb)
```

- On peut également émettre un nombre donné de paquets :

```
set size 20000
$ns at $startt "$ftp($k) send $size"
```

- Une source Poisson :

```
set poisson [new Application/Traffic/Poisson]
$poisson set interval_ X (secondes)
$poisson set size_ (bytes)
```

A noter !

Tout est dans : `ns-X.XX/tcl/lib/ns-default.tcl`
Attention pour la source Poisson il faut appliquer un patch, ce n'est pas en standard dans ns-2 :
`http://www.matlab.nitech.ac.jp/~khpoo/research/index-poisson.htm`

Les traces ns-2

- Si l'utilisateur désire tracer des résultats relatifs à chaque paquet (e.g., le temps d'arrivée d'un paquet à un buffer), les objets `Trace` du simulateur sont nécessaires. Ces objets s'associent aux objets du simulateur et écrivent dans un fichier de sortie le temps d'arrivée et de départ de chaque paquet
- La classe `Simulator` dispose d'une méthode `trace-all` qui associe des objets `Trace` à tous les buffers et les liens d'un réseau. Une ligne est écrite dans un fichier de sortie chaque fois qu'un paquet arrive à un buffer, quitte un buffer, est jeté à l'entrée d'un buffer ou bien atteint la sortie d'un lien

```
set f [open File w]
$ns trace-all $f
```


Les traces ns-2

- Par exemple : traçage de la fenêtre d'une connexion TCP en fonction du temps :

```
proc windowtrace {} {
  global tcp0 ftrace
  set ns [Simulator instance]
  set time 0.01
  set now [$ns now]
  puts $ftrace "$now [$tcp0 set cwnd_]"
  $ns at [expr $now+$time] "windowtrace"
}
(...)
$ns at 0.0 "windowtrace"
```


Les traces ns-2

- Méthode alternative au traçage de la fenêtre d'une connexion TCP en fonction du temps :

```
set f [open cwnd.tr w]

# monitoring cwnd
set traceur [new Trace/Var]
$traceur attach $f
$tcp1 trace cwnd_ $traceur
```

- Le format de sortie est du type : `f t0.333504 a_o128 ncwnd_ v5`

Les traces ns-2

- On peut aussi demander à la classe Simulator d'associer un objet Trace seulement au buffer situé entre les deux nœuds n1 et n2 :

  ```
  $ns trace-queue $n1 $n2 $f
  ```

- Les moniteurs des buffers permettent d'avoir à n'importe quel moment dans la simulation des informations sur les buffers à l'entrée des liens du réseau (e.g., nombre des paquets reçus, nombre des paquets rejetés, nombre des paquets renvoyés) :

  ```
  set monitor [$ns monitor-queue $n1 $n2 stdout]
  puts [$monitor set parrivals_ (pdepartures_)(pdrops_)(barrivals_)]
  ```

- Il y aussi des objets pour surveiller les paquets d'un flot donné

Visualisation avec nam

- nam permet de voir votre topologie et les paquets circulant sur un lien
- permet notamment de "voir" concrètement un slowstart TCP
  - on peut aussi voir un slowstart avec les traces tcpdump :)
- Mise en place de nam :

  ```
  # mise en place des traces nam
  set nf [open out.nam w]
  $ns namtrace-all $nf
  ```

- Execution de nam dans le script Tcl :

  ```
  close $nf
  exec nam out.nam &
  ```


Un exemple du début à la fin I

```
set ns [new Simulator]

# Color
$ns color 1 Blue
$ns color 2 Red

# mise en place des traces
set f [open out.tr w]
$ns trace-all $f

# mise en place des traces nam
set nf [open out.nam w]
$ns namtrace-all $nf

# La topologie:
#
# n0
#    \
# 5Mb \ 2ms
#      \
#      n2 ----------- n3
#      /      1Mb
```


Un exemple du début à la fin II

```
# 5Mb / 2ms   10ms
#    /
# n1

# Creation des noeuds
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Creation des liens
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

# Un flot CBR/UDP
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$udp0 set class_ 1
```


Un exemple du début à la fin III

```
# Un trafic CBR
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 10kb

# On attache un FTP sur TCP/Tahoe de $n1 vers $n3
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set class_ 2

# un generateur de traffic FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp

# Start Simu
$ns at 1.0 "$ftp start"
$ns at 2.0 "$cbr0 start"
$ns at 60.0 "$cbr0 stop"
$ns at 60.0 "finish"
```


Un exemple du début à la fin IV

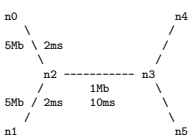
```
proc finish {} {
global ns f nf
$ns flush-trace
close $f
close $nf
puts "running nam..."
exec nam out.nam &
exit 0
}

# Lancement de la simu
$ns run
```


Et maintenant ... simulons !

- Téléchargez l'archive sous LMS contenant ce script de simulation et executez le en lançant launch.sh
- Modifiez la valeur du débit UDP par 500kb et faite démarrer ce flot entre t=[20,30] secondes. Qu'observez-vous et pourquoi ?

Topologie butterfly avec deux flots TCP

- Générer deux flots TCP et calculer le débit instantané et cumulé pour chacun avec la topologie ci-dessous :



Fin

Document réalisé avec LaTeX le 16 octobre 2019
Style de document beamer
Dessins réalisés avec xfig
Emmanuel Lochin http://personnel.isae.fr/emmanuel-lochin/