PROTOCOALE DE COMUNICAȚIE : Laborator 5

Protocoale de Dirijare cu NS2

Responsabili: Cătălin LEORDEANU, Dorinel FILIP

Cuprins

Instalarea NS-2

NS-2 este unul dintre cele mai folosite simulatoare de rețea. Pe un sistem Debian-based, acesta poate fi instalat rulând (ca root) următoarele comenzi:

```
apt-get install ns2
apt-get install nam
```

Descrierea modului de lucru cu NS2

Folosirea NS necesită descrierea experiementelor în limbajul TCL. Atunci când este rulat fără niciun argument, ns așteaptă furnizarea experimentului la stdin, însă, de cele mai multe ori, se preferă descrierea experimentului într-un fișier separat (de ex. test.tcl) și rularea ns test.tcl.

În fișierul de descriere a experimentului se poate specifica un fișier în care acesta să salveze rezultatele simulării. Pentru vizualizarea (grafică) a lor putem folosi utilitarul Network Animator (nam) care poate fi rulat folosind: nam fișier_ieșire.

Descrierea fișierului routing.tcl

În cadrul acestui laborator, toate experimentele vor fi construite pornind de la fișierul routing.tcl disponibil în resursele de laborator. În continuare vom descrie, pe scurt, ce funcționalitate implementează fiecare parte a scriptului.

Crearea unei simulari / specificarea unui fișier de ieșire

În limbajul TCL, sintaxa pentru definirea unei noi variabile este set variabila valoare, iar referirea la valoarea unei variabile deja definite se poate face folosind $variabla.

Inițializarea unui experiement în NS-2 începe prin instanțierea unui obiect de tipul Simulator și specificarea fișierului de ieșire:

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
```

Prima linie crează simulatorul. A doua deschide pentru citire fișierul out.nam, iar ultima instruiește ns să logheze rezultatele în fișierul de mai sus.

Finalizarea simulării

Pentru a opri simularea (care altfel ar fi infinită), se definește o funcție care golește bufferele de scriere ale ns (apelul flush-trace), închide fișierul și apelează Network Animator (nam) pentru fișierul de output generat, înainte de a încheia execuția scriptului:

```
proc finish {} {
        global ns nf
        $ns flush-trace
    #Close the trace file
        close $nf
    #Execute nam on the trace file
        exec nam out.nam &
        exit 0
}
```

Aceasta este apelată atunci când timpul simulării ajunge la 5s:

```
$ns at 5.0 "finish"
```

Pentru a realiza simulări cu durată mai mare se poate modifica această valoare.

Crearea de noduri

Pentru a face lucruri utile în ns trebuie să definim graful rețelei. Un nod se crează apelând funcția $ns node. Mai jos creăm două noduri numite n0 și n1.

```
set n0 [$ns node]
set n1 [$ns node]
```

Întrucât acestea sunt variabile, numele lor trebuie să fie unice în cadrul simulării.

Adăugarea legăturilor

Pentru a conecta două noduri trebuie să definim o legătură de date. Parametrii ce caracterizează o astfel de legătură sunt nodurile capete, lățimea de bandă, latența și politica de punere în coada de așteptare a pachetelor.

```
$ns duplex-link $n0 $n1 100Kb 100ms DropTail
```

- duplex-link înseamnă că legătura permite trimiterea de pachete în ambele sensuri
- DropTail înseamnă că că mesajele se pierd atunci când coada este deja plină

Definirea costurilor

Odată definită întreaga topologie, se poate configura un cost pe legătura de date, pentru a fi folosit de algoritmul de dirijare. Costurile pentru o legătura duplex se configurează pe o singura direcție, astfel:

```
$ns cost $n0 $n1 3
```

Activarea rutării dinamice

În mod standard, ns calculează static rutele între noduri, folosind algoritmul Dijsktra. Astfel, el nu va răspunde în niciun fel posibilelor modificări/defecte apărute după momentul de început al simulării. Dacă dorim ca rutarea să se facă dinamic, trebuie să configurăm explicit acest lucru. Modurile de lucru pot fi DV (distance vector) sau LS (link state):

```
$ns rtproto DV
```

Generarea de trafic

Pentru a putea genera trafic în topologie, trebuie să creăm atât un transmițător, cât și un receptor de nivel transport (TCP sau UDP). Pentru a fi vizibile în simulare și a nu genera erori / warning-uri, acestea vor fi atașate unor noduri. În exemplul de mai jos, se crează o sursă UDP ce trimite de la nodul n0 la nodul n2. Sursa va fi configurată să înceapă transmisia dupa 0.5 secunde de la începutul simulării și să se oprească după 4 secunde.

```
#creaza un transmitator care foloseste protocolul UDP
set udp0 [new Agent/UDP]
#ataseaza transmitatorul nodului n0
$ns attach-agent $n0 $udp0

#creaza o sursa de pachete de rata constanta (constant bit rate CBR) care emite
    un pachet de 500 octeti la fiecare 100ms.
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.1
$cbr0 attach-agent $udp0

#creaza un receptor care pur si simplu "pierde" pachete
set null0 [new Agent/Null]
#ataseaza-l nodului n2
$ns attach-agent $n2 $null0
```

```
16
17 #conecteaza transmitatorul si receptorul
18 $ns connect $udp0 $null0
19
20 #la secunda 0.5 incepem transferul de date
21 $ns at 0.5 "$cbr0 start"
22 #la secunda 4.5 oprim transferul de date
23 $ns at 4.5 "$cbr0 stop"
```

Simularea de defecte

Pentru a putea observa modul de funcționare al rutării dinamice, putem simula apariția de defecte ale legăturilor de date. În exemplu de mai jos legătura dintre n1 și n2 este oprită la secunda 1 și repornită în secunda 3.

```
1 $ns rtmodel-at 1.0 down $n0 $n2
2 $ns rtmodel-at 3.0 up $n0 $n2
```

Pornirea simulării

Ultima linie din fișier rulează simulatorul:

```
1 $ns run
```

Exerciții

1. În fișierul routing.tcl este definită o topologie formată din 3 noduri (n0, n1 și n2) dispuse în triunghi.
   (a) modificați costurile astfel încât protocolul de rutare să prefere ruta n0-n1-n2 între n0 și n2;
   (b) simulați un defect pe legătura n1-n2 și observați ce se întâmplă.
2. Realizați în NS-2 topologia din fișierul fig.png, configurați un transmițător de la nodul n0 la n3, apoi:
   (a) Simulați minim 2 defecte și observați ce se întâmplă în funcție de tipul de rutare dinamică folosit. Câte pachete se pierd?
   (b) Setați costurile pentru topologia dată pentru a obține viteză maximă
   (c) Setați costurile pentru a obține latență minimă

   Observație: Pentru rezolvarea ultimelor 2 cerințe se vor folosi fișiere .tcl diferite.