Università di Verona
Dipartimento di Informatica

Architetture Multimediali

Introduzione al simulatore NS-2

Davide Quaglia
Giuseppe Di Guglielmo

Scopo di questa esercitazione è imparare l'utilizzo del simulatore di rete ***NS-2*** e di alcuni suoi tool di supporto al fine di costruire scenari di rete, simularli ed ottenere informazioni sul loro comportamento.

Al fine di una completa comprensione dei contenuti è prerequisito importante la conoscenza dei fondamenti di rete soprattutto nel contesto TCP/IP.

Per approfondimenti sui comandi e sulle opzioni si faccia riferimento al manuale di NS-2 scaricabile presso http://www.isi.edu/nsnam/ns/ns-documentation.html.

1. Scaricamento e installazione

NS-2 e relativi tool si possono scaricare liberamente dalla pagina web http://www.isi.edu/nsnam/ns/ns-build.html. è consigliabile scaricare la versione "all-in-one" che contiene il simulatore e tutte le librerie accessorie in modo da non dover verificare preventivamente la loro presenza nel sistema. Quando l'installazione è completata occorre inserire nel PATH la directory "ns-allinone/bin". Occorre anche impostare le variabili d'ambiente LD_LIBRARY_PATH e TCL_LIBRARY secondo quanto scritto a video dal programma di installazione prima di terminare.

2. Avvio di NS-2 e di NAM

Il simulatore si avvia con il comando "ns <tclscript>" dove "tclscript" è la descrizione in linguaggio TCL dello scenario di rete e dei relativi eventi preimpostati.
NAM (Network AniMator) è un tool accessorio che prende in ingresso un file di statistiche prodotto da NS-2 durante la simulazione e visualizza graficamente il comportamento della rete nel tempo. Per generare il file di statistiche occorre mettere un particolare comando nello script TCL di simulazione. NAM può essere lanciato dallo script TCL di simulazione oppure con il comando "nam <namfile>".

3. Il primo script TCL

Ciò che andiamo a scrivere ora è un semplice script TCL che contiene anche tutti gli elementi che verranno usati in script più complessi. In un certo senso si può considerarlo un modello. Questi file possono essere scritti con un comune editor ASCII ed è consuetudine che abbiano estensione .tcl. Le righe che iniziano col carattere "#" sono considerate commenti.
La prima cosa da fare è istanziare un oggetto simulatore e definire il tipo di simulazione con i comandi:

```
set ns [new Simulator]
```

```
$ns set cosim_type 0
```

ATTENZIONE: la seconda riga non è riconosciuta dalla distribuzione standard di NS-2 ma è necessaria in quella installata in laboratorio.

Successivamente occorre creare un file per NAM e dire al simulatore di scriverci dentro le opportune statistiche

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

viene ora creata una procedura "finish" da invocare alla fine della simulazione; essa chiude il file di statistiche, lancia NAM e poi termina la simulazione.

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

A questo punto occorre descrivere la topologia di rete (Figura 1); in questo esempio verranno creati due nodi, n0 e n1, collegati da un link full-duplex con capacità 1Mb/s, ritardo di 10ms e con coda del router di tipo tradizionale (DropTail). In NS-2, per convenzione, ogni link ha una coda in ingresso relativa all'interfaccia del router su tale link.

```
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

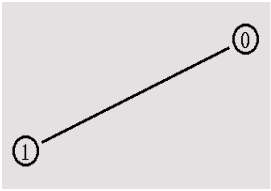

Figura 1. Topologia descritta.

In riferimento al modello ISO/OSI e alla terminologia TCP/IP i nodi modellano il livello 3 (trasmissione/ricezione di pacchetti e routing) presente negli host oppure nei router mentre i link sono elementi di livello 2. Al fine di modellare degli host completi occorre istanziare oggetti che rappresentano il livello 4 (trasporto) e il livello applicazione (creazione/ricezione di dati).

Con i seguenti comandi viene creato un agente UDP su n0 mentre sul nodo n1 viene creato un semplice agente che fa da collettore di pacchetti.

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

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

Occorre poi connettere i due agenti tra loro; si noti che questo avviene anche se i rispettivi nodi non sono adiacenti poichè ciò rispecchia il modello TCP/IP in cui il livello 4 è host-to-host.

```
$ns connect $udp0 $null0
```

Infine occorre creare un'applicazione sopra l'agente UDP in modo che generi dei dati da imbustare nei datagram; qui viene mostrata una sorgente di traffico constant bitrate (CBR) con pacchetti di 500 byte inviati ogni 5ms.

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

Il prossimo comando imposta l'evento di chiamata della procedura "finish" al quinto secondo di simulazione

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

L'uso di "at" è la modalità con cui si inseriscono gli eventi di simulazione; infatti allo stesso modo viene programmata l'accensione e lo spegnimento della sorgente CBR come segue

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

La simulazione inizia col comando

```
$ns run
```

Si può ora lanciare il simulatore; alla fine della simulazione comparirà la finestra di NAM in cui è possibile animare la simulazione. Si notino gli istanti di accensione e spegnimento della sorgente. è possibile clickare sul link e su ciascun pacchetto per ottenere informazioni. Si consiglia di riprovare la simulazione con parametri diversi per il link e la sorgente CBR.

4. Uno scenario più complesso

Vedremo ora uno scenario più complesso con 4 nodi e 3 link (Figura 2)

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

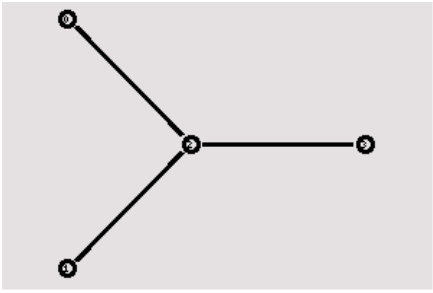

Figura 2. Uno scenario più complesso.

Sui nodi n0 e n1 vengono create due sorgenti UDP/CBR che trasmettono al nodo n3. Si vede bene qui il ruolo di n2 come router.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n2 $null0

set null1 [new Agent/Null]
$ns attach-agent $n3 $null1

$ns connect $udp0 $null1
$ns connect $udp1 $null1
```

Le sorgenti si accendono e spengono secondo i comandi:

```
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
```

Facendo alcuni calcoli si vede che quando le due sorgenti sono contemporaneamente attive il traffico tra n2 e n3 eccede la capacità del link e quindi vengono persi dei pacchetti come visualizzato da NAM.
Per sapere a quale dei due flussi appartengono i pacchetti persi si può colorare i flussi con i comandi:

```
$udp0 set class_ 1
$udp1 set class_ 2
```

Occorre definire i colori per le due classi; questo va fatto all'inizio, subito dopo la creazione dell'oggetto simulatore con i comandi:

```
$ns color 1 Blue
$ns color 2 Red
```

Rieseguendo la simulazione si può osservare se i pacchetti persi sono uniformemente distribuiti tra i due flussi; questo è un modo per valutare la "fairness" del router in n2.

è possibile visualizzare la coda in uscita da n2 e il suo contenuto durante la simulazione con il comando:

```
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

Nel caso si notasse che il comportamento del router non è molto equo si può provare a cambiare la politica di gestione della coda sull'interfaccia di uscita cambiando la dichiarazione del link nel seguente modo:

```
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
```

SFQ sta per "Stochastic Fair Queueing".

5. Uso di xgraph

In questo capitolo vedremo come utilizzare il tool accessorio xgraph per visualizzare grafici sui risultati di simulazione. Sia data la seguente topologia (Figura 3):

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail
```

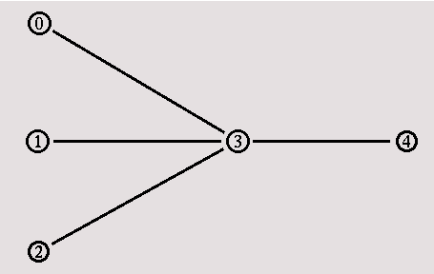

Figura 3. Topologia di rete.

Al fine di automatizzare la creazione delle sorgenti viene creata una procedura parametrica che instanzia gli agenti UDP e instanzia sopra un modello di traffico esponenziale.

```
proc attach-expoo-traffic { node sink size burst idle rate } {
```

```
      #Get an instance of the simulator
      set ns [Simulator instance]

      #Create a UDP agent and attach it to the node
      set source [new Agent/UDP]
      $ns attach-agent $node $source

      #Create an Expoo traffic agent and configure it
      set traffic [new Application/Traffic/Exponential]
      $traffic set packetSize_ $size
      $traffic set burst_time_ $burst
      $traffic set idle_time_ $idle
      $traffic set rate_ $rate

      # Attach traffic source to the traffic generator
      $traffic attach-agent $source
      #Connect the source and the sink
      $ns connect $source $sink
      return $traffic
}
```

Di seguito vengono creati tre agenti destinatari speciali che memorizzano il numero di byte ricevuti. Vengono create le corrispondenti sorgenti con traffico esponenziale con diverso rate di picco.

```
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]
```

All'inizio del file TCL occorre creare 3 file

```
set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]
```

Questi file verranno chiusi alla fine della simulazione; occorre quindi cambiare la procedura "finish".

```
proc finish {} {
      global f0 f1 f2
      #Close the output files
      close $f0
      close $f1
      close $f2
      #Call xgraph to display the results
      exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &
      exit
}
```

Tale procedura chiama ora Xgraph invece di NAM; notare il parametro "-geometry" che va modificato in base alla risoluzione del proprio schermo.

Ora bisogna scrivere la procedura che salva sui file i byte ricevuti dai tre destinatari.

```
proc record {} {
      global sink0 sink1 sink2 f0 f1 f2
      #Get an instance of the simulator
      set ns [Simulator instance]
      #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 bw0 [$sink0 set bytes_]
      set bw1 [$sink1 set bytes_]
      set bw2 [$sink2 set bytes_]
      #Get the current time
      set now [$ns now]
      #Calculate the bandwidth (in MBit/s) and write it to the files
      puts $f0 "$now [expr $bw0/$time*8/1000000]"
      puts $f1 "$now [expr $bw1/$time*8/1000000]"
      puts $f2 "$now [expr $bw2/$time*8/1000000]"
      #Reset the bytes_ values on the traffic sinks
      $sink0 set bytes_ 0
      $sink1 set bytes_ 0
      $sink2 set bytes_ 0
      #Re-schedule the procedure
      $ns at [expr $now+$time] "record"
}
```

La procedura "record" legge il contatore di byte ricevuti dai tre destinatari, lo azzera, usa il valore letto per calcolare il bitrate istantaneo che viene scritto sui rispettivi file assieme all'istante di tempo a cui si riferisce. La procedura viene chiamata all'istante 0 e poi ri-schedula se stessa ogni "time" secondi.

I comandi per programmare gli eventi sono:

```
$ns at 0.0 "record"
$ns at 10.0 "$source0 start"
$ns at 10.0 "$source1 start"
$ns at 10.0 "$source2 start"
$ns at 50.0 "$source0 stop"
$ns at 50.0 "$source1 stop"
$ns at 50.0 "$source2 stop"
$ns at 60.0 "finish"
```

Provare a rifare la simulazione con valori di "time" pari a 0.1 e 1.0 e vedere cosa cambia nella dimensione dei file e nel grafico visualizzato.

Le tracce di statistiche generate con NS-2 possono essere visualizzate anche con il generatore di grafici Gnuplot.

6. Comportamento dinamico della rete

Si vuole ora creare una topologia più complessa (Figura 4) utilizzando un ciclo ed un array per instanziare i nodi.

```
for {set i 0} {$i < 7} {incr i} {
      set n($i) [$ns node]
}
```

```
for {set i 0} {$i < 7} {incr i} {
      $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
```

Da notare che in TCL non serve dichiarare l'array prima di usarlo.

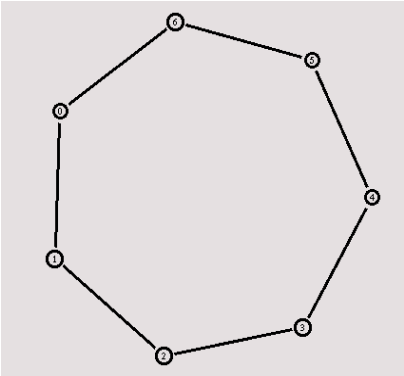

Figura 4. Topologia ad anello.

Creiamo ora del traffico tra il nodo 0 e il nodo 3.

```
#Create a UDP agent and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Se si esegue la simulazione si vedrà che i pacchetti seguono la strada più breve attraverso i nodi 1 e 2.

Si ripeta ora la simulazione introducendo un malfunzionamento al link tra il nodo 1 e il nodo 2 all'istante 1 s e fino all'istante 2 s.

```
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
```

Si nota ora che i pacchetti vengono persi durante il periodo di malfunzionamento.

Si ripeta la simulazione aggiungendo dopo la creazione dell'istanza del simulatore il comando:

```
$ns rtproto DV
```

Si notano ora due cose. 1) piccoli pacchetti transitano nella rete oltre a quelli dei dati. Se si clicka col mouse sopra questi pacchetti si vedrà che essi sono di tipo "rtProtoDV" e quindi appartenenti ad un protocollo di routing che sta "girando" sui nodi. 2) quando il link si interrompe i pacchetti seguono un'altra strada grazie al protocollo di routing.

7. Generazione di un file di traccia

è possibile impostare la scrittura su un file di una traccia completa dei pacchetti che circolano nella rete; si usano i comandi:

```
set traceFile [open out.tr w]
$ns trace-all $traceFile
```

inoltre occorre modificare la procedura “finish” in modo che chiuda anche tale file:

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

Ciascuna riga del file prodotto segnala un evento avvenuto ad un certo pacchetto ed è organizzata in 12 campi (Figura 5).

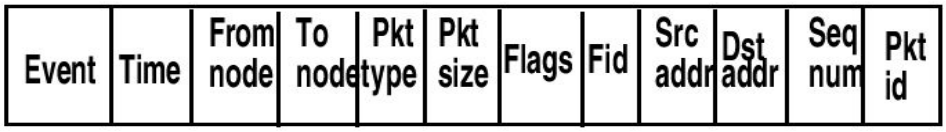

Figura 5. Campi del file di traccia.

I campi hanno il seguente significato:
1) Tipo di evento sul pacchetto: assume 4 possibili valori r, +, -, d corrispondenti a received (all'uscita del link), enqueued (all'ingresso del link), dequeued (all'ingresso del link), dropped (all'ingresso del link).
2) Istante di tempo
3) Primo nodo del link
4) Secondo nodo del link
5) Tipo di pacchetto
6) Dimensione del pacchetto
7) Flags
8) Flow Id
9) Indirizzo sorgente nella forma nodo.porta
10) Indirizzo destinazione nella forma nodo.porta
11) Numero di sequenza all'interno di un flusso
12) ID univoco del pacchetto in NS

Se si vuole monitorare solo un link invece che tutta la rete (per ridurre la dimensione della traccia) si può sostituire

```
$ns trace-all $traceFile
```

con

```
$ns trace-queue $n0 $1 $traceFile
```

La seguente pagina WEB riporta una guida di riferimento per i possibili formati delle tracce generabili in NS
http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats#Normal_trace_formats

Il campo 8 della traccia è il ***Flow Id***, un identificativo di flusso: a ciascun agente della rete può essere assegnato un determinato ID in modo da poter monitorare tutti i pacchetti del traffico da esso generato.

Per assegnare uno specifico Flow Id ad un agente:

```
set $udp0 [new Agent/UDP]
$udp0 set fid_ 1000
```

L'utilizzo di un Flow Id specifico permette di effettuare delle misure sul traffico della rete mediante alcuni dei seguenti script (scaricabili dalla pagina del corso).

`pck-rate-at-node.pl <tracefile> <node> <granularity>`
misura la frequenza di arrivo dei pacchetti al nodo `<node>` in intervalli di tempo sucessivi di ampiezza `<granularity>`

`throughput-at-node.pl <tracefile> <node> <granularity>`
misura il bit-rate arrivato al nodo `<node>` in intervalli di tempo successivi di ampiezza `<granularity>`

`flow-F-at-N.pl <tracefile> <flowID> <node>`
estrae la traccia con i pacchetti appartenenti ad un certo flusso `<flowID>` arrivati ad un certo nodo `<node>`

Se si salva l'output di `flow-F-at-N.pl` e poi lo si passa a `pck-rate-at-node.pl` o a `throughput-at-node.pl` si ottiene il pck-rate e throughput per lo specifico flusso.

`jitter-flow-F-at-N.pl <tracefile> <flowID> <node> <period>`
misura la deviazione standard dei tempi di interarrivo `<period>` dei pacchetti del flusso `<flowID>` al nodo `<node>`

`jitter-flow-F-at-N-granularity.pl <tracefile> <flowID> <node> <period> <granularity>`
misura la deviazione standard dei tempi di interarrivo `<period>` dei pacchetti del flusso `<flowID>` al nodo `<node>` in intervalli di tempo successivi di ampiezza `<granularity>`

Per eseguire ciascuno script, dopo aver generato un tracefile, eseguirli come nell'esempio:

```
./flow-F-at-N.pl out.tr 1000 3
```

8. Flussi TCP

Per creare un agente TCP anzichè UDP usare i seguenti comandi:

```
# TCP connections between n0 and n1
set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $n0 $tcp
$ns_ attach-agent $n1 $sink
$ns_ connect $tcp $sink
```

Per creare una sorgente FTP sopra un agente TCP usare i seguenti comandi:

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 10.0 "$ftp start"
```

Esercizio. Modificare lo scenario del capitolo 4 in modo da avere una sorgente UDP ed una TCP che interferiscono su un tratto della rete. Si colorino i due flussi diversamente e si riduca progressivamente la capacità del link comune osservando cosa succede al flusso TCP.