Universidade de Coimbra
Faculdade de Ciências e Tecnologia

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E COMPUTADORES
Redes de Computadores

Ficha 4 - Network Simulator 2

Ano Lectivo de 2021/2022
www.isi.edu/nsnam/ns/tutorial

---

1) Introdução

A simulação é um elemento chave no projecto de redes informáticas e na avaliação de novas soluções e protocolos de rede. Através da simulação é possível comparar soluções alternativas, evitando o recurso à implementação real de cenários de avaliação, muitas das vezes impossíveis de realizar por questões financeiras ou por motivos tecnológicos.
Existem diversos pacotes comerciais e gratuitos para a simulação de redes de comunicações. Existem simuladores orientados ao estudo de tecnologias específicas, outros orientados para a avaliação de modelos e algoritmos de rede. No entanto, o *Network Simulator 2* (NS-2) tem-se revelado o simulador por excelência nas redes de comunicação, dada a sua crescente utilização. A riqueza deste simulador resulta do seu código aberto e de um trabalho voluntário de muitos utilizadores e de programadores espalhados por todo o mundo.
A sua estrutura generalista, que apresenta um vasto conjunto de módulos adicionais permitindo cobrir qualquer camada protocolar nos sistemas de comunicação, e a sua divulgação fazem do NS-2 o simulador escolhido nos meios académicos e empresariais.

2) Objectivos

Neste trabalho pretende-se desenvolver-se alguns *scripts* Tcl que simulam topologias simples, e avaliar o seu comportamento em diversos cenários utilizando para esse fim o *nam – The Network Animator*.

3) Exemplo 1 - Rede básica

Apresenta-se, de seguida, um conjunto de instruções essenciais e genéricas a qualquer simulação a realizar no NS-2. Nesta secção são ainda apresentadas instruções que irão permitir fazer a monitorização (trace) das simulações.

3.1. Criação de um cenário

Para criar uma nova simulação é necessário definir um objecto *Simulator* a partir do comando:

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

Com o referido comando a variável *ns* torna-se uma instância do objecto *Simulator*.

3.2. Monitorização da simulação

Posteriormente, é necessário criar um ficheiro para armazenar os resultados da simulação. Este ficheiro poderá, posteriormente, ser utilizado pelo *nam* para produzir as animações finais.

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

Na primeira linha do código abre-se o ficheiro out.nam para escrita e atribui-se-lhe o *handler* `nf`. Na segunda linha informa-se o simulador para escrever todos os resultados da simulação no respectivo ficheiro.

O passo seguinte consiste numa sequência de comandos, organizados no procedimento "fim", para que, no final da simulação, se proceda ao fecho do ficheiro de resultados e se evoque o *nam*.

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

Nota: tome em atenção o caminho do ficheiro executável *nam* .

No `NS-2` o escalonamento dos eventos realiza-se através do comando `at`. Assim,

```
$ns at 5.0 "fim"
```

evoca o procedimento fim após 5.0 segundos de tempo de simulação.

3.3. Iniciação da simulação

A simulação é iniciada com o comando:

```
$ns run
```

O conjunto de instruções apresentado poderá ser guardado para posterior utilização, uma vez que é um modelo genérico a qualquer simulação, e ao qual se pode adicionar posteriores comandos de criação de diferentes topologias.
Grave o ficheiro com o nome *exemplo1.tcl*.

3.4. - Tráfego entre dois nós

Esta secção pretende definir uma topologia simples através da geração de tráfego entre dois nós.

3.4.1. Adição de dois nós

Insira as seguintes linhas no ficheiro *exemplo1.tcl* antes da linha "`$ns at 5.0 "fim"`":

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

Um objecto nó é criado através do comando `$ns node`.

3.4.2. Estabelecimento de uma ligação

A linha seguinte define uma ligação entre os dois nós.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

É definida uma ligação duplex entre os nós `n0` e `n1` com a largura de banda de 1 Mbps, com um atraso de 10ms e com uma fila do tipo DropTail.
Guarde o ficheiro com o nome *exemplo1.tcl.*
Para correr o *script* chame o comando "`ns exemplo1.tcl`". No final da simulação o *nam* será iniciado e será visível uma topologia semelhante à apresentada na figura 1.

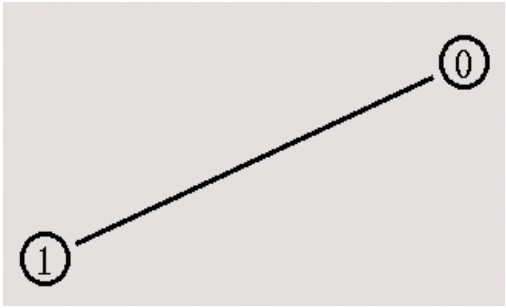

Figura 1

3.4.3. Geração de tráfego

Depois de ter sido definida a topologia é necessário estabelecer as fontes geradoras de tráfego. Para este exemplo vamos colocar no nó `n0` uma aplicação a gerar pacotes para o nó `n1`.
No NS-2 os dados são enviados de um agente para o outro. Assim, é necessário criar dois agentes, um que irá enviar dados do nó `n0` e outro que irá receber os dados no nó `n1`:

```
#Cria um agente UDP e liga-o ao nó n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Cria uma fonte de tráfego CBR e liga-a ao udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

É definido um agente UDP que é ligado ao nó `n0`. Por sua vez, será definida uma fonte de tráfego do tipo CBR (bit rate constante) a gerar pacotes com o tamanho de 500 bytes à taxa de 200 pacotes por segundo (1 em cada 0.005 segundos).

3.4.4. Recepção de tráfego

No NS-2 define-se um agente NULL que irá actuar como um receptor de dados.

```
#Cria um agente Null e liga-o ao nó n1
```

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

3.4.5. *Ligações*

Depois de definidos os dois agentes, é necessário proceder à sua ligação.

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

3.4.6. Período de transmissão

Numa simulação é necessário definir os instantes nos quais a fonte CBR inicia a transmissão de dados e cessa essa mesma transmissão.

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

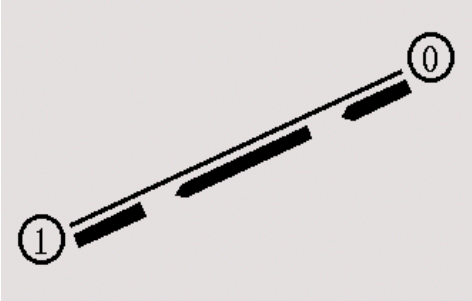

Figura 2

Nesta fase, deverá guardar novamente o ficheiro. Posteriormente, corra o *script* que criou e analise as alterações realizadas no *nam* . Deverá visualizar uma imagem semelhante à apresentada na figura 2.
É possível ainda "clicar" com o rato em qualquer pacote na janela do *nam* para monitoriza-lo. É também possível "clicar" numa determinada ligação para obter informações sobre as suas propriedades.

4) Exemplo 2 - Comportamento de um encaminhador

Neste exemplo irão definir-se quatro nós, onde um deles encaminha os dados enviados por dois dos nós para o quarto nó.

4.1. Adição de dois nós

As linhas de código apresentadas de seguida estabelecem uma rede com quatro nós. As linhas devem ser inseridas no ficheiro *exemplo1.tcl* antes da linha "`$ns at 5.0 "fim"`". Posteriormente, deverá guardar o ficheiro resultante com o nome *exemplo2.tcl*.

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

4.2. Estabelecimento das ligações

O código seguinte estabelece as ligações entre os três nós:

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
```

```
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
```

Se experimentar executar o *script* no *nam* verificará que a distribuição dos nós não é a mais desejável. No entanto, é possível controlar a disposição dos nós e das ligações através da introdução de algum parâmetros opcionais. As linhas seguintes exemplificam o controlo do *layout* da topologia segundo a figura 3.

```
$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
```

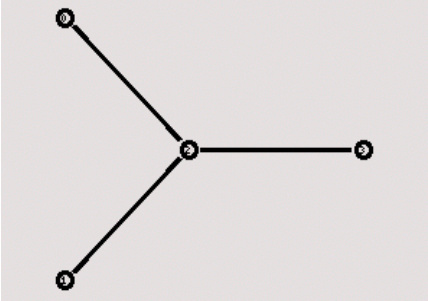

Figura 3

4.3. Geração de tráfego

Para esta topologia irão criar-se dois agentes UDP com fontes de tráfego CBR ligados aos nós n0 e n1. Será criado um terceiro agente NULL, ligado ao nó n3, que irá receber os dados produzidos.

```
#Cria um agente UDP e liga-o ao nó n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Cria uma fonte de tráfego CBR e liga-a ao udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_  500
$cbr0 set interval_  0.005
$cbr0 attach-agent $udp0

#Cria um agente UDP e liga-o ao nó n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

#Cria uma fonte de tráfego CBR e liga-a ao udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_  500
$cbr1 set interval_  0.005
$cbr1 attach-agent $udp1

#Cria um agente Null e liga-o ao nó n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

Os dois agentes CBR devem ser conectados ao agente NULL.

```
$ns connect $udp0 $null0
$ns connect $udp1 $null0
```

É necessário definir os instantes nos quais as fontes CBR iniciam e terminam a transmissão de dados.

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

Poderá experimentar-se o correr o *script* com o comando "`ns exemplo2.tcl`".

4.4. Distinção de fluxos

Tal como se pode observar os dois fluxos são representados no *nam* de cor preta. Desta forma não é possível distinguir quais os pacotes que são descartados: se são todos os pacotes de uma única fonte, de duas fontes ou se o descarte é realizado de forma equitativa.
Para permitir a distinção dos diferentes fluxos o NS-2 suporta a utilização de cores distintas. Este processo pode ser realizado através das linhas seguintes (colocadas antes de "$ns at 5.0 "fim""):

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

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

Estas últimas duas linhas devem ser colocadas imediatamente a seguir à criação do objecto `Simulator`.
Se correr novamente a simulação e visualizar o resultado com o *nam* irá observar que os fluxos já são diferenciáveis. Irá verificar que, na ligação `n2-n3`, um dos fluxos será preponderante, pelo que será interessante verificar o que se passa na fila de saída do encaminhador (`n2`). Ao visualizar a simulação com o *nam* irá observar uma imagem semelhante à da figura 4

4.5. Monitorização da fila

A linha seguinte monitoriza a fila de saída para a ligação de `n2` para `n3` (colocada antes de "$ns at 5.0 "fim""):

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

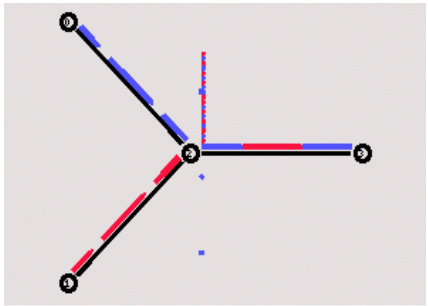

Figura 4

É possível observar agora os pacotes que se encontram na fila e reparar que só os pacotes de uma fonte é que estão a ser descartados. Recorde que foi definida uma fila do tipo *DropTail*, que descarta todos os pacotes que chegam quando tem o *buffer* da fila se encontra cheio.

Iremos modificar o tipo de gestão da fila de espera recorrendo a políticas mais justas. Defina uma fila SFQ (*stochastic fair queueing*) e altere a definição do *link* `n2-n3` para:

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

Se executar novamente a simulação irá verificar que o descarte de pacotes se realiza de forma mais equitativa.

5) Exercício 1: Através de uns cálculos simples também facilmente se conclui que, para o exercício anterior, a ligação entre os nós n2 e n3 apresenta uma largura de banda inferior ao tráfego gerado para o período de 1 e 4 segundos. Prove então, efetuando cálculos simples, que a ligação entre os nós n2 e n3 apresenta uma largura de banda inferior ao tráfego gerado para o período de 1 e 4 segundos.

6) Dinâmica de rede

Neste exemplo irá observar-se o comportamento de uma rede numa situação de falha em uma das ligações.

6.1. Criação da topologia

As linhas de código apresentadas de seguida definem uma rede com sete nós. As linhas devem ser inseridas no ficheiro *modelo.tcl* antes da linha "`$ns at 5.0 "fim"`". Posteriormente deverá gravar o ficheiro com o nome *exemplo3.tcl*.

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

Neste exercício utilizou-se o ciclo de repetição para facilitar a definição dos nós e o seu armazenamento no array n(). É uma solução mais expedita, à qual se recorre muitas das vezes quando se pretendem criar muitos nós.
De seguida, irão realizar-se as ligações entre os nós adoptando-se uma topologia circular. O código apresentado permite que cada nó se ligue ao nó vizinho. No final, o último nó será ligado ao primeiro nó.

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

A topologia obtida será semelhante à apresentada na figura 5. Caso o aspecto seja bastante distinto deste deverá carregar no botão "re-layout" para aproximar o aspecto àquele da figura.

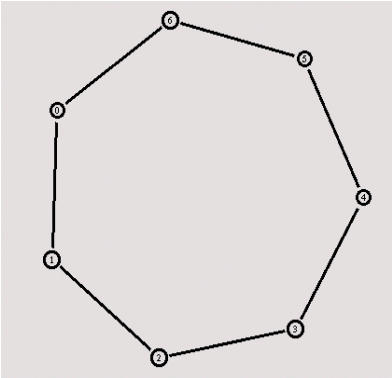

Figura 5

6.2. Geração de tráfego

Nesta etapa irá gerar-se tráfego entre dois dos nós. Cria-se um agente UDP como fonte de tráfego do tipo CBR no nó 0 e um agente NULL ligado ao nó n3.

```
#Cria um agente UDP e liga-o ao nó n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

#Cria uma fonte de tráfego CBR e liga-a ao udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_  500
$cbr0 set interval_  0.005
$cbr0 attach-agent $udp0

#Cria um agente Null e liga-o ao nó n(3)
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”
```

Ao visualizar a simulação no *nam* irá verificar que o tráfego tomará o caminho mais curto, utilizando para tal a ligação que une os nós n(0), n(1), n(2) e n(3).

6.3. Introdução de uma falha na topologia

Agora irá simular-se uma falha da ligação dos nós n(1) e n(2). Para esse efeito deverá usar-se o comando **`rtmodel-at`**, que tanto pode ser usado em nós como em ligações. Este comando permite, através de alguns parâmetros, introduzir eventos na topologia. Assim, a ligação entre os nós n(1) e n(2) será colocado em baixo durante um segundo.

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

Correndo novamente o *script* poderá observar no *nam* que, quando a ligação se encontra em baixo, o nó n(3) deixa de receber tráfego. Por outro lado constata-se que existe uma ligação física alternativa entre os nós n(0) e n(1) mas que esta não é tomada. Porquê?

Para resolver o problema deverá adoptar-se um algoritmo de encaminhamento dinâmico. Para definir um cenário com encaminhamento dinâmico adicione a linha apresentada em baixo, imediatamente a seguir à definição do objecto *Simulator*.

```
$ns rtproto DV
```

Esta linha irá definir no cenário um encaminhamento baseado no *Distance Vector*. Este protocolo de encaminhamento envia periodicamente (tempo definido pelo parâmetro `advertInterval`) uma mensagem de actualização de caminhos.
Quando a ligação entre os nós n(1) e n(2) é interrompida, o nó n(3) deixa de receber pacotes. Só depois da mensagem de actualização das tabelas de encaminhamento ser enviada é que este nó recomeça a receber os pacotes, mas agora por outro caminho.

7) Exercício 2

Depois de testar os exemplos anteriores, pretende-se que, com base no código da alínea 4, implemente uma rede de acordo com a figura:

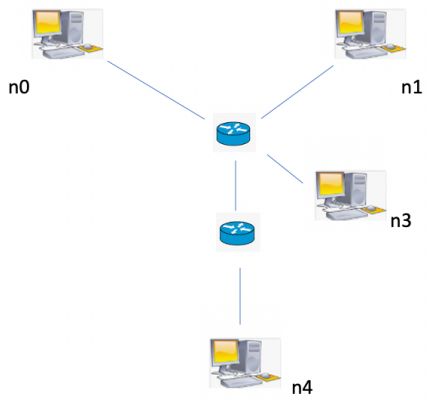

Figura 6 - Rede a implementar

No entanto, para este exercício, deverão existir 2 nós emissores e 2 nós receptores.
O nó n0 envia pacotes de côr azul para o nó n3, e o nó n1 envia pacotes de côr amarela para o nó n4.

Deverá dar o nome de *exercicio2.tcl* ao ficheiro e começar por definir os vários nós:

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
```

É necessário ainda adicionar as ligações entre os nós, como fez no exercício da alínea 4, podendo utilizar um qualquer tipo de gestão da fila de espera.

O tráfego existente deverá ser:

```
#Cria um agente UDP e liga-o ao nó n0
set udp0 [new Agent/UDP]
```

```
$ns attach-agent $n0 $udp0

#Cria uma fonte de tráfego CBR e liga-a ao udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Cria um agente UDP e liga-o ao nó n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

#Cria uma fonte de tráfego CBR e liga-a ao udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#Cria um agente Null e liga-o ao nó n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

#Cria um agente Null e liga-o ao nó n4
set null1 [new Agent/Null]
$ns attach-agent $n4 $null1
```

Os dois agentes CBR devem ser conectados ao respetivo agente NULL.

É necessário definir os instantes nos quais as duas fontes CBR iniciam e terminam a transmissão de dados.

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

Não se esqueça de adicionar as côres aos pacotes.

7) Exercício 3

Pretende-se fazer o download de um pequeno jogo de 1 Kbytes recorrendo a um servidor que se situa em Bruxelas, a uma distância de 2.000 Km, e que utiliza um cabo coaxial com uma velocidade de 4 Mbps.

Utilizando o NS-2, construa o ambiente de trabalho, configurando correctamente os respectivos tempos de propagação e taxas de transmissão.
Corra a simulação e complemente as conclusões com os cálculos analíticos.
Para o envio dos 1 KBytes de informação recorra ao parâmetro *maxpkts_* que, para aplicações CBR, determina o número máximo de blocos enviados.