FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Mobilidade em Sistemas de Telecomunicações

2010/2011

Introdução ao simulador de redes

"*The Network Simulator ns-2*"

Rodolfo Oliveira

rado@fct.unl.pt

Conteúdo

1. Introdução

O simulador "*The Network Simulator ns-2"* [1] (consultar o anexo 2 para instruções de instalação), abreviado neste documento por ns-2, é um simulador *open source* baseado em eventos discretos e especialmente vocacionado para simulação de redes. O projecto que lhe deu origem começou em 1989. Porém, nos últimos anos, evoluiu substancialmente, sendo o simulador mais popular na comunidade de investigação, utilizado nas principais universidades e institutos de investigação.

O simulador trabalha ao nível do pacote/trama, sendo muito popular a sua utilização em trabalhos que incluam os seguintes temas:

- Protocolos de transporte, nomeadamente o TCP (inclui módulos com as versões Reno, Tahoe, Vegas e Sack, etc.);
- Protocolos de controlo de acesso ao meio (inclui módulos de TDMA, norma 802.3, norma 802.11, etc.);
- Encaminhamento para redes *ad hoc* (inclui os protocolos AODV, DSDV, DSR e TORA, etc.);
- Protocolos específicos para redes de sensores sem fios (inclui os protocolos diffusion e gaf, etc);
- Protocolos Multicast;
- Protocolos para comunicações via satélites;
- etc.

Dado que o simulador é *open source* e se encontra devidamente documentado, facilmente se analisam as realizações dos protocolos já incluídos na distribuição e se desenvolvem novos protocolos. Embora a distribuição do simulador não inclua alguns protocolos interessantes, tais como o protocolo de encaminhamento BGP, ou do protocolo de encaminhamento OLSR, não é difícil encontrar múltiplas realizações de módulos que implementam os protocolos, principalmente nas páginas dos diversos grupos de investigação ligados a empresas ou a universidades. Estas são as principais razões que justificam a sua grande popularidade.

2. Descrição do Simulador

O simulador foi inicialmente escrito no ambiente Unix, baseando-se num sistema de eventos discretos. Basicamente, todas as actividades de interacção originadas pelos nós são traduzidas em eventos. Estes eventos podem representar o expirar de um relógio, o início de envio de uma trama para o canal, o aumento da janela de congestão do TCP, etc. Uma vez originados os eventos, estes são guardados numa fila de espera na ordem pela qual são agendados para serem executados. O tempo de simulação avança de acordo com os eventos executados.

A figura 1 pretende exemplificar a situação em que o nó 1 envia um pacote para o nó 2 no instante de simulação 1.5 segundos e, o nó 2 recebe o pacote no instante 1.7 s. Após isso, o nó 2 envia um pacote de reconhecimento (*Acknowledge*) para o nó 1 no instante 1.8 s, o qual é recebido pelo nó 1 no instante 2.0 s. Neste exemplo, são gerados 4 eventos nos instantes temporais referidos. Esses

eventos são colocados na fila de eventos de forma ordenada, ficando os eventos com instantes menores na cabeça da fila para que sejam os primeiros a serem executados.

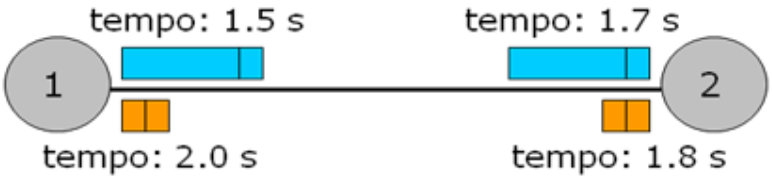

Figura 1 – Exemplo demonstrativo da geração de eventos.

Quando o nó 1 envia o pacote, executando um evento de envio de pacote no instante 1.5s, agenda um evento de recepção no nó 2 para o instante 1.7s. Após se ter executado o evento do instante 1.5s, o simulador retira-o da lista de eventos e inicia o próximo, que neste exemplo é o evento de recepção no instante 1.7s. Desta forma, o simulador avança o tempo de simulação de 1.5s para 1.7s, pois não existem eventos agendados na fila de espera entre estes dois instantes. A simulação termina quando não existirem mais eventos na fila de espera.

O simulador utiliza duas linguagens de programação:

- A linguagem C++ é utilizada para programar os módulos responsáveis pela execução dos protocolos ou aplicações. O comportamento detalhado do protocolo exige uma linguagem de programação de sistemas de baixo nível e, neste aspecto, a utilização da linguagem C++ permite a manipulação de Bytes, processamento de pacotes, e implementação dos algoritmos. Além disso apresenta um tempo de execução computacional muito baixo.

- A linguagem Tcl é utilizada para definir de uma forma rápida as parametrizações dos módulos programados em C++. Dessa forma, é possível parametrizar, através de um script Tcl, um conjunto de módulos contidos nos diversos nós da rede num curto espaço de tempo. Além disso, como o Tcl acede directamente aos atributos dos objectos programados em C++, a alteração de um parâmetro no script Tcl não implica a compilação de todo o código C++.

De uma forma geral, a linguagem C++ é utilizada quando se deseja alterar o comportamento de um módulo existente ou alguma outra operação em que seja necessária a manipulação de pacotes/tramas (consultar o anexo 3 para integrar o código do simulador num projecto em ambiente ECLIPSE). Para a configuração/modificação do cenário a simular utiliza-se a linguagem Tcl.

A figura 2 ilustra o exemplo apresentado na figura 1, onde se pretendem simular uma rede constituída por 2 nós. O script Tcl começa por definir uma instância do simulador representada pela variável ns_. Após isso define o *array* node_ com dois nós, inserindo nas posições 0 e 1 do array uma instanciação da classe MobileNode, a qual executa o comportamento do protocolo de comunicação utilizado pelos nós. Os objectos MobileNode são programados em C++. É fácil criar novos nós num script Tcl utilizado para executar as simulações. Enquanto os objectos C++ são

programados uma única vez e integrados no simulador, os scripts Tcl permitem utilizar esses objectos em múltiplos cenários de simulação.

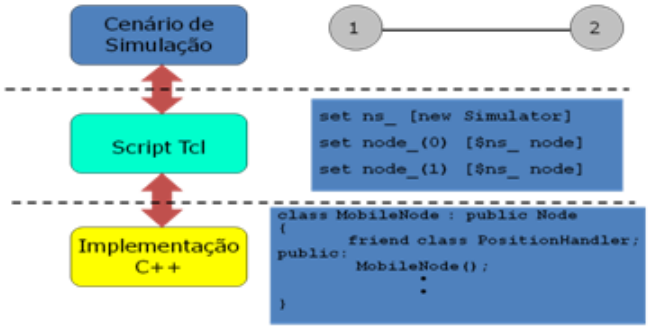

Figura 2 – Ambiente de Simulação.

3. Organização das Camadas de Protocolos

Nesta secção descrevem-se as camadas que constituem a pilha de protocolos utilizada pelo simulador. É nestas camadas que se devem incluir as classes que implementam os módulos dos protocolos a simular. Daí que o simulador apresente classes mãe que constituem a interface para as camadas, sendo cada módulo programado através de um método de herança da classe mãe.

Descrevem-se de seguida cada uma das camadas de protocolos do simulador, para as quais se podem desenvolver novos módulos (programados em C++). As camadas encontram-se esquematizadas na figura 3, apresentando-se as relações entre elas para o caso específico do protocolo de encaminhamento DSDV.

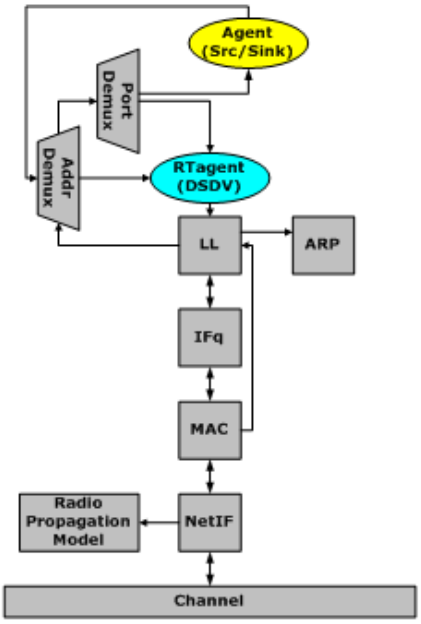

Figura 3 – Esquematização da pilha de protocolos do simulador ns-2.

Começando pela camada de mais baixo nível denominada “Radio Propagation Model”, esta é responsável por simular o modelo de propagação de rádio, caso se simulem redes sem fios. É nesta camada que se podem utilizar modelos relacionados com o tipo de antenas utilizadas na simulação. Esta camada decide ainda quando é que uma trama pode ser recebida por um determinado nó, dado que o emissor o transmitiu com uma dada potência, se encontra a uma dada distância e, utiliza uma dada frequência de rádio. Esta decisão é possível porque a camada utiliza modelos para quantificar as perdas de propagação, incluindo os modelos *free space* e *two-ray*.

As interfaces de hardware utilizadas por um nó para aceder ao canal são modeladas na camada “NetIf”. Estas tratam dos parâmetros relacionados com a integridade do sinal, colisões e erros de transmissão. Além disso, associam a cada transmissão os valores de potência, frequência e outros utilizados na simulação.

A terceira camada da pilha implementa o protocolo de acesso ao meio (MAC). O código das classes que implementam esta camada pode muitas vezes ser uma transcrição para C++ do *firmware* das placas de comunicações, responsáveis pela implementação real do protocolo de acesso ao meio. O simulador já inclui alguns protocolos devidamente testados, tais como a norma IEEE 802.11.

A camada “IFq – interface queue” é uma fila de espera. Esta existe para dar prioridade aos pacotes de encaminhamento. Caso seja utilizado um protocolo nesta camada, denominado *PriQueue,* todos os pacotes de encaminhamento são inseridos na cabeça da lista de espera, de forma a serem os primeiros a serem servidos. Esta camada permite ainda a configuração de filtros com endereços de pacotes que se desejem remover da fila.

A camada responsável pelo nível de ligação (“LL – link layer”) executa os protocolos de ligação de dados. Basicamente os protocolos definidos nesta camada são responsáveis pela fragmentação e reagrupamento dos pacotes. Esta camada está ligada ao protocolo ARP (“Address Resolution Protocol”), utilizado para realizar a resolução dos endereços IP associados aos endereços MAC.

Os protocolos de encaminhamento são especificados na camada “RTagent”. Na camada “agent” são especificadas as aplicações geradoras de tráfego. Esta camada pode ser vista como a camada aplicação da pilha OSI. O simulador já é distribuído com geradores clássicos de tráfego implementados para esta camada, tais como aplicações CBR, TCP, Sink, FTP, etc.

4. Exemplos de parametrização de simulações

Nesta secção exemplifica-se a definição de um script Tcl para executar uma simulação de dois nós móveis ligados sem fios. Pretende-se que o nó 0 envie dados utilizando uma aplicação FTP para o nó 1. Os nós encontram-se numa área de 500x500 metros e ambos se movimentam. A figura 4 ilustra a simulação a definir.

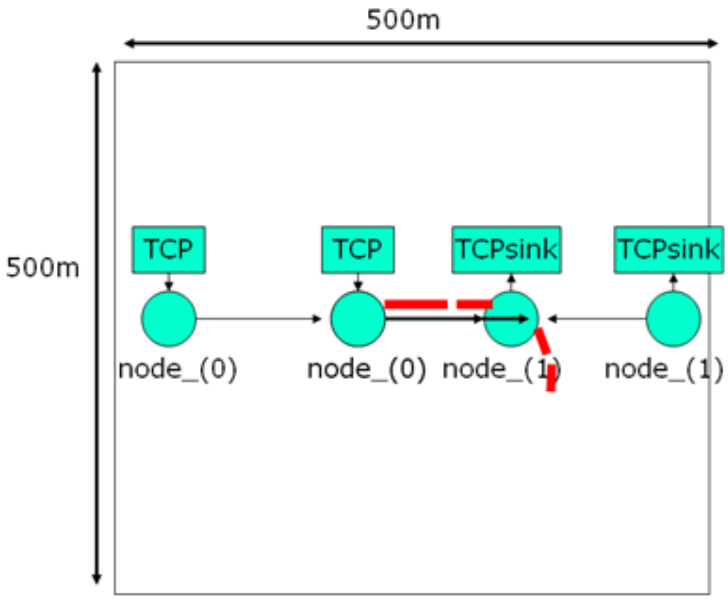

Figura 4 – Simulação pretendida.

4.1 Definição de variáveis

É boa prática começar por definir todas as variáveis que são utilizadas na simulação. Em Tcl o comando *set val* atribui um valor a uma variável (*val*). Tal como se poderá observar no código Tcl apresentado a seguir, começa-se por definir a variável *chan* que representa o tipo de canal utilizado. Neste caso, a variável é definida com um identificador de canal do tipo sem fios. As restantes variáveis representam, por ordem, o tipo de modelo de propagação rádio, o tipo de antena, o tipo de nível de ligação, o tipo de fila de espera IFq, o comprimento dessa fila de espera, o tipo de interface física utilizada, o protocolo de acesso ao meio, o protocolo de encaminhamento, o número de nós simulados e, finalmente, a definição do comprimento e da largura da região a simular (definida em metros).

```
#===================================================================
# Define options
#===================================================================
set val(chan) Channel/WirelessChannel    ;# channel type
set val(prop) Propagation/TwoRayGround   ;# radio-propagation model
set val(ant) Antenna/OmniAntenna         ;# Antenna type
set val(ll) LL                           ;# Link layer type
set val(ifq) Queue/DropTail/PriQueue     ;# Interface queue type
set val(ifqlen) 50                       ;# max packet in ifq
set val(netif) Phy/WirelessPhy           ;# network interface type
set val(mac) Mac/802_11                  ;# MAC type
set val(rp) DSDV                         ;# ad-hoc routing protocol
set val(nn) 2                            ;# number of mobilenodes
set val(x)  500                          ;
set val(y)  500                          ;# x-y simulation's area
```

Seguidamente, atribui-se à variável *ns_* uma instanciação da classe “Simulator”. *ns_* passa a ser um objecto que serve para controlar as operações do simulador.

```
set ns_ [new Simulator]
```

Quando as simulações acabam, o ns2 possibilita o acesso aos dados reportados pelos distintos nós/protocolos contidos na simulação (*trace data*) bem como a todo o percurso realizado por cada um dos nós móveis. Para se definir o ficheiro para onde os dados dos protocolos são escritos, começa-se por abrir um ficheiro para escrita "simple.tr", o qual é associado a um objecto *tracefd*. Posteriormente indica-se ao objecto simulador *ns_* o objecto ficheiro *tracefd* (linha 2). A definição do ficheiro de trace de movimentos (objecto "namtrace") é executado de forma semelhante.

```
#===================================================================
# Define trace files
#===================================================================
set tracefd [open simple.tr w]
$ns_ trace-all $tracefd
set namtrace [open movement.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
```

Para definir a área onde os nós de podem movimentar, instancia-se um objecto do tipo *Topography.* Este objecto é depois informado acerca da área de simulação, através da invocação do método *load_flatgrid.* No exemplo abaixo define-se o quadrado com 500 metros por 500 metros.

```
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

O comando *create-god* recebe como argumento o número de nós e é usado por um deus (God) para criar uma matriz com informação da conectividade da topologia.

```
set god_ [create-god $opt(nn)]
```

4.2 Configuração de um nó móvel sem fios

Um nó é configurado através da especificação do tipo de protocolos que utiliza nas diferentes camadas que constituem a pilha de protocolos. Para o efeito, invoca-se no objecto *ns_* o método *node-config,* o qual permite especificar os protocolos usados em cada camada. No exemplo representado a seguir, começa-se por definir o protocolo de encaminhamento, o tipo de camada LL, o tipo de fila de espera IFq, o seu comprimento, o tipo de antena, o modelo de propagação, o tipo de interface física, e o tipo de canal, pela ordem representada. Posteriormente indica-se a topologia definida na secção 4.1 e representada na variável "topo". Nas linhas seguintes, indica-se o estado dos traces nas diferentes camadas. O comando "agentTrace ON" activa a capacidade dos agentes definidos poderem reportar os seus estados (no ficheiro de trace) à medida que a simulação decorre. Da mesma forma existem os comandos "routerTrace" e "macTrace" que servem para activar/inibir a possibilidade do nível de encaminhamento ou do sub-nível mac reportarem as suas operações. Já o comando "movementTrace", é utilizado para o registo do movimentos dos nós simulados.

No ciclo "for" instanciam-se os diferentes nós incluindo as referencias para os objectos no *array* "node_". É inibida a realização de movimentos aleatórios através do comando "random-motion 0".

```
$ns_ node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
             -ifqType $val(ifq) \
             -ifqLen $val(ifqlen) \
             -antType $val(ant) \
             -propType $val(prop) \
             -phyType $val(netif) \
             -channelType $val(chan) \
             -topoInstance $topo \
             -agentTrace ON \
             -routerTrace ON \
             -macTrace ON\
            -movementTrace ON

for {set i 0} {$i < $val(nn) } {incr i} {
          set node_($i) [$ns_ node ]
          $node_($i) random-motion 0 ;# disable random motion
}}
```

4.3 Configuração de Movimento dos Nós

Existem basicamente duas formas de obter o movimento dos nós: através de uma configuração estática definida no script Tcl, ou através de um ficheiro obtido automaticamente através de uma aplicação que realize um determinado modelo de mobilidade.

Nas duas formas apresentadas anteriormente utiliza-se o mesmo método para especificação do movimento. O método, apresentado nos próximos dois exemplos, começa por definir a posição inicial dos diferentes nós da seguinte forma:

```
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 390.0
$node_(1) set Y_ 385.0
$node_(1) set Z_ 0.0
```

Posteriormente, definem-se movimentos para cada um dos nós. No exemplo seguinte, a primeira linha indica que no instante de simulação 50s o nó 1 se move para a posição (x=25.0, y=20.0) com uma velocidade de 15.0 m/s. As seguintes linhas podem ser lidas da mesma forma, embora com outras parametrizações.

```
# Node_(1) starts to move towards node_(0)
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"

# Node_(1) then starts to move away from node_(0)
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0
```

4.4 Estabelecimento de fluxos de tráfego

A figura 5 exemplifica o fluxo de dados que se pretende definir entre os dois nós 0 e 1. O nó 0 deverá simular o protocolo FTP (*File Transfer Protocol*), utilizando para isso o protocolo de transporte TCP (*Transmission Control Protocol*).

Figura 5 – Fluxo de tráfego a definir entre os dois nós.

Exemplifica-se abaixo a especificação do fluxo de tráfego apresentado na figura 5. Começa-se por instanciar um objecto responsável por realizar o protocolo TCP, o qual é denominado de "tcp" (linhas 1 e 2). O nó 1 necessita de receber informação vinda do nó 0. Daí ser definido um objecto de recepção de dados (denominado "sink") (linha 4). Os objectos "tcp" e "sink" são associados aos nós móveis através do comando "attach-node" (linhas 3 e 5). Posteriormente é realizada a conexão entre os agentes "tcp" do nó 0 ao "sink" do nó 1 através do método "connect" (linha 6).

A aplicação FTP é instanciada, sendo representada pelo objecto "ftp". Após ser associada ao protocolo TCP, já associado ao nó 1 (linha 8), a aplicação FTP inicia a geração de dados no instante 10.0s. Estas operações encontram-se realizadas nas duas últimas linhas do exemplo.

```
#=================================================================
# Define Traffic Generators
#=================================================================
set tcp [new Agent/TCP]
$tcp set class_ 1   #TCP Tahoe
$ns_ attach-agent $node_(0) $tcp
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 10.0 "$ftp start"
```

4.5 Controlo da simulação

A simulação é controlada pelos comandos inseridos no próprio script Tcl. No exemplo abaixo, executam-se todos os procedimentos para finalizar a simulação quando estiverem decorridos 150.0001 segundos de simulação. Começa-se por invocar o método "reset" em todos os nós, o que invocará o destrutor de todas as classes dos módulos que constituem a pilha de protocolos (linhas 1

a 3). Na linha 4 invoca-se o comando "stop" no objecto simulador "ns", afixando na consola no instante 150.0002 a mensagem "NS EXITING…" (linha 5). Nas linhas 6 a 13, define-se um procedimento Tcl que é invocado quando a simulação é parada.

```
#=================================================================
# Simulation Control
#=================================================================
for {set i 0} {$i < $val(nn) } {incr i} {
$ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0001 "stop"
$ns_ at 150.0002 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
        global ns_ tracefd
        global ns_ namtrace
        $ns_ flush-trace
        close $tracefd
        close $namtrace
        exit 0
}
```

No final do script insere-se o comando responsável pelo início da simulação, o que é realizado através da invocação do método "run" no objecto "ns_".

```
puts "Starting Simulation..."
$ns_ run
```

4.6 Execução do script Tcl

O anexo 1 apresenta o script Tcl final do exemplo descrito "my_first_script.tcl", o qual deverá ser executado na linha de comandos através do comando:

```
ns my_first_script.tcl
```

4.7 Geração de padrão de movimento dos nós

No exemplo anterior os movimentos dos nós foram especificados através de uma configuração estática definida no script Tcl. Aborda-se nesta secção outra forma de especificação: através de um ficheiro obtido automaticamente através de uma aplicação que realize um determinado modelo de mobilidade.

A aplicação "setdest" contida na pasta "indep-utils/cmu-scen-gen/setdest/" é uma realização do modelo *"Random-waypoint"*, a qual gera automaticamente um ficheiro de mobilidade de nós a partir da parametrização pretendida. Aconselha-se a parametrização segundo a Universidade de Michigan (versão 2), pois permite obter resultados mais precisos. A parametrização da aplicação "setdest" é realizada através do parâmetros definidos a seguir.

```
<modified 2003 U.Michigan version (version 2)>
 ./setdest        -v <2> -n <nodes> -s <speed type> -m <min speed> -M <max speed>
                  -t <simulation time> -P <pause type> -p <pause time> -x <max X> -y <max Y>
```

Criado o ficheiro contendo o padrão de movimento dos nós (aqui denominado “scen_xpto”), não se declara no script a especificação estática apresentada na secção 4.3. Em vez disso, realiza-se a inserção do conteúdo do ficheiro de padrão de movimentos no espaço de execução Tcl.

```
set val(sc)         "scen_xpto"; # scenario file
## loading scenario file
source $val(sc)
```

4.8 Outras configurações

Definição do alcance de transmissão dos nós

A definição do alcance de rádio dos nós, quando é especificado o mac IEEE 802.11, é realizada através da inserção de uma das seguintes linhas no script Tcl (consoante o alcance desejado):

```
# 100 meters ../indep-utils/propagation
#Phy/WirelessPhy set RXThresh_ 1.92278e-06; #10 meters
#Phy/WirelessPhy set RXThresh_ 7.69113e-08; #50 meters
#Phy/WirelessPhy set RXThresh_ 3.00435e-08; #80 meters
Phy/WirelessPhy set RXThresh_ 1.42681e-08; #100 meters
#Phy/WirelessPhy set RXThresh_ 8.91754e-10; #200 meters
#Phy/WirelessPhy set RXThresh_ 1.76149e-10; #300 meters
#Phy/WirelessPhy set RXThresh_ 5.57346e-11; #400 meters
#Phy/WirelessPhy set RXThresh_ 2.28289e-11;#500 meters
```

As parametrizações acima são obtidas com a aplicação contida na pasta “indep-utils/propagation”. Para mais informações acerca desta aplicação deve consultar o manual [2].

Definição de padrões de geração de tráfego

É possível gerar automaticamente padrões de geração de tráfego, tal como no caso da geração automática de movimentos a partir de um modelo. Para mais informações acerca desta configuração deve consultar o manual [2].

5. Análise dos ficheiros de *trace*

5.1 Traces dos protocolos

O ficheiro “simple.tr”, gerado quando se executa o script de exemplo apresentado no anexo 1, contém informação dos traces activos no script Tcl. Descreve-se neste capítulo o significado dos dados contidos nesse ficheiro. Os dados seguintes apresentam as primeiras linhas do ficheiro.

```
s 0.029290548 _1_ RTR  --- 0 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 0.029365548 _1_ MAC  --- 0 message 90 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
s 1.119926192 _0_ RTR  --- 1 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
s 1.120361192 _0_ MAC  --- 1 message 90 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
M 10.00000 0 (5.00, 2.00, 0.00), (20.00, 18.00), 1.00
s 10.000000000 _0_ AGT  --- 2 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 10.000000000 _0_ RTR  --- 2 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 12.941172739 _1_ RTR  --- 3 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 12.941787739 _1_ MAC  --- 3 message 90 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
s 13.000000000 _0_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 13.000000000 _0_ RTR  --- 4 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 13.242656084 _0_ RTR  --- 5 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
s 13.242891084 _0_ MAC  --- 5 message 90 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
s 19.000000000 _0_ AGT  --- 6 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 19.000000000 _0_ RTR  --- 6 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 24.799296167 _1_ RTR  --- 7 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 24.799571167 _1_ MAC  --- 7 message 90 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
```

```
s 27.719583723 _0_ RTR  --- 8 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
s 27.720098723 _0_ MAC  --- 8 message 90 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
s 31.000000000 _0_ AGT  --- 9 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 31.000000000 _0_ RTR  --- 9 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 39.083332528 _1_ RTR  --- 10 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 39.083947528 _1_ MAC  --- 10 message 90 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
s 40.918940913 _0_ RTR  --- 11 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
s 40.919375913 _0_ MAC  --- 11 message 90 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
M 50.00000 1 (390.00, 385.00, 0.00), (25.00, 20.00), 15.00
…
D 76.430670539 _0_ IFQ  ARP 4 tcp 80 [0 0 0 800] ------- [0:0 1:0 32 1] [0 0] 0 0
```

A primeira letra de cada linha especifica o tipo de operação descrita no trace. As operações de envio são identificadas pela letra s, a letra r identifica uma recepção, a letra D identifica um DROP (retirada indevida do pacote da fila de espera), a letra f identifica uma situação de reencaminhamento (*forward*) e a letra c descreve uma colisão na camada MAC. A letra M é utilizada para reportar posições de um determinado nó.

Para as linhas do tipo s, f, r, D e c, logo após cada letra, existe um espaço que divide o próximo campo. Tendo como exemplo a linha abaixo, descreve-se de seguida cada um dos campos da linha, os quais são separados por um espaço.

```
s 0.029365548 _1_ MAC  --- 0 message 90 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
```

O campo "0.029365548" especifica o tempo em que a operação ocorreu. Após o tempo surge o nó onde foi executado a operação ("_1_" indica que foi executada no nó 1). O próximo campo (MAC, RTR, AGT, etc.) identifica a camada que gerou a operação. O campo representado por "---" justifica a razão pelo qual foi realizado o trace. Segue-se um campo identificador do pacote (0 na linha acima) sendo seguidos do nome (message, tcp, etc.) e do tamanho(90 bytes) do pacote. O campo "[0 ffffffff 1 800]" identifica o tempo de duração do pacote contido no cabeçalho MAC (0), o endereço de MAC de destino (ffffffff – endereço *broadcast*), endereço MAC de quem enviou o pacote (1), e o tipo de cabeçalho MAC do pacote (800 - ARP). De seguida, é representado o estado das 7 flags do pacote "-------", sendo seguido do cabeçalho IP "[1:255 -1:255 32 0]". Este cabeçalho identifica o endereço de IP do nó que originou o pacote "1" seguido do porto de origem ":255". Da mesma forma se representa o endereço e o porto de destino "-1:255". O campo "32" representa o campo TTL (*time-to-live*), sendo seguido o campo "0" que indica que o próximo *hop* (0 indica o nó 0 ou uma situação de *broadcast;* caso seja diferente de 0 este campo indica o endereço do próximo *hop*).

5.2 Nam - visualizador da rede

O ficheiro "movement.nam", gerado quando se executa o script de exemplo apresentado no anexo 1, apresenta a informação relativa às posições e movimentos efectuados pelo nó. Este ficheiro é utilizado pela aplicação NAM (executando o comando de linha "*nam movement.nam"*), que integra o pacote de aplicações da distribuição do simulador ns-2. Este visualizador permite observar os movimentos/posições efectuados pelos nós.

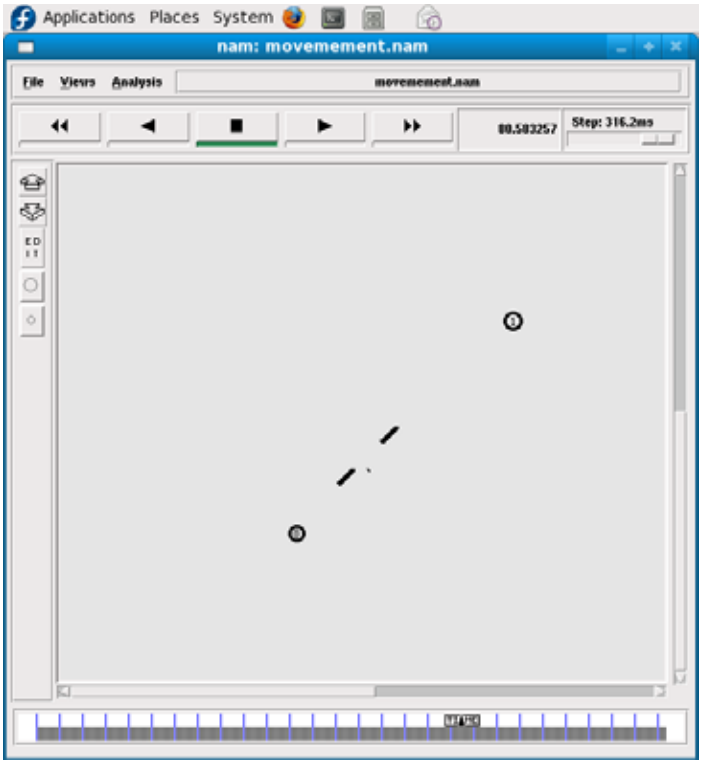

Figura 6 – Aspecto do visualizador de rede NAM.

Bibliografia

[1] http://www.isi.edu/nsnam/ns/

[2] Manual do simulador. Disponível em http://www.isi.edu/nsnam/ns/

Anexo 1 – ficheiro "my_first_script.tcl"

```
#===================================================================
# My first Tcl script
#
# Rodolfo Oliveira - Sept 2009
#
#===================================================================


#===================================================================
# Define options
#===================================================================
set val(chan) Channel/WirelessChannel    ;# channel type
set val(prop) Propagation/TwoRayGround   ;# radio-propagation model
set val(ant) Antenna/OmniAntenna         ;# Antenna type
set val(ll) LL                           ;# Link layer type
set val(ifq) Queue/DropTail/PriQueue     ;# Interface queue type
set val(ifqlen) 50                       ;# max packet in ifq
set val(netif) Phy/WirelessPhy           ;# network interface type
set val(mac) Mac/802_11                  ;# MAC type
set val(rp) DSDV                         ;# ad-hoc routing protocol
set val(nn) 2                            ;# number of mobilenodes
set val(x)  500                          ;
set val(y)  500                          ;# x-y simulation's area

#===================================================================
# New ns2 instance
#===================================================================
set ns_ [new Simulator]

#===================================================================
# Define trace files
#===================================================================
set tracefd [open simple.tr w]
$ns_ trace-all $tracefd
set namtrace [open movement.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

#===================================================================
# Define Topology
#===================================================================
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)


create-god $val(nn)

#===================================================================
# Configure nodes
#===================================================================
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
```

```
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
            -movementTrace ON

for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node ]
      $node_($i) random-motion 0 ;# disable random motion
}

#===================================================================
# Define Node's Movement
#===================================================================
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 390.0
$node_(1) set Y_ 385.0
$node_(1) set Z_ 0.0

# Node_(1) starts to move towards node_(0)
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"

# Node_(1) then starts to move away from node_(0)
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0"


#===================================================================
# Define Traffic Generators
#===================================================================
## TCP
set tcp [new Agent/TCP]
$tcp set class_ 1   #TCP Tahoe
$ns_ attach-agent $node_(0) $tcp
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
## FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 10.0 "$ftp start"

#===================================================================
# Simulation Control
#===================================================================
for {set i 0} {$i < $val(nn) } {incr i} {
$ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0001 "stop"
$ns_ at 150.0002 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
      global ns_ tracefd
      global ns_ namtrace
      $ns_ flush-trace
      close $tracefd
      close $namtrace
      exit 0
}
```

```
puts "Starting Simulation..."
$ns_ run
```

Anexo 2 – Instalação do Simulador

Enumeram-se de seguida os passos necessários para instalar o simulador em ambiente Linux:

1. Realizar o download do simulador (ficheiro “ns-allinone-2.xx.tar.gz”) a partir do endereço: http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34/

2. Descompactar o ficheiro para uma pasta;

3. Executar o script “install” no interior da pasta ns-2.xx;

4. Validar a instalação executando o script “validate” no interior da pasta ns-2.xx.

Anexo 3 – Integração do código do simulador no ambiente de desenvolvimento ECLIPSE

Enumeram-se de seguida os passos necessários para constituir um projecto no ambiente de desenvolvimento ECLIPSE que contém o código fonte do simulador:

1. Instalar o ambiente de desenvolvimento ECLIPSE com o pacote CDT;
2. Instalar o simulador ns2 (ver anexo anterior);
3. Definir o caminho raiz do ns2 (pasta ns-2.xx) como o "workspace path" do ECLIPSE quando este é lançado;
4. No ECLIPSE deve utilizar-se a perspectiva C/C++;
5. Criar um novo projecto no eclipse para o ns2: "New" - -> "C++ Project";
6. Selecionar "Makefile project" - -> "Empty Project", atribuindo o nome da mesma subdirectoria "ns-2.xx" ao nome do projecto;
7. Executar o comando "Project" --> "Build All";
8. A partir de agora pode utilizar-se o ambiente ECLIPSE para realizar código, compilar e realizar *debug* nos diversos módulos que constituem o simulador.