

# **NETWORK SIMULATOR**

## **Guia Básico para Iniciantes**



Por

Mauro Margalho Coutinho

Doutor em Engenharia Elétrica – UFPa

2007

Network Simulator.  
Guia Básico para Iniciantes

<b>1</b>	<b>Introdução .....</b>	<b>3</b>
<b>2</b>	<b>A Ferramenta NS .....</b>	<b>6</b>
2.1	Informações Genéricas.....	6
2.2	Fazendo o Download .....	7
2.3	Procedimentos de Instalação e Configuração .....	7
2.4	Entendendo o Princípio de Funcionamento do NS .....	11
2.4.1	Planejando a Simulação.....	11
2.4.2	Definindo os Nós .....	14
2.4.3	Definindo os Enlaces .....	15
2.4.4	Definindo o Tráfego .....	16
2.4.5	Visualizando a Simulação com o NAM .....	19
2.4.6	Analisando o trace e gerando os gráficos .....	19
<b>3</b>	<b>Estudo de caso baseado em Simulação .....</b>	<b>27</b>
3.1	Redes WAN com QoS .....	27
3.2	Redes Sem Fio/NOMADIC/MANETs (Wireless 802.11)	
	41	
<b>4</b>	<b>Referência Bibliográfica Básica.....</b>	<b>49</b>
	<b>ANEXO IV.....</b>	<b>50</b>

# 1 Introdução

O objetivo deste guia é orientar, de forma prática e rápida, os candidatos a usuários do NS (Network Simulator), um dos simuladores de rede de computadores mais utilizados atualmente. Trata-se, portanto, de uma fonte de referência básica àqueles estudantes de graduação ou pós-graduação que encontraram alguma dificuldade quer na instalação do software, quer no planejamento da simulação.

Alguns conceitos de avaliação de desempenho serão abordados de forma superficial, uma vez que a aplicação desses conceitos torna os resultados obtidos bem mais consistentes.

As dez principais etapas de um processo de avaliação de desempenho (Declarar os objetivos e definir o sistema, listar serviços e saídas, selecionar métricas, listar os parâmetros, selecionar os fatores, selecionar a técnica de avaliação, selecionar a carga, projetar os experimentos, analisar e interpretar os resultados e apresentar os resultados e conclusões) serão apresentadas com um enfoque prático, embora a compreensão da teoria seja de fundamental importância. Para consolidar esses conceitos, recomenda-se a leitura dos livros citados na referência bibliográfica.

Para situá-lo melhor no contexto de avaliação de desempenho, via simulação, apresentamos a seguir um resumo extraído do artigo [9] de autoria do professor Dr. Carlos Renato Francês.

Existem diferentes técnicas para avaliação de desempenho em sistemas computacionais. Mas, basicamente, elas podem ser agrupadas em experimentação ou aferição e modelagem.

A técnica de experimentação é indicada somente quando o sistema a ser avaliado já existe. Caso contrário, deve-se optar pela construção de um protótipo, que é uma simplificação do sistema original, com o qual serão realizados os experimentos. As medições devem ser feitas pelo uso de *benchmarks*, que são padrões de desempenho, que avaliam o comportamento do sistema a partir de dados coletados.

A aquisição dos dados tende a produzir melhores resultados se envolver os dados reais do sistema. Normalmente, a coleta de dados durante a operação do sistema não é uma tarefa trivial uma vez que deve ser feita sem que nele haja interferência, pois isso afetaria os resultados.

Já a técnica de modelagem requer uma abstração a partir das características essenciais de um sistema real e é representada de acordo com um modelo escrito com um método formal. Esses métodos envolvem, normalmente, equações matemáticas e uma representação gráfica. Três formas de modelagem são redes de filas, redes de Petri e Statecharts. Um modelo de redes de filas oferece diferentes entidades para representar os recursos, os centros de serviços e os clientes.

Sistemas computacionais normalmente podem ser representados por redes de filas (enquanto um processo está usando um recurso, o outro interessado nesse recurso normalmente espera em uma fila). Redes de Petri permitem uma representação matemática do sistema. Sua flexibilidade permite a representação de sistemas paralelos, concorrentes, assíncronos e não determinísticos. As redes de Petri têm dois elementos básicos: um elemento ativo (transition), representado por barras, e um elemento passivo (place), representado por círculos. Places são equivalentes a estados do sistema e transition são as ações executadas pelo sistema. Esses dois elementos são conectados por meio de arcos. As redes de Petri são especialmente usadas no estudo de protocolos de redes. Statechart é uma extensão de máquina de

estados finitos e permite uma representação hierárquica com concorrência e comunicação entre os estados de um sistema. É particularmente indicado quando o sistema é reativo, ou seja, quando há uma reação a eventos internos e externos, normalmente sob condições críticas de tempo.

Tendo escolhido a técnica de modelagem a ser usada, faz-se necessário decidir de que maneira o modelo será resolvido. Há duas possibilidades: solução analítica e simulação. O método analítico produz um resultado mais rápido, mas nem sempre é aplicável porque, para se encontrar uma solução analítica são necessárias muitas simplificações do modelo. Essas simplificações impõem restrições que podem distorcer a realidade do sistema. Simulação, por outro lado, permite a inclusão de muitos detalhes do sistema dentro do modelo sem a imposição de restrições. O problema aqui é com o tempo requerido pelo programa que simula o processo.

Esse tempo normalmente cresce com a complexidade do modelo. Para resolver esse problema pode-se usar ambientes da computação paralela e distribuída.

Dependendo do domínio da aplicação, algumas técnicas são mais apropriadas que outras. Por isso é imprescindível o conhecimento das técnicas e do sistema em estudo para se decidir qual usar.

## 2 A Ferramenta NS

### 2.1 Informações Genéricas

O NS (Network Simulator) é um simulador de eventos discreto resultante de um projeto conhecido como VINT (Virtual InterNetwork Testbed). Dentre outros, compõem esse projeto a DARPA, USC/ISI, Xerox PARC, LBNL, e a universidade de Berkeley. Uma grande vantagem do NS reside no fato de ele ser livre, ou seja, totalmente gratuito e com código fonte aberto, o que permite ao usuário proceder os ajustes que julgar necessários. O simulador oferece suporte à simulação de um grande número de tecnologias de rede (com e sem fio), diferentes cenários baseados nos protocolos TCP e UDP, diversos escalonadores e políticas de fila, caracterização de tráfego com diversas distribuições estatísticas e muito mais.

A programação do NS é feita em duas linguagens: C++ para a estrutura básica (protocolos, agentes, etc) e OTCL (Object-oriented Tool Command Language) para uso como *frontend*. OTCL é uma linguagem interpretada, desenvolvida pelo MIT (Massachusetts Institute of Technology). Nela serão efetivamente escritos os códigos das simulações. O motivo para se utilizar duas linguagens de programação baseia-se em duas diferentes necessidades. De um lado existe a necessidade de uma linguagem mais robusta para a manipulação de bytes, pacotes e para implementar algoritmos que rodem um grande conjunto de dados. Nesse contexto C++, que é uma linguagem compilada e de uso tradicional, mostrou-se a ferramenta mais eficaz. De outro lado é fato que, durante o processo de simulação, ajustes são necessários com certa frequência. Muda-se o tamanho do enlace e faz-se um teste, muda-se o atraso e faz-se um teste, acrescenta-se um nó e faz-se um teste. Enfim, haveria um desgaste muito grande se, a cada mudança de parâmetro, e elas são muitas em uma simulação, houvesse a necessidade de se compilar o programa para testá-lo. O uso da linguagem OTCL, que é interpretada, evita esse

desgaste por parte do usuário, pois há uma simplificação no processo interativo de mudar e re-executar o modelo.

## 2.2 Fazendo o Download

O *website* oficial do NS é <http://www.isi.edu/nsnam/ns/>. Existem várias formas para se fazer o *download*. Sugere-se, principalmente para os iniciantes, que o formato chamado *allinone* (tudo em um) seja utilizado. Nesse formato todos os pacotes, sejam eles opcionais ou não, são baixados em um único arquivo com cerca de 50 megabytes. Isso realmente facilita a instalação apesar de requerer um espaço maior em disco (cerca de 250 megabytes). Os módulos opcionais do NS como o NAM, que é um visualizador gráfico das simulações e o Xgraph, que permite a criação de gráficos não são fundamentais, embora sejam fortemente recomendados aos iniciantes. Existem versões do NS para diversos sistemas operacionais dentre os quais FreeBSD, Linux, SunOS, Solaris e para a família windows (95,98,2000,NT e XP). Os arquivos disponíveis para download nas plataformas padrão “X”(linuX, Unix, aiX, etc) estão no formato “*.tar.gz*”.

Exceto quando for explicitamente citado de forma diferente, todas as referências adiante utilizadas devem ser baseadas na premissa de que o sistema operacional utilizado pelo usuário é o Linux. Posto isso, o procedimento inicial é descompactar o arquivo obtido. Isso é feito através do comando:

```
[ns]$ tar -zxvf <nome-do-arquivo.tar.gz>
```

Ex.: [ns]\$ tar -zxvf ns-allinone-2.26.tar.gz

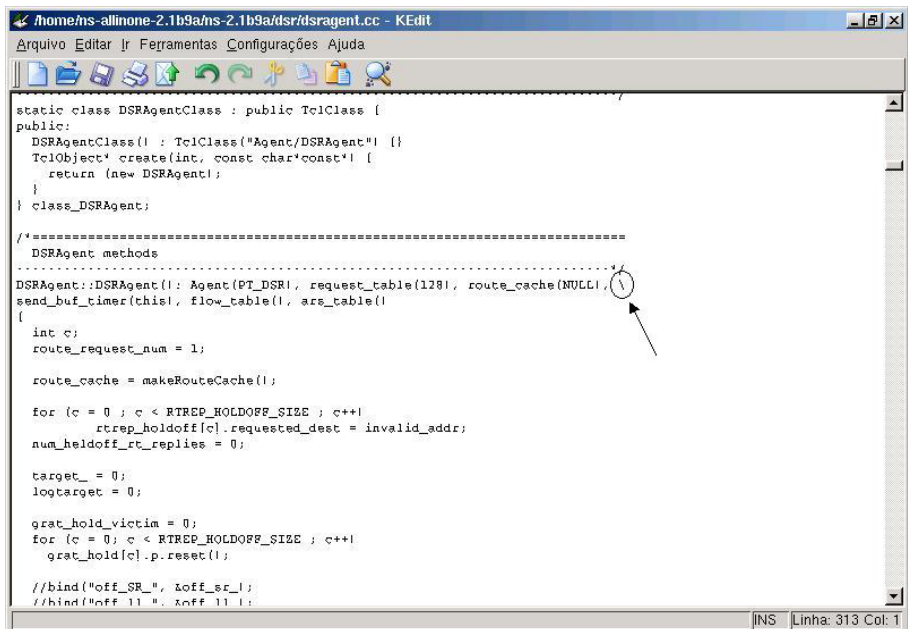
## 2.3 Procedimentos de Instalação e Configuração

- ¶ Sugere-se, principalmente aos iniciantes, que a instalação do Sistema Operacional Linux seja feita com a opção “Instalar Todos os Pacotes”. Isso é requerido em função do uso de algumas bibliotecas no processo de compilação dos módulos do software. Para efeito

didático considera-se que o NS será instalado dentro do diretório */home* em sistema operacional Linux Conectiva versão 8.

Muitos usuário acabam ficando frustrados e desistem logo nesta fase. Realmente existem algumas situações não documentadas que podem causar irritação. Um bom exemplo é o da distribuição Conectiva. Se o sistema operacional a ser utilizado for o Conectiva 8 e a versão do NS for a 2.1b9a, há necessidade de um pequeno ajuste no código antes de ele ser instalado. O usuário deve editar o arquivo *dsragent.cc*, como é mostrado na Figura 1, localizado no diretório */home/ns-allinone-2.1b9a/ns-2.1b9a/dsr/* e modificar a linha de número 313. Ao final da linha existe uma “\” que deve ser removida. O código da linha 314 deve ser concatenado ao final da linha 313 pois essa quebra de linha, indicada com uma barra, não é reconhecida pelo Conectiva 8. Caso o Sistema Operacional seja o RedHat 7.1 ou superior ou a versão do NS seja a 2.26, o procedimento acima descrito não é necessário.





**Figura 1 – Arquivo dsragent.cc**

Procedidos os ajustes necessários, o passo seguinte é instalar o NS. Para isso basta entrar no diretório `/home/ns-allinone-2.26/` e digitar `./install`. Um arquivo de lote (*batch*) fará todo o resto. O tempo requerido para a instalação varia de acordo com o equipamento que estiver sendo utilizado, mas uma coisa é certa, haverá tempo de sobra para um cafezinho. Portanto relaxe e espere.

O processo de instalação basicamente compila os arquivos da estrutura central do NS, que são escritos em C++, gerando um binário capaz de executar as simulações escritas em TCL.

Ao final da instalação será mostrado um caminho que deve ser copiado da interface textual e adicionado ao seu arquivo de inicialização, como mostra a Figura 2.

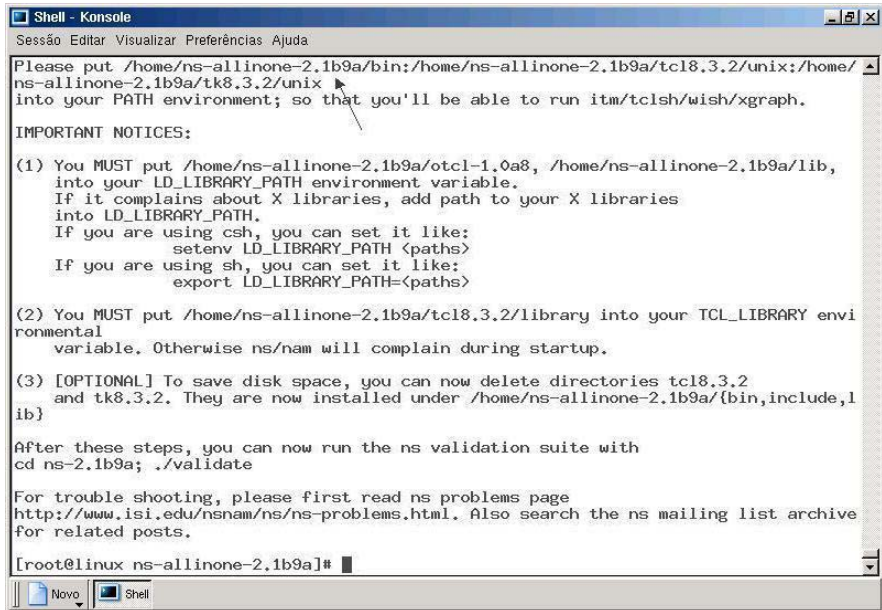


Figura 2 – Path na Instalação do NS

¶ O arquivo de inicialização do usuário chama-se *bashrc*. Todavia, se esse arquivo for utilizado, apenas o usuário poderá executar o NS. Caso o ajuste seja feito no arquivo *profile*, dentro do diretório */etc* todos os usuários do computador poderão executar o NS. Qualquer editor de textos (kate, vi, mcedit, etc) poderá ser usado nesse processo.

Ex. do ajuste no arquivo *profile* do Conectiva 8:

```

...
PATH="$PATH:/usr/bin/X11:/usr/games:/usr/local/bin"
PATH="$PATH:/home/ns-allinnone-2.26/bin:/home/ns-allinnone-
2.26/tcl8.3.2/unix:/home/ns-allinnone-2.26/tk8.3.2/unix"
MANPATH="$MANPATH:/usr/local/man"
...

```

## 2.4 Entendendo o Princípio de Funcionamento do NS

Basicamente uma simulação com o NS consiste em 5 passos:

- Planejar a simulação
- Definir os nós
- Definir a ligação entre os nós (topologia)
- Definir o tráfego que será injetado na rede
- Analisar os resultados

Para se escrever a simulação qualquer editor de textos pode ser utilizado, desde os baseados em texto como o *emacs*, *vi* ou o *mcedit* até os editores gráficos como o *kedit* ou o *kate* que já vêm com a interface gráfica kde. Os arquivos devem ser gravados com a extensão “*.tcl*”. Para se executar a simulação basta que se digite

```
[ns]$ ns <nome-do-arquivo>.tcl
```

Ex.: [ns]\$ ns exemplo1.tcl

### 2.4.1 Planejando a Simulação

Antes de efetivamente começar a programar é importante um planejamento de prancheta. Esse esboço do que se quer, em uma folha de papel, ajuda o usuário a ter uma visão macro das simulações pretendidas.

Um bom modelo é apresentado na Figura 3. Toda a estrutura da simulação, desde os agentes de transporte, passando pelas aplicações e todo o mapeamento físico da topologia está fielmente representada.

A noção de tempo no NS é obtida através de unidades de simulação que podem ser associadas, para efeitos didáticos, a segundos. No planejamento mostrado na Figura 3, a aplicação de vídeo contínuo ou *cbr*(*Constant Bit Rate*) é iniciada no

momento 0.1 e encerrada no momento 4.5, enquanto que a aplicação de transferência de arquivo ou *ftp* (*File Transfer Protocol*) é iniciada no momento 1.0 e encerrada no momento 4.0. No intervalo entre os momentos 1.0 e 4.0 ambas as aplicações estão sendo transmitidas e é nesse intervalo onde, provavelmente, os congestionamentos irão surgir. No NS os agentes precisam de um repositório que receberá seus pacotes. No caso do agente *tcp* (*transmission control protocol*) esse repositório chama-se *sink* (tanque) e tem a incumbência de gerar os pacotes de reconhecimento (*ACK - Acknowledge*). No caso do agente *udp* (*user datagram protocol*) o repositório chama-se *null* (nulo).

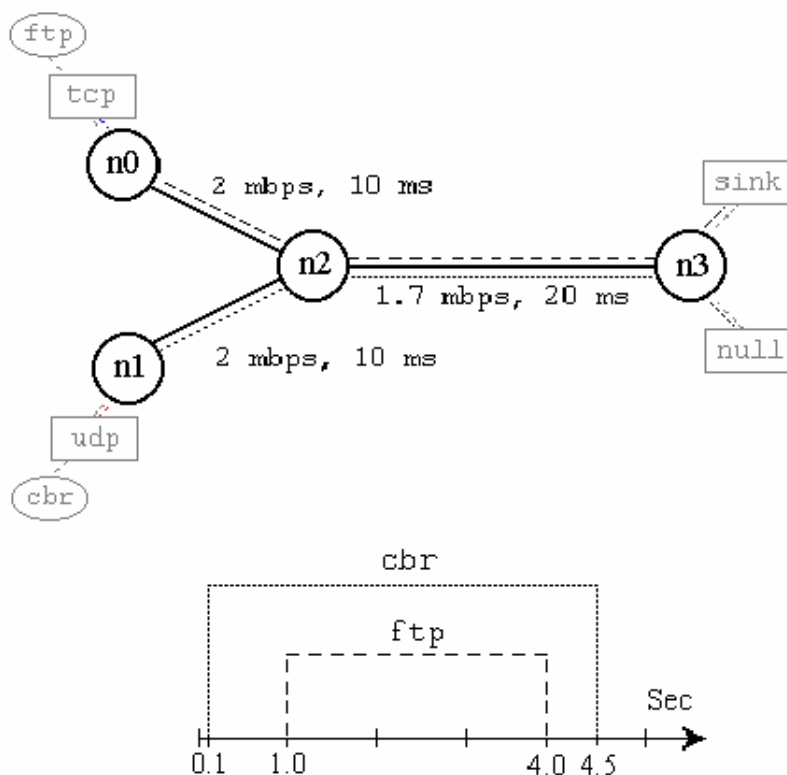
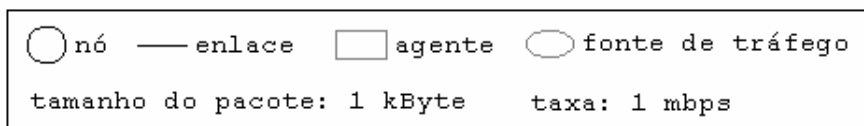
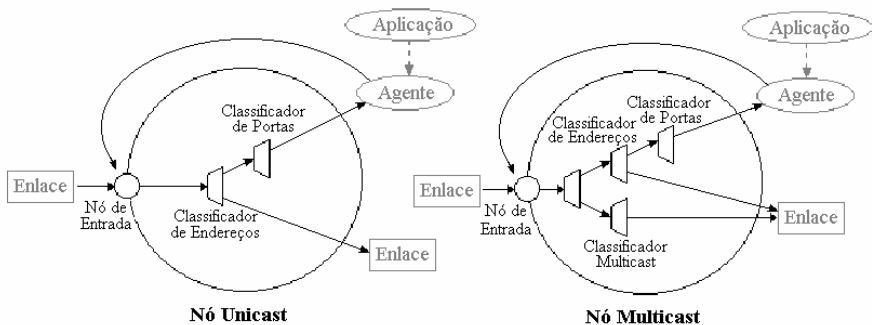


Figura 3 – Planejamento da Simulação

## 2.4.2 Definindo os Nós



A estrutura básica de um componente do tipo nó é mostrada na Figura 4.

**Figura 4 – Estrutura dos Nós**

Para se definir um Nó deve-se utilizar o formato

```
set <nome do Nó> [$ns node]
```

No exemplo abaixo, define-se um roteador (Nó) localizado em Belém. A linha seguinte serve para colocar um rótulo que será exibido durante a animação apresentada no NAM (Network Animator), se houver.

Comentários podem ser inseridos colocando-se o caracter “#” na primeira coluna.

```
# Definição do Roteador Belém
set rt_belem [$ns node]
$ns at 0.0 "$rt_belem label RoteadorBelem"
```

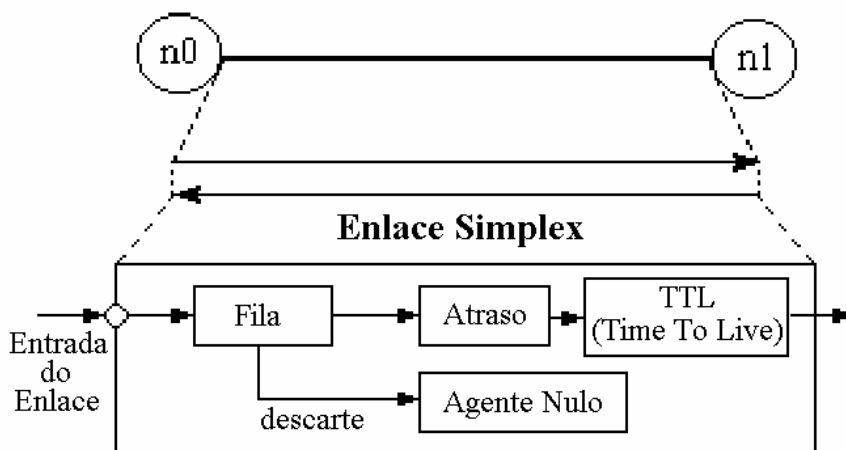
Para uma grande quantidade de Nós um laço pode ser utilizado como mostrado no código abaixo:

```
set NumerodeNos      60; # Número de Nós
...
for {set j 1} {$j<=$ NumerodeNos } { incr j } {
    set roteador($j) [$ns node]
```

```
}
```

### 2.4.3 Definindo os Enlaces

Um enlace ou *link* é uma estrutura que conecta os Nós dando forma à topologia. A estrutura desse componente pode ser observada na Figura 5.



**Figura 5 – Estrutura dos Enlaces.**

Uma notação importante no NS é a associação das políticas de fila na própria estrutura do enlace. Dessa forma, o atraso no encaminhamento dos pacotes será uma composição do atraso proveniente da fila, que depende do grau de congestionamento da rede, mais o atraso do próprio enlace, que é fixo e definido, em milissegundos, pelo usuário.

O formato de código requerido para a especificação do enlace tem o padrão descrito adiante. *Simplex* e *duplex* dizem respeito à capacidade do *link* em carregar dados apenas em um sentido ou em ambos, simultaneamente:

```
$ns duplex-link[1] $rt_belem[2] $rt_rio[3] 1Mb[4] 10ms[5] DropTail[6]
```

```
[1] - Tipo do Link (Simplex ou Duplex)
[2] - Nó origem
[3] - Nó destino
[4] - Velocidade do Nó em Megabits por segundo
[5] - Atraso do link em milissegundos
[6] - Política de Fila - DropTail = FIFO = 1º que entra é o 1º que sai
```

É importante planejar a topologia e o tráfego de forma que ocorram gargalos na rede. Só assim, problemas como descarte de pacotes poderão aparecer e ser avaliados.

#### 2.4.4 Definindo o Tráfego

Para se definir o tráfego no NS dois itens são requeridos:

**a) O agente ou protocolo de transporte que irá conduzir os pacotes.**

Os mais utilizados são o tcp (transmission control protocol) e o udp (user datagram protocol), embora o NS também ofereça suporte a outros agentes como o rtp (real time protocol).

**b) O tipo de aplicação que fará uso desse transporte.**

Vídeo contínuo pode ser caracterizado por aplicações do tipo CBR (Constant Bit Rate). Também existe suporte para aplicações FTP (File Transfer Protocol) que caracterizam os downloads e HTTP (Hyper Text Transfer Protocol). Além disso, existem distribuições estatísticas que podem ser usadas para caracterizar outros tráfegos de entrada como por exemplo a exponencial e a de pareto (para rajadas).

Dois exemplos são apresentados abaixo. O primeiro referente a uma transmissão de aplicação FTP via TCP e o segundo referente a uma transmissão CBR via UDP (ver Códigos 1 e 2).



```

$ns color 7[1] orange[2]
...
set tcpl[3] [new Agent/TCP/Newreno][4]
$tcpl[3] set class_ 7[1]
$tcpl[3] set fid_ 1[5]
$tcpl[3] set windows_ 2000[6]
set sink1[7] [new Agent/TCPSink][8]
$ns attach-agent $transmissor1[9] $tcpl[3]
$ns attach-agent $receptor1[10] $sink1[7]
$ns connect $tcpl[3] $sink1[7]
set ftp1[11] [$tcpl[3] attach-source FTP][12]

```

- [1] - Um número que será associado a uma cor para uso no NAM
- [2] - Cor escolhida para se exibir o fluxo. Orange = Laranja
- [3] - Mnemônico atribuído à instância do protocolo de transporte
- [4] - Tipo do Protocolo ou agente de transporte
- [5] - Número que identificará o fluxo. fid = Flow Identification
- [6] - Tamanho da janela de transmissão do protocolo TCP
- [7] - Mnemônico atribuído à instância do agente que receberá os dados
- [8] - Tipo do agente que receberá os dados (SINK)
- [9] - Nó onde será conectado o fluxo para efeito de transmissão
- [10] - Nó onde será conectado o tanque de recepção do fluxo
- [11] - Mnemônico atribuído à instância da aplicação
- [12] - Tipo da Aplicação

### Código 1– Transmissão de Aplicação FTP via TCP

```

$ns color 1 blue
...
set udpl [new Agent/UDP]
$ns attach-agent $transmissor2 $udpl
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udpl
$udpl set packetSize_ 1000[1]

$udpl set class_ 1
$cbr1 set rate_ 4000000[2]

set null1 [new Agent/Null]
$ns attach-agent $receptor2 $null1
$ns connect $udpl $null1

```

- [1] - Tamanho dos pacotes em bytes
- [2] - Taxa de transmissão. No exemplo a taxa é de 4Mbps

### Código 2– Transmissão de Aplicação CBR via UDP

Em algumas situações, pode-se desejar obter todos os dados (nós, links e tráfego) de um arquivo de lote. Isso também é possível. O exemplo mostrado no Anexo I

apresenta um programa completo em TCL que obtém todas as suas configurações de três arquivos de lotes chamados *nos.txt*, *ramos.txt* e *trafego.txt*. Estruturas de programação freqüentemente utilizadas nas simulações do NS são mostradas no exemplo, dentre elas:

- Estruturas de repetição e decisão (*while/if*)
- Especificação do tempo de simulação
- Especificação do tempo de disparo de cada fluxo (*start*)
- Construção de *procedures*

Ao final são exibidos tanto a animação da simulação, através do NAM, como os gráficos de vazão, através do Xgraph.

Os arquivos de lote apresentam a seguinte estrutura:

***NOS.TXT (Nome do Nó em cada linha)***

```
belem  
brasil  
saopaulo  
riodejaneiro  
recife  
belohorizonte  
portoalegre
```

***RAMOS.TXT (Nó origem, Nó destino, largura de banda do enlace e atraso)***

```
belem recife 2 20  
recife saopaulo 2 20  
belohorizonte portoalegre 2 20  
brasília saopaulo 2 20  
saopaulo belohorizonte 2 20  
portoalegre riodejaneiro 2 20
```

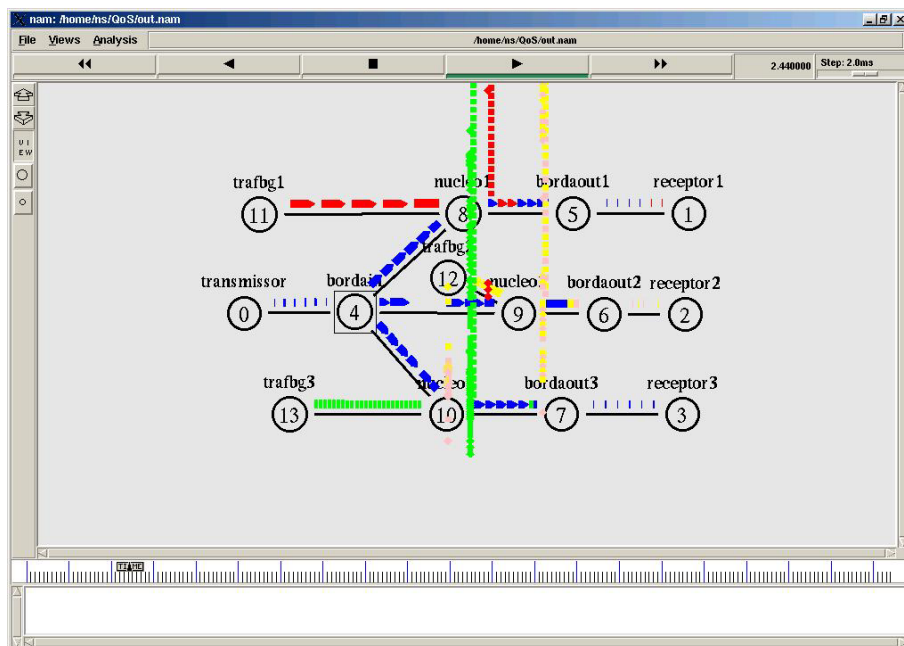
***TRAFEGO.TXT (Nó de onde parte o fluxo, Nó destino e o tipo de aplicação)***

```
belem riodejaneiro cbr  
brasil riodejaneiro cbr  
saopaulo belohorizonte cbr  
recife belohorizonte cbr
```

- Os atributos devem ser separados por um espaço em branco.

### 2.4.5 Visualizando a Simulação com o NAM

A ferramenta NAM (Network Animator) é muito importante para se ter uma idéia gráfica, baseada em animações, do andamento da simulação. Com o NAM pode-se observar a transmissão dos fluxos, a formação de filas, o descarte de pacotes, etc. (ver exemplo Figura 6). A execução do NAM requer a instalação de uma interface gráfica no ambiente linux.



**Figura 6 – NAM (Network Animator)**

### 2.4.6 Analisando o trace e gerando os gráficos

Concluída a simulação, inicia-se uma das fases mais importantes: a análise dos resultados. Afinal, estes serão utilizados na elaboração de gráficos que servirão de suporte a trabalhos a serem submetidos a eventos científicos. É fundamental que se busque consistência e coerência nesses resultados antes de apresentá-los. O NS

gera o **log** de todos os eventos ocorridos durante o processo de simulação em um arquivo de texto chamado **trace file**. O **trace** pode chegar a vários Megabytes de tamanho (Ex. 400 MB) e exige cuidado em sua análise. O formato padrão do arquivo **trace** é mostrado na Figura 7.

O primeiro campo diz respeito ao evento ocorrido. Pode ser uma entrada em fila (+), uma saída de fila (-), um descarte de pacote (d), um recebimento de pacote (r), etc. O campo seguinte é o momento da simulação onde o evento ocorreu. Os dois campos seguintes são referentes ao intervalo de Nós onde o evento ocorreu. Os dois próximos campos dizem respeito ao tipo (tcp, udp, etc) e tamanho do pacote (em bytes), respectivamente. Uma série de *flags* relacionados a notificação antecipada de congestionamento (ENC) vêm a seguir, mas normalmente não são utilizados. Depois, tem-se a identificação do fluxo, os endereços do transmissor e do destinatário, o número de sequência do pacote e, finalmente, um número que identifica de forma única o pacote na rede.

evento	tempo	nó origem	destino	tipo pacote	tamanho pacote	flags	id fluxo	endereço fonte	endereço destino	num seq	id pacote
--------	-------	--------------	---------	----------------	-------------------	-------	-------------	-------------------	---------------------	------------	--------------

```

r : pacote recebido      (no destino)
+ : entrada de pacote   (na fila)      endereço fonte   : nó.porta(3.0)
- : saída de pacote     (da fila)      endereço destino : nó.porta(0.0)
d : pacote descartado   (da fila)

```

```

r 1.3556 3 2 ack 40 ----- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207

```

**Figura 7 – Formato do arquivo de Trace**

Um exemplo de código que analisa o arquivo de *Trace* e calcula os atrasos é mostrado no Anexo II.

Existem algumas ferramentas que podem ser obtidas gratuitamente na Internet para análise do *trace file*. Um exemplo de uma dessas ferramentas é o Tracegraph <http://www.geocities.com/tracegraph/>. Entretanto, para se usar o Tracegraph existe a exigência do software Matlab (que não é gratuito) para versão windows e de algumas bibliotecas do Matlab para a versão Linux, mas estas são disponibilizadas juntamente com o Tracegraph. Outra possibilidade é a construção, já no código tcl, de um trace personalizado. Um exemplo, que é apresentado no tutorial de Mark Greis <http://www.isi.edu/nsnam/ns/tutorial/index.html>, grava duas colunas (tempo de simulação e vazão) em intervalos predefinidos, em um arquivo de *trace* que posteriormente será usado como parâmetro de entrada no utilitário XGraph. O resultado é um gráfico com a evolução da vazão de cada um dos fluxos utilizados na simulação conforme mostrado na Figura 8.

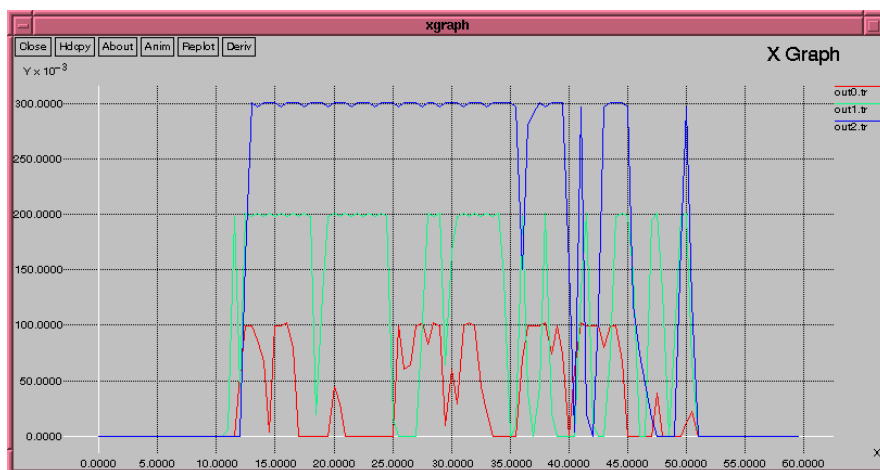


Figura 8 – Vazão

Para obter os arquivos *out1.tr*, *out2.tr* e *out3.tr* observados no gráfico, três arquivos de *trace* foram criados no início do *script* de simulação, um para cada fluxo.

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

Um procedimento que é chamado recursivamente grava, a cada 0.5 unidades de simulação, os dados em cada um dos arquivos de *trace* (ver Código 3).

```
proc record {} {
    global sink0 sink1 sink2 f0 f1 f2
    #Obtem uma instância do objeto Simulator
    set ns [Simulator instance]
    #Configura o intervalo de chamada recursiva
    set time 0.5
    #Calcula o número de bytes recebidos pelos Sink
    #      em determinado instante
    set bw0 [$sink0 set bytes_]
    set bw1 [$sink1 set bytes_]
    set bw2 [$sink2 set bytes_]
    #Obtém o tempo corrente
    set now [$ns now]
    #Calcula a largura de banda (em MBit/s) e as
    #      escreve nos arquivos
    puts $f0 "$now [expr $bw0/$time*8/1000000]"
    puts $f1 "$now [expr $bw1/$time*8/1000000]"
    puts $f2 "$now [expr $bw2/$time*8/1000000]"
    #Inicializa as variáveis nos Sinks
    $sink0 set bytes_ 0
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0
    #Chamada recursiva
    $ns at [expr $now+$time] "record"
}
```

**Código 3– Procedimento de Gravação do Trace**

Ao final, um procedimento faz uma chamada ao Xgraph fornecendo como parâmetros os arquivos gravados: *out0.tr*, *out1.tr* e *out2.tr* (ver Código 4).

```
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[1] &  
    exit 0  
}  
[1] - resolução de vídeo onde o gráfico será exibido. Ex. 640x480
```

#### Código 4 – Procedimento de Execução do Xgraph

Há casos, porém, que nenhum desses recursos resolve. Nesses casos das duas uma: ou o usuário escreve um programa para ler o trace e proceder os cálculos necessários ou ele importa o *trace* para um banco de dados, como o MS-Access® por exemplo, e aplica instruções SQL que facilmente podem filtrar o conteúdo do *trace*.

¶ Para os que pretendem utilizar esta última sugestão a dica é editar o trace pelo MS-Word® e gravá-lo novamente com a opção **Somente Texto com quebras de linha (\*.txt)**. Sem isso não será possível importá-lo em função de sua natureza linux.

Na verdade o uso dos resultados de forma bruta, como mostrado no procedimento anterior não é a forma mais correta se o objetivo for avaliar o desempenho do

sistema. É importante aplicar alguns conceitos para tornar os resultados mais confiáveis. Esses conceitos serão abordados de forma simplificada abaixo. Todavia, sugerimos ao leitor consultar uma referência especializada para um entendimento mais completo.

### **a) Warm-Up**

Warm-up é o tempo que deve ser aguardado antes de efetivamente se computar as medições que serão avaliadas, ou seja, é o tempo para o sistema entrar em estado estacionário. Isso é necessário porque, muitas vezes, o sistema só se estabiliza após esse período e as medições realizadas antes dele se completar podem deturpar os resultados. Um exemplo é o tempo necessário para que os roteadores atualizem as suas tabelas de roteamento. Nesse período a rede sofre um tráfego adicional que pode mascarar os resultados. Um tempo considerado razoável para essa espera é por volta de 10% do tempo de simulação. Se a simulação tem 100 segundos de duração, os resultados só devem ser computados após decorridos 10 segundos. Vale frisar que existe um cálculo apropriado para se obter esse tempo. Esse procedimento pode ser obtido em MacDougall[87].

### **b) Intervalo de Confiança**

Outro conceito extremamente importante chama-se intervalo de confiança e diz respeito à dispersão dos valores em torno da média. O intervalo de confiança é um valor que deve ser somado e subtraído a média. Qualquer resultado que estiver fora desse intervalo deve ser ignorado para efeito de avaliação de desempenho. Por exemplo, se a vazão média foi de 2Mbps e o intervalo de confiança for igual a 1, então só deverão ser consideradas para o resultado as vazões entre 1 e 3 Mbps. Se,



eventualmente, algum dos fluxos obteve vazão de 4Mbps ou 0,512 Kbps estes valores devem ser desconsiderados na apresentação dos resultados.

Para efeitos didáticos o procedimento de cálculo do intervalo de confiança será mostrado considerando-se que os dados obtidos foram importados para o MS-Excel®, onde certamente os gráficos poderão ser construídos com uma gama bem variada de opções.

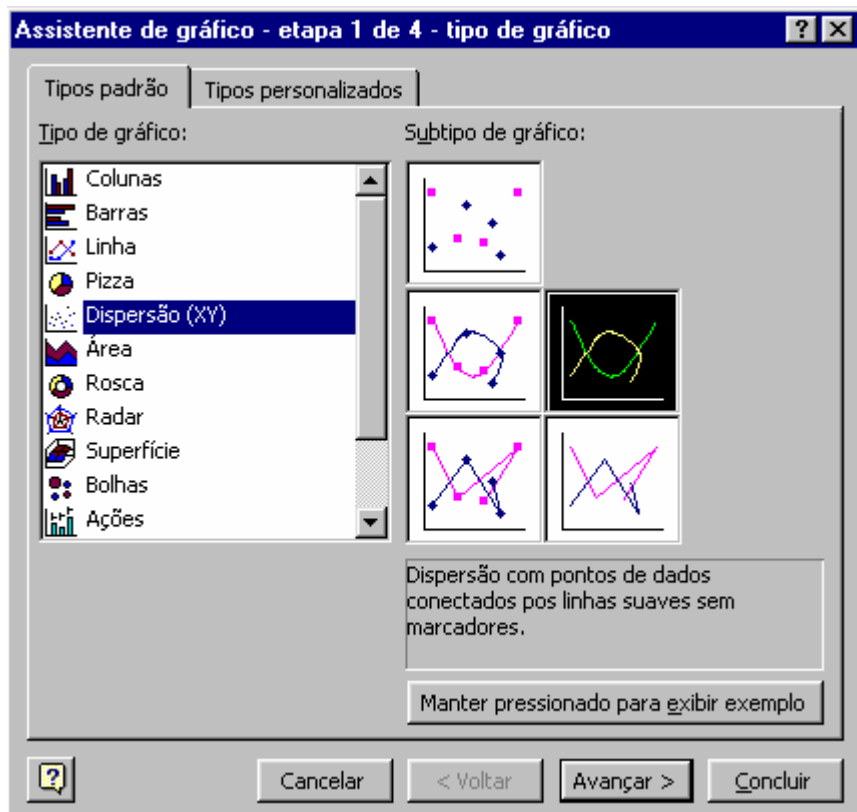
- O primeiro passo é calcular a média, o que é feito utilizando-se a formula `=MÉDIA(CélulaInicial:CélulaFinal)`.
- Feito isso deve-se calcular o desvio padrão. Utilize a fórmula `=DESVPAD(CélulaInicial:CélulaFinal)`.
- Outro parâmetro importante é o nível de significância. Um bom valor para este parâmetro é `0,05`.
- Finalmente para se calcular o intervalo de confiança utilize a fórmula:

`=INT.CONFIANÇA(nívelsignif.;Desvio Padrão;Nº de Itens)`

	A	B	C	D	E
1				<b>Vazão</b>	<b>Considerar?</b>
2				0,13115	=SE(E(C15<=D2;D2<=E15);"S";"N")
3				0,195538	N
4				0,20736	N
5				0,112699	S
6				0,126832	S
7				0,107389	S
8				0,047234	N
9			<b>Média:</b>	=MÉDIA(D2:D8)	
10			<b>Desvio Padrão:</b>	=DESVPAD(D2:D8)	
11			<b>Nível de Significância:</b>	0,05	
12			<b>Intervalo de Confiança:</b>	=INT.CONFIANÇA(D11,D10;7)	
13					
14			<b>Média - IC</b>	<b>Média</b>	<b>Média + IC</b>
15			=D9 - D12	=D9	=D9 + D12

**Tabela 1- Intervalo de Confiança**

Uma vez que nosso cálculo já foi feito no MS-Excel® pode-se aproveitar e gerar ali mesmo os gráficos.




- ¶ Para gráficos como atraso e vazão sugere-se o uso do modelo chamado dispersão(XY).

### 3 Estudo de caso baseado em Simulação

#### 3.1 Redes WAN com QoS

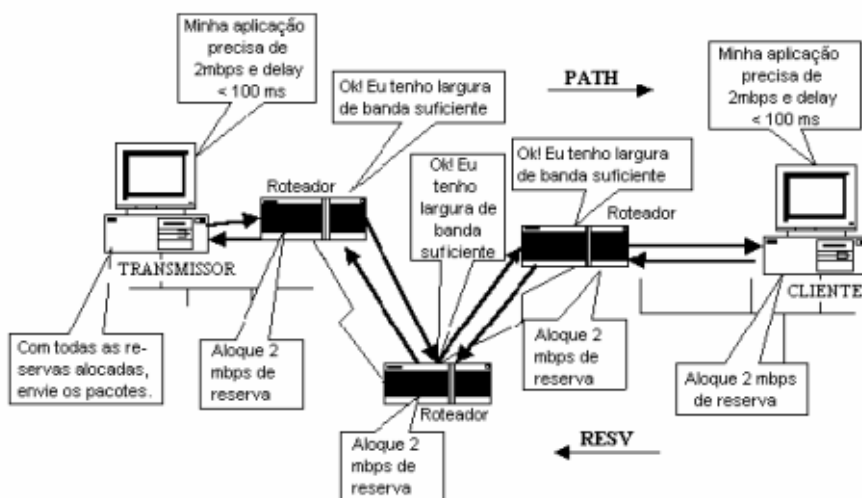
A partir de sua versão 2.1b8a, o NS passou a contar com um módulo chamado DiffServ (Differentiated Services) que permitiu simulações de QoS (Quality of Service) em redes como a Internet. QoS envolve uma série de propostas que buscam garantir a qualidade de serviços, como transmissão de vídeo por exemplo, mesmo em redes bastante congestionadas. Existem diversas técnicas para isso sendo que as mais conhecidas são mostradas de forma resumida a seguir:

**a) Técnica de Serviços Integrados ou IntServ (Integrated Services):** A proposta de serviços integrados baseia-se na premissa de que não há como se obter uma verdadeira garantia sem reserva de recursos. Isto não deixa de ser uma verdade.

 *Considere a existência de uma auto-estrada onde o congestionamento ocorre sistematicamente. A única forma de garantir que os transportes coletivos fluam rapidamente seria através de uma reserva de recursos, ou seja, a solução IntServ para este problema envolveria reservar uma ou mais pistas apenas para esse tipo de transporte. Sendo assim, por mais que ocorresse uma situação generalizada de congestionamento nas pistas adjacentes, o tráfego de ônibus continuaria a fluir sem problemas.*

Apesar de resolver o problema da reserva de recursos esta solução é inviável em redes como a Internet. Os motivos são basicamente dois. Em primeiro lugar tem-se o fator escalabilidade, ou seja, o crescimento. Torna-se inviável gerenciar, de maneira eficaz, milhares de solicitações de reservas. Um outro problema dessa técnica é o excesso de sinalização que ela produz, o que poderia sobrecarregar a rede.

O NS já oferece suporte à técnica de Serviços Integrados há algum tempo. Isso é feito através de um protocolo chamado RSVP (Resource Reservation Protocol) (ver Figura 9). Como já existe um certo consenso de que esta não é uma técnica indicada para resolver os problemas de QoS na Internet, ele não será foco de nosso estudo. Aos interessados, sugerimos uma pesquisa no arquivo *test-suite-intserv.txt*, que acompanha o NS e encontra-se no diretório ...ns-2-26/tcl/ex.



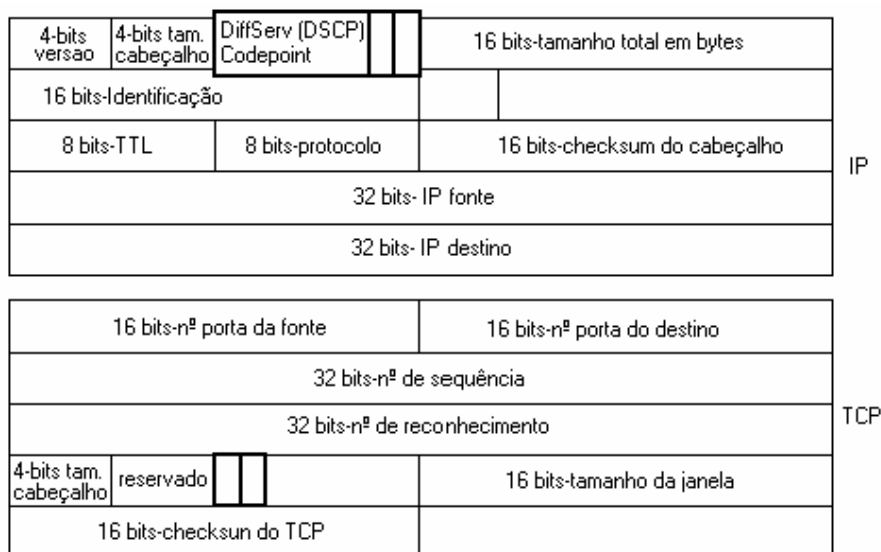
**Figura 9 – Reserva de recursos**

**b) MPLS (Multi Protocol Label Switing):** A proposta de prover QoS com MPLS está ligada particularmente à engenharia de tráfego. Basicamente o uso de MPLS implica na escolha do caminho a ser percorrido na transmissão de dados. Muitas vezes o critério padrão para a escolha da rota se baseia no número de saltos entre roteadores localizados entre o transmissor e o receptor. É provável que, em algumas situações, há mais vantagem em se enviar o dado por uma caminho mais longo, mas com largura de banda maior, por exemplo. Nessas situações MPLS pode configurar um proposta interessante. Aos interessados, sugerimos uma

pesquisa no arquivo *MPLS-sim-template.txt*, que acompanha o NS e encontra-se no diretório ...ns-2-26/tcl/ex.

**c) Técnica de Serviços Diferenciados ou DiffServ (*Differentiated Services*):** A proposta de Serviços Diferenciados teve uma aceitação tão boa que acabou por ser adotada na Internet2 através de uma projeto conhecido como QBONE (Um backbone com QoS). Muitos se perguntam o porquê de uma rede baseada em tecnologia ATM, com bastante largura de banda e velocidade altíssima, precisar de QoS. A resposta é bem simples. Por mais largura de banda que seja disponibilizada, sempre haverá necessidade de mais dela. Isso realmente ocorre. Hoje as reclamações giram em torno do serviço de multimídia na Internet, que praticamente é inviável para a maioria dos usuários. Amanhã, quando esse serviço for disponibilizado, a briga será por recursos que viabilizem realidade virtual, depois... quem sabe! O fato é que nunca haverá banda suficiente e isso é um indicativo da necessidade de se disponibilizar uma técnica de QoS.

Para entender os códigos exemplo de DiffServ no NS é preciso entender como a técnica funciona. O mecanismo DiffServ utiliza a marcação dos pacotes para priorizar o tráfego. Dentro dos roteadores um mecanismo chamado PHB (Per Hop Behavior) interpreta a marcação e encaminha o pacote para uma fila que flui mais ou menos rapidamente. Para a marcação dos pacotes (ver Figura 10) o chamado DS byte (byte de Serviços Diferenciados) é usado no cabeçalho de cada pacote IP. No IPv4 há um mapeamento do octeto Type of Service (ToS) e no IPv6 do Traffic Class (TC). Seis bits desse byte, chamados *Codepoint*, são combinados para definir o comportamento do pacote por salto ou PHB (Per Hop Behavior) que é analisado em cada roteador no despacho do pacote. Os outros dois bits foram preservados para uso em futuras propostas, são os chamados CU (*Current Unused*).



**Figura 10 – Campo DSCP dentro do Pacote IP**

Um modelo de referência chamado serviço olímpico é mostrado na tabela 1. Nessa proposta existem três prioridades de encaminhamento (ouro, prata e bronze) e três precedências de descarte (baixa, média e alta).

DESCARTE	CLASSE			
	Serviço Olímpico			Classe 4
	Classe 1/Ouro	Classe 2/Prata	Classe 3/Bronze	
Baixo	AF11 = 001010	AF21 = 010010	AF31 = 011010	AF41 = 100010
Médio	AF12 = 001100	AF22 = 010100	AF32 = 011100	AF42 = 100100
Alto	AF13 = 001110	AF23 = 010110	AF33 = 011110	AF43 = 100110

**Tabela 1 - Proposta de Codificação para PHB**

É importante que cada domínio só deixe entrar a quantidade de tráfego que ele consiga gerenciar para que o sistema não entre em colapso generalizado. Para isso toda a complexidade da técnica de DiffServ foi transferida para os nós de borda. Estes precisam implementar as chamadas políticas de admissão que poderão limitar as taxas de entrada de acordo com configurações preestabelecidas. Um fluxo que exceda a taxa acordada pode, por exemplo, ter o excedente descartado ou remarcado para uma prioridade menor ou mesmo autorizado a entrar sendo que o custo adicional será cobrado com os juros equivalentes.



*A analogia para se explicar o mecanismo de serviços diferenciados pode ser feita com um banco. Dentro do banco existem diversas filas. Clientes VIP têm acesso a filas menores com caixas mais experientes e que, por conseguinte, fluem mais rapidamente. De acordo com uma marcação, o cartão do banco, o cliente se dirige a uma fila mais ou menos rápida. Cabe ao banco, entretanto, adotar políticas para que as admissões de clientes não sejam excessivas, o que poderia congestionar todas as filas.*

Para entendermos melhor o script exemplo do NS para serviços diferenciados deve-se analisar a topologia a ser utilizada (ver Figura 11).

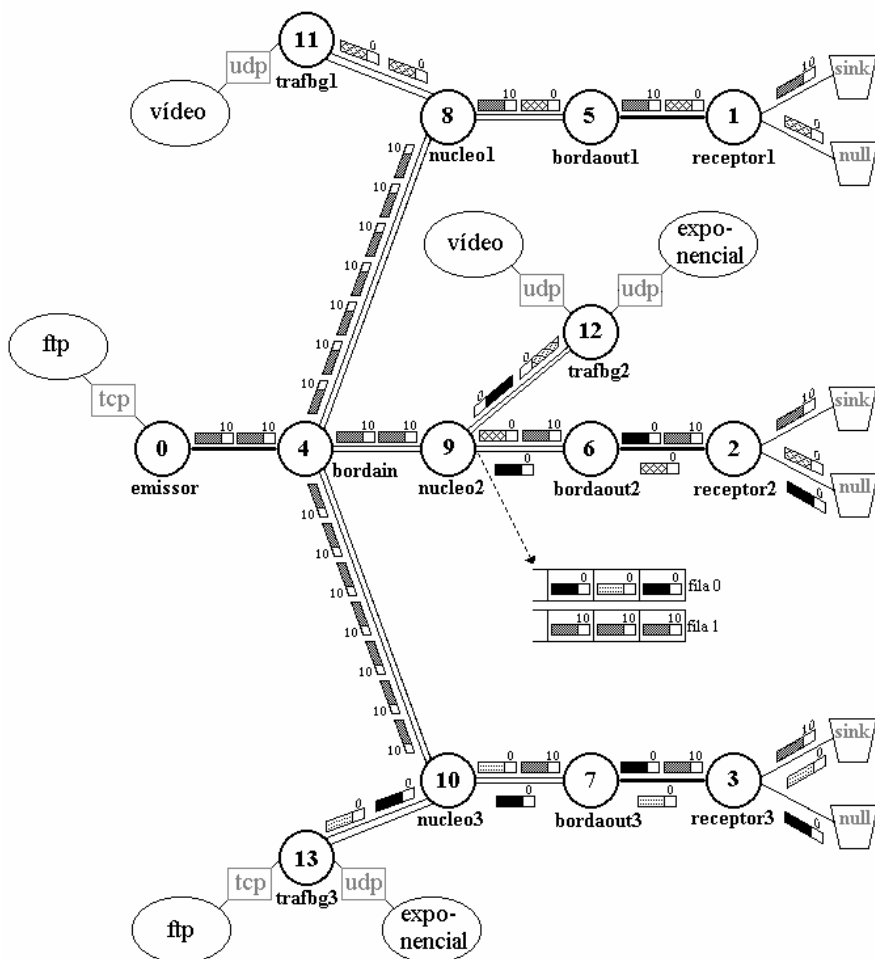


Figura 11 – Topologia da Simulação DiffServ

A proposta aqui é transmitir fluxos, com QoS, do nó *Transmissor* com destino aos nós *Receptores*. Serão aplicadas políticas de admissão nas bordas, especialmente na de entrada, e tráfego de fundo será injetado na rede para torná-la congestionada. Ao final o tráfego marcado deve ter prioridade sobre os demais, mesmo em situações de congestionamento.



Como a simulação também envolve tráfego baseado no protocolo TCP, que emite um sinal de reconhecimento *ACK* ao receber um pacote, haverá necessidade de se configurar uma política de admissão em ambas as bordas, tanto na de entrada como na de saída.

O primeiro passo consiste na criação dos Nós (ver exemplo do Código 5).

```
set transmissor [$ns node]
...
set trafbg3[1] [$ns node]

[1] - nome do nó
```

### Código 5 – Criação de Nós

Na segunda etapa deve-se conectar esses nós configurando assim a topologia da rede. Em situações normais as informações requeridas são basicamente o tipo do enlace (*simplex/duplex*), a largura de banda em Megabits por segundo, o atraso em milissegundos e a política de fila a ser utilizada (ver exemplo do Código 6).

```
$ns duplex-link[1] $emissor[2] $bordain[3] 100Mb[4] 5ms[5]
DropTail[6]

[1] - tipo do enlace
[2] - nó origem
[3] - nó destino
[4] - largura de banda em Mega bits por segundo
[5] - atraso em milissegundos
[6] - política de fila a ser adotada entre os nós. DropTail = FIFO
```

### Código 6 – Ligação dos Nós

Os enlaces que compõem o domínio Diffserv têm algumas características próprias. Uma delas é que sempre são implementados com *links* do tipo *simplex*. Havendo a necessidade de os dados trafegarem em ambos os sentidos será preciso a definição de dois desses *links*. Uma outra característica é o uso da política de fila *dsRED* (*Differentiated Service Random Early Discarded*) nesses enlaces. Também é necessário indicar a procedência do enlace, se da borda para o núcleo ou vice-versa. Isso é feito acrescentando-se a palavra **edge**, quando o *link* for da borda para o núcleo ou **core**, quando o *link* for do núcleo para a borda (ver exemplo do Código 7).

É importante frisar que os nós que não pertencem ao domínio DiffServ não precisam seguir essa configuração, embora possam estar conectados ao domínio sem problemas como é o caso do código anterior.

```
$ns simplex-link $bordain $nucleo1 10Mb 5ms dsRED/edge
$ns simplex-link[1] $nucleo1[2] $bordain[3] 10Mb[4] 5ms[5] dsRED[6]/
                                                    core[7]
```

[1] - tipo do enlace

[2] - nó origem

[3] - nó destino

[4] - largura de banda em Mega bits por segundo

[5] - atraso em milissegundos

[6] - política de fila a ser adotada entre os nós

[7] - direção do fluxo de dados (core ou edge)

### Código 7 – Links dentro do domínio DiffServ

Embora não tenha tanta importância em um documento científico, a visualização gráfica da simulação em forma de animação é fundamental para que se possa entender, acompanhar e fazer os ajustes necessários durante os testes. Por isso o uso do Network Animator (NAM) é fundamental nesse processo. O Código 8

indica quais as filas a serem monitoradas graficamente pelo NAM para que se possa constatar a priorização do tráfego que recebeu a marcação.

```
$ns simplex-link-op[1] $nucleo1[2] $bordaout1[3] queuePos 0.5[4]
```

[1] - tipo do link a ser monitorado

[2] - nó origem

[3] - nó destino

[4] - intervalo de tempo entre os monitoramentos

### Código 8 – Monitoramento das Filas

As políticas de admissão serão sempre aplicadas em uma das filas definidas nos nós de borda do domínio DiffServ. Já as políticas de encaminhamento (priorização) serão sempre aplicadas em uma fila do nó de núcleo do domínio DiffServ. Em ambos os casos a criação da fila é requerida (ver exemplo do Código 9).

```
set qBinN1[1] [[ $ns link $bordain[2] $nucleo1[3]] queue]
```

[1] - nome da fila.

Ex.: q (de queue); Bin (Borda de Entrada); N1 (Núcleo1)

[2] - nó origem

[3] - nó destino

### Código 9 – Criação das Filas

Após as criação das diversas filas, cabe especificar as políticas de admissão ou encaminhamento. A forma de segregação do tráfego é feita através da definição de filas de prioridade. São criadas filas físicas (*numQueues*), que representam os PHB's e, dentro delas, filas virtuais (*setNumPrec*). Quando um fluxo está fora do perfil pode-se, por exemplo, remarcá-lo e enviá-lo a uma fila virtual de menor prioridade dentro do mesmo PHB.

No exemplo mostrado no Código 10, foram criadas duas filas físicas e uma fila virtual dentro de cada uma delas. Uma das filas físicas acondicionará o tráfego a ser priorizado e a outra receberá o tráfego de melhor esforço.

```
$qBinN1 meanPktSize 1000[1]
$qBinN1 set numQueues_ 2[2]
$qBinN1 setNumPrec 1[3]
$qBinN1 addPolicyEntry
    [$transmissor[4] id] [$receptor1[5] id] TSW2CM[6] 10[7] 1000000[8]
$qBinN1 addPolicerEntry TSW2CM[9] 10[7] 10[10]
$qBinN1 configQ 0[11] 0[12] 20[13] 40[14] 0.02[15]
$qBinN1 addPHBEntry 10[16] 0[17] 0[18]
```

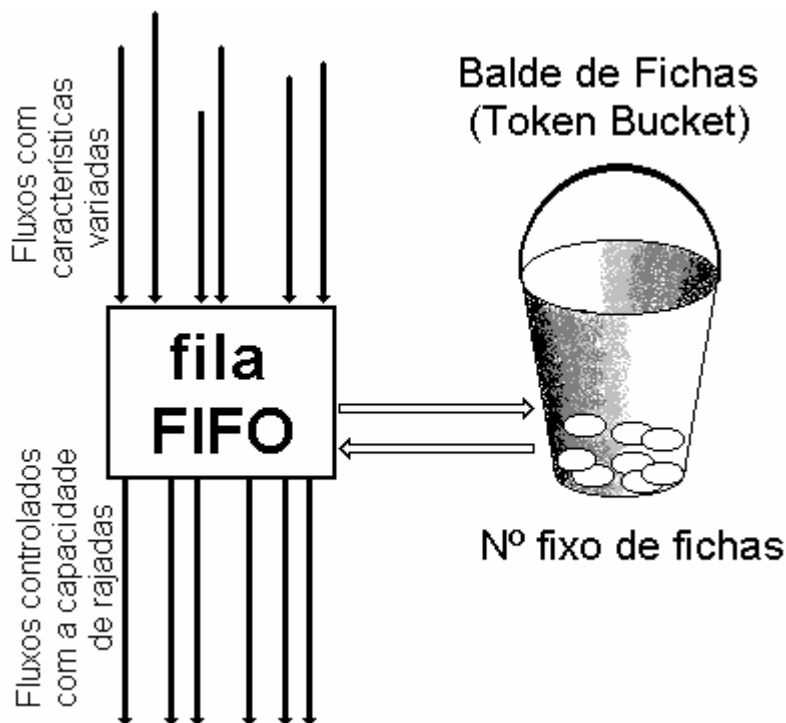
- [1] - tamanho do pacote em bytes.
- [2] - número de filas físicas
- [3] - número de filas virtuais
- [4] - nó origem da transmissão
- [5] - nó destino da transmissão
- [6] - política de admissão/classificação a ser adotada
- [7] - Code Point inicial
- [8] - CIR (Committed Information Rate) Taxa de Entrada
- [9] - mesma política adotada em [6]
- [10]- Code Point para remarcação, caso esteja fora do perfil
- [11]- configuração da fila física 0
- [12]- configuração da fila virtual 0
- [13]- Limite inferior RED em pacotes
- [14]- Limite superior RED em pacotes
- [15]- prioridade de descarte da fila. Ex.: 2%; 10%
- [16]- definição de um PHB para o Code Point 10 [17]- na fila física 0 e [18]- na fila virtual 0.

### **Código 10 – Configuração das políticas na borda**

Outro aspecto importante é a escolha da política de admissão do fluxo dentro de domínio DiffServ. O NS oferece cinco políticas de filas, sendo:

- 1) **TSW2CMPolicer** (Time Sliding Window with 2 Color Marking). Utiliza as informações das taxas recebidas (CIR) e duas precedências de descarte. A menor precedência é usada probabilisticamente quando a CIR é excedida.
- 2) **TSW3CMPolicer** (Time Sliding Window with 3 Color Marking). Utiliza as informações das taxas recebidas (CIR), as informações das taxas de pico (PIR) e três precedências de descarte. A precedência média de descartes é usada probabilisticamente quando a CIR é excedida e a menor precedência de descarte é usada, também probabilisticamente, quando o quando a PIR é excedida.
- 3) **TokenBucketPolicer**. Utiliza as informações das taxas recebidas (CIR), o tamanho das rajadas recebidas (CBS) e duas precedências de descarte. Nesse caso um pacote que chega ao domínio DiffServ é marcado com a menor precedência se, e somente se, ele exceder o token bucket (ver Figura 12).
- 4) **SrTCMPolicer** (Single Rate Three Color Marker). Utiliza as informações das taxas recebidas (CIR), o tamanho das rajadas recebidas (CBS) e o excesso no tamanho das rajadas (EBS) para escolher entre três precedências de descarte.
- 5) **TrTCMPolicer** (Two Rate Three Color Marker). Utiliza as informações das taxas recebidas (CIR), o tamanho das rajadas recebidas (CBS), as informações das taxas de pico (PIR) e o tamanho das rajadas de pico (PBS) para escolher entre três precedências de descarte.

Além da configuração das políticas de admissão dos nós de borda, deve-se configurar os nós de núcleo para procederem o encaminhamento prioritário com base nas configurações estabelecidas (ver exemplo do Código 11).

**Figura 12 - Balde de Fichas**

```
$qNlBout1 setSchedulerMode WRR[1]
```

```
$qNlBout1 addQueueWeights 0[2] 8[3]
```

[1] - Configuração do escalonador.

Ex.: WRR (Weighted Round Robin) - Baseado em pesos

[2] - Fila Física

[3] - Peso da Fila. 8 = 80% de peso para a fila 0

**Código 11 – Configuração dos Nós de Núcleo**

Concluídas as configurações deve-se injetar tráfego suficiente para congestionar a rede e constatar se a QoS solicitada realmente está sendo provida.

Os Códigos 12 e 13 mostram dois exemplos de tráfego que podem ser utilizados nesse processo. Apenas o tráfego proveniente do nó entitulado *transmissor* receberá o *code point 10* e, portanto será priorizado.

```
set tcpl [new Agent/TCP/Newreno]
$tcpl set class_ 1
$tcpl set fid_ 1
$tcpl set windows_ 4000
set sink1 [new Agent/TCPSink]
$ns attach-agent $transmissor $tcpl
$ns attach-agent $receptor1 $sink1
$ns connect $tcpl $sink1
set ftp1 [$tcpl attach-source FTP]
$ftp1 set codePt_ 10
```

**Código 12 – Tráfego com QoS (FTP)**

```
set udpl [new Agent/UDP]
$ns attach-agent $trafbg1 $udpl
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udpl
$cbr1 set packet_size_ $packetSize
$udpl set packetSize_ $packetSize
$udpl set class_ 2
$cbr1 set rate_ $rate0
$cbr1 set codePt_ 0
set null1 [new Agent/Null]
$ns attach-agent $receptor1 $null1
$ns connect $udpl $null1
```

**Código 13 – Tráfego sem QoS (Vídeo)**

O código completo de uma simulação DiffServ é mostrado no Anexo III.

Após a execução do script de simulação, o trace file deve ser importado para o MS-Access© e as seguintes consultas SQL devem ser criadas:

Seleciona os pacotes enviados pelo transmissor

```
SELECT * INTO tb_sendfid1
FROM trace
WHERE trace.fid=1 AND trace.noentrada=0 and (origem = 0 and destino =
1) and evento = "+";
```

Seleciona os pacotes recebidos pelos receptores

```
SELECT * INTO tb_receivefid1
FROM trace
WHERE trace.fid=1 AND trace.evento="r" and nosaida = 1 and (origem =
0 and destino = 1);
```

Subtrai o tempo dos pacotes recebidos do tempo dos pacotes enviados para obtenção do atraso

```
SELECT S.fid, S.tempo, R.tempo, R.tempo-S.tempo AS atraso INTO atraso
FROM tb_sendfid1 AS S, tb_receivefid1 AS R
WHERE S.idpacote = R.idpacote
ORDER BY R.tempo;
```



### 3.2 Redes Sem Fio/NOMADIC/MANETs (Wireless 802.11)

As redes sem fio vêm ganhando cada vez mais espaço no mercado brasileiro. Diferentes termos são usados para caracterizar esse tipo de rede. O termo Nomadic vem de nômade e faz analogia aos povos que não se fixavam por muito tempo em uma região. Nesse caso específico, além de não serem cabeadas, as redes wireless podem ainda ser móveis, pois seus usuários podem se mover livremente de um ponto ao outro mantendo a funcionalidade.

Existem, basicamente, dois tipos de rede sem fio: As chamadas redes infraestruturadas que se caracterizam por possuir um ponto de acesso que centraliza toda a comunicação. Um bom exemplo são as redes da telefonia celular. Nesse tipo de rede, por mais que se esteja a uma distância muito pequena do dispositivo para o qual se fará a transmissão, deve-se antes transmitir a intermediário (no caso da telefonia celular esse ponto chama-se ERB – Estação Radio Base) para que o sinal seja retransmitido ao destinatário. Outro tipo de rede sem fio chama-se *Ad-Hoc*, e caracteriza-se pela ausência da necessidade de uma infra-estrutura. Normalmente utiliza-se esse tipo de rede em cenários como campos de batalha ou em ações de resgate onde não se justifica ou não há tempo hábil para a montagem de uma infra-estrutura. Nessas redes, cada nó (um notebook, por exemplo) pode atuar horas como um transmissor, horas como um receptor, horas como um roteador. O termo MANET (**M**ó**b**ile **A**d-**H**oc **N**et**w**orks) faz alusão a esse tipo de rede que normalmente trabalha com uma faixa de frequência liberada e segue as especificações de um padrão IEEE conhecido como 802.11.

Padrão	Frequência	Velocidade
802.11	914 MHz	5 Mbps
802.11a	5 GHz	54 Mbps
802.11b	2.4 GHz	11 Mbps
802.11g	2.4 GHz	54 Mbps

**Tabela 2 - Padrões da Família IEEE 802.11**

O módulo de redes móveis do NS não é parte nativa do simulador. Na verdade trata-se de uma contribuição do grupo CMU (Carnegie Mellon University). Alguns ajustes ainda são requeridos para que a funcionalidade do módulo fique mais abrangente. Um exemplo de erro ocorre quando se utiliza a tráfego baseado nas distribuições estatísticas do NS (exponencial ou de pareto). Nesses casos a seguinte mensagem de erro é exibida:

(null) - invalid packet type (exp)

Para corrigir esse problema, deve-se alterar o programa **cmu-trace.cc**, localizado no diretório “/home/ns-allinone-2.26/ns-2.26/trace”, acrescentando-se as linhas mostradas abaixo:

```
case PT_CBR:
    format_rtp(p, offset);
    break;
case PT_DIFF:
    break;
case PT_GAF:
    break;

//Aqui entra o ajuste *****//
case PT_EXP:
format_msg(p, offset);
break;
//*****//

default:
    fprintf(stderr, "%s - invalid packet type (%s).\n",
        __PRETTY_FUNCTION__, packet_info.name(ch->ptype()));
    exit(1);
}
}
}
...
```

Após isso, deve-se recompilar o NS digitando-se “make” no diretório “/home/ns-allinone-2.26/ns-2.26” a partir de um terminal.

A principal dificuldade na simulação de uma rede sem fio é inerente à configuração dos parâmetros que precisam ser ajustados de acordo com o cenário. Todo o desempenho da rede está atrelado a esses parâmetros. Os principais são apresentados na tabela abaixo:

Parâmetro	Valor (Exemplo)	Unidade
Limiar de Captura	10.0	db
Limiar de Carrie Sense	1.559e-11	W
Limiar de Potencia de Recepção	3.653e-10	W
Largura de Banda	2*1e6	Mbps
Potência de Transmissão	0.2818	W
Frequência	2.4e9	GHz
Fator de Perda	1.0	-
Antena Omni direcional	1.5 (altura)	m
<b>Para o modelo de Sombreamento</b>		
PathlossExp (Perda no Caminho)	4	-

**Tabela 3.** Parâmetros de uma placa Lucent WaveLan (Orinoco)

O primeiro passo requerido na simulação de uma rede sem fio é a especificação da topografia ou *flatgrid*. A topografia consiste na área onde a simulação será realizada especificada em metros. É importante que haja um planejamento criterioso, pois os nós só poderão movimentar-se dentro dessa área pré-especificada. Supondo que o cenário de simulação é um Campus Universitário com 250.000 m<sup>2</sup>, a configuração da topografia poderia ser feita em uma área de 500 x 500, da seguinte maneira:

```
set topografia [new Topography]
$topografia load_flatgrid 500 500
```

Existe também a necessidade de definição de um objeto chamado GOD (General Operations Director), que armazena informações globais sobre o ambiente (Rede e Nós) e é considerado um observador onipresente, mas não é conhecido por nenhum dos participantes da simulação. Trata-se de um componente de controle do próprio

simulador. Na definição do objeto GOD, deve-se explicitar o número de nós móveis, que no exemplo abaixo é igual a "2".

```
create-god [expr 2]
```

Também deve-se definir um canal de comunicação, conforme ilustrado abaixo:

```
set chan_1_ [new Channel/WirelessChannel]
```

E os parâmetros mencionados acima:

```
$ns_ node-config -adhocRouting AODV[1] \
                 -llType LL[2] \
                 -macType Mac/802_11[3] \
                 -ifqType Queue/DropTail/PriQueue[4] \
                 -ifqLen 50[5] \
                 -antType Antenna/OmniAntenna[6] \
                 -propType Propagation/TwoRayGround[7] \
                 -phyType Phy/WirelessPhy[8] \
                 -channel $chan_1_[9] \
                   -topoInstance $topografia[10] \
                   -agentTrace ON[11] \
                 -routerTrace OFF[12] \
                 -macTrace OFF[13]
```

[1]-protocolo de roteamento

[2]-tipo de camada de enlace

[3]- tipo MAC (Media Access Control)

[4]-tipo de interface de fila

[5]-número máximo de pacotes na interface de fila

[6]-modelo de antena (OMNI -irradia o sinal por 360°)

[7]-modelo de propagação de rádio

[8]-tipo de interface de rede

[9]-tipo de canal definido acima

[10]- Topografia definida anteriormente

[11],[12],[13]- Parâmetros de Ajuste do Trace. Ex. *routertrace ON* mostra no NAM círculos concêntricos imitando as ondas de rádio.

O passo seguinte é especificar os nós, seus respectivos movimentos e o tráfego que será estabelecido entre eles. A definição dos Nós pode ser feita em um laço, conforme mostrado no código abaixo (Considerar que estão sendo criados apenas 2 Nós):

```
for {set i 0} {$i < 2} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0[1];
}
```

[1]- Desabilita o movimento aleatório dos Nós

O Cenário pode ser definido conforme mostrado abaixo:

```
set god_ [God instance]
$node_(0) set X_ 5.8[1]
$node_(0) set Y_ 22.1[2]
$node_(0) set Z_ 0.0[3]
$node_(1) set X_ 21.0[4]
$node_(1) set Y_ 7.7[5]
$node_(1) set Z_ 0.0[6]
$ns_ at 10.0[7] "$node_(0) setdest 40.0[8] 22.1[9] 0.5[10]"
$ns_ at 30.0 "$node_(1) setdest 21.0 5.4 0.5"
$ns_ at 50.0 "$node_(1) setdest 20 5.4 0.5"
$ns_ at 80.0 "$node_(1) setdest 20 0.5 0.5"
$ns_ at 130.0 "$node_(1) setdest 20 22,1 0.5"
```

Anteriormente foram criados dois Nós denominados node\_(0) e node\_(1). [1],[2] e [3] especificam as coordenadas que marcam a posição inicial do primeiro nó node\_(0). [4],[5] e [6] fazem o mesmo para o nó node\_(1).

A partir daí define-se o movimento dos nós. No momento 10.0 de simulação [7], o nó node\_(0) deve-se deslocar para as coordenadas X= 40.0 [8] Y=22.1[8] a uma velocidade média de 0.5 metros por segundo. [10]

O anexo 4 apresenta um exemplo completo de uma rede wireless ad-hoc configurada para operar em ambiente interno (*indoor*).

Alguns experimentos requerem o uso de redes wireless infraestruturadas. Nesse caso deve-se proceder algumas modificações no simulador e recompilá-lo. Se você precisar dessa funcionalidade siga os passos abaixo:

### Agente de roteamento (NOAH)

NOAH é um agente de roteamento wireless que, em contraste com os protocolos DSDV, DSR, ..., suporta somente a uma comunicação direta entre nós wireless e as estações base (*access points* ou *base stations*) no caso de IP móvel ser usado. Isto permite simular cenários onde multi-saltos no roteamento wireless não são desejados. NOAH não envia qualquer relato de roteamento dos pacotes. Ele foi atualizado em novembro de 2003 para trabalhar com o ns-2.26 e com cenários IP sem movimento.

Instruções de instalação passo a passo para o ns-2.26/ns-2.27

<b>Makefile.in</b>	adicione <code>noah/noah.o</code> \ em <code>OBJ_CC</code> e <code>tcl/mobility/noah.tcl</code> \ em <code>NS_TCL_LIB</code>
<b>noah/noah.{h,cc}</b>	adicione <code>noah.h</code> e <code>noah.cc</code> em um novo subdiretório <code>noah/</code>
<b>tcl/mobility/noah.tcl</b>	adicione <code>noah.tcl</code> ao <code>tcl/mobility/</code>
<b>tcl/lib/ns-lib.tcl.h</b>	<pre> linha 191: adicione <code>source ../mobility/noah.tcl</code> linha 603ff: adicione     NOAH {                                 set ragent [\$self create- noah-agent \$node]     } linha 768ff: adicione Simulator instproc create-noah-agent { node } {     # Create a noah routing agent for this node     set ragent [new Agent/NOAH]      ## setup address (supports hier-addr) for noah agent     ## and mobilenode     set addr [\$node node-addr]      \$ragment addr \$addr     \$ragment node \$node      if [Simulator set mobile_ip_] {         \$ragment port-dmux [\$node demux]     }     \$node addr \$addr     \$node set ragent_ \$ragment     return \$ragment } </pre>

## SIMULANDO COM REDES AD-HOC

O processo de simulação envolvendo redes *wireless* é consideravelmente diferente em relação ao das redes cabeadas. Uma das mudanças mais significativas ocorre em relação ao arquivo de *trace* que possui um lay-out diferente ao das redes cabeadas. Na verdade existem duas opções de trace quando se utiliza uma simulação com redes sem fio: o trace clássico e o novo trace. O novo trace (adotado nos exemplos deste livro) é invocado apenas com o acréscimo do comando: `$ns_ use-newtrace`.

```
s -t 1.382517059 -Hs 3 -Hd -2 -Ni 3 -Nx 1.00 -Ny 1.00 -Nz 0.00 -Ne -
1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 4196353.0 -Id
0.0 -It pareto -Il 210 -If 1 -Ii 284 -Iv 32 -Pn cbr -Pi 0 -Pf 0 -Po 0
```

item	Significado
s/r/+/-/d	Evento ocorrido com o pacote {envio, recebimento, enfileiramento, desenfileiramento, descarte, etc}
-t	Tempo de simulação na ocorrência do evento
-Hs	Identificação do nó onde ocorreu o evento
-Hd	Identificação do próximo nó a ser percorrido pelo pacote
-Ni	Identificação do Nó
-Nx	Coordenada X do Nó
-Ny	Coordenada Y do Nó
-Nz	Coordenada Z do Nó
-Ne	Nível de energia do Nó
-NI	Nível do Trace (AGT, RTR, MAC) {Agente, Roteamento, MAC}
-Nw	Razão do evento: {END: Descarte no Fim da Simulação; COL: Descarte por colisão MAC; TTL: Tempo de vida alcançou 0, etc.
-Ma	Duração do pacote no nível MAC
-Md	Endereço ethernet de destino
-Ms	Endereço ethernet de envio
-Mt	Tipo ethernet
-Is	Endereço (IP) fonte. Porta Fonte
-Id	Endereço (IP) destino. Porta destino
-It	Tipo do Pacote (CBR, Pareto, Exponencial, etc)
-Il	Tamanho do Pacote
-If	Identificação do fluxo ao qual o pacote pertence
-Ii	Identificação única do pacote
-Iv	Valor do TTL
-Pn	Dependendo do tipo de pacote (CBR, ARP, DSR, TCP) estes campos
-Pi	podem assumir significados diferentes.
-Pf	
-Po	



## 4 Referência Bibliográfica Básica

[1] MacDougall, M. H.; "Simulating Computer System Techniques and Tools", The MIT Press, Cambridge, Massachusetts, London, England, 1987.

[2] Jain, Raj; "The Art of Computer Systems Performance Analysis Techniques for Experimental Design, Measurement, Simulation and Modeling"; John Wiley & Sons Inc., ISBN: 0-471-50336-3, 1998. <http://www.cis.ohio-state.edu/~jain>

[3] Fall, K.; Varadhan, K.; "The NS Manual"; Network Simulator 2.1b9a, VINT Project; 2002. <http://www.isi.edu/nsnam/ns/>

[4] Braden, R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S.; "Resource Reservation Protocol", RFC2205, September 1997.

[5] Rosen, E; "Multiprotocol Label Switching Architecture, Internet Draft"; draft-ietf-mpls-arch-05.txt; April 1999.

[6] Xiao, X.; Ni, L.M., "Internet QoS: A Big Picture, IEEE Network", March/April 1999.

[7] OTCL Tutorial. <http://hegel.ittc.ukans.edu/topics/tcltk/tutorial-noplugin/>

[8] Mark Greis Tutorial. <http://www.isi.edu/nsnam/ns/tutorial/index.html> .

[9] Santana, M; Santana, R.; Francês, R.; Orlandi, R; "Tools and Methodologies For Performance Evaluation of Distributed Computing Systems – A Comparison Study"

## ANEXO IV

### Guia de referência básica TCL/TK

#### Definição de variáveis

```
set nome_da_variável valor_inicial  
Ex.: set wvetfid ""  
    set x 0
```

#### Estruturas básicas de repetição

```
for {set g 1} {$g < $wcontafluxos+1} {incr g 1} {  
    comandos  
}
```

```
while {[eof $wtracel]} {  
    comandos  
}
```

```
set i 1  
foreach value {1 3 5 7 11 13 17 19 23} {  
    comandos  
}
```

#### Estruturas de decisão

```
switch $x {  
    a {incr t1}  
    b {incr t2}  
    c {incr t3}  
}  
  
if (strcmp(argv[1], argv[2]) == 0) {  
    interp->result = "1";  
} else {  
    interp->result = "0";  
}
```

#### Sub Rotinas

```
$ns at 10.0 "encerra"  
  
proc encerra {} {  
    global variáveis globais  
    comandos  
}
```

**Comentários**

# A linha toda torna-se um comentário

**Expressões Matemáticas**

Expr 7.2 / 3

=> 2.4