UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2569

A Ferramenta *Network* Simulator 2 (NS-2)

Eduardo Martinelli Galvão de Queiroz
Sarita Mazzini Bruschi

Nº 255

RELATÓRIOS TÉCNICOS

São Carlos – SP
Abr./2005

A ferramenta *Network Simulator 2 (NS-2)*

Eduardo Martinelli Galvão de Queiroz, Sarita Mazzini Bruschi

eduq@grad.icmc.usp.br, sarita@icmc.usp.br

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Laboratório de Sistemas Distribuídos e Programação Concorrente
C.P. 668, CEP 13560-970, São Carlos – SP, Brasil

Resumo:

Este relatório tem o objetivo de introduzir ao leitor a ferramenta Network Simulator 2 (NS-2), que simula vários modelos de redes de computadores. Este documento mostra a arquitetura, o funcionamento, algumas ferramentas para a análise das simulações, além de um exemplo de simulação de uma topologia de uma rede de computadores. O programa realiza várias simulações de tipos de redes, como redes sem fio (Wireless) e conexões com satélite. Neste texto será dada ênfase ao funcionamento do simulador para redes de computadores com fio (Wired).

Abril de 2005

Sumário

Lista de Figuras

1. Introdução

Uma das maiores revoluções que aconteceram nos últimos anos foi o rápido avanço da Internet. Com a evolução da Internet surgiram muitos problemas, os quais são alvos constantes de pesquisa. Um desses problemas são as redes de computadores, as quais se tornaram cada vez mais rápidas e mais complexas. As pesquisas na área de redes de computadores envolvem: projetos, protocolos, serviços, gerenciamento e aplicações.

Devido à grande evolução, tanto em termos tecnológico quanto em termos de utilização, tornou-se essencial o desenvolvimento de um estudo sobre o desempenho das redes, estudo este que traz grandes benefícios para o funcionamento das mesmas. Nesse sentido, uma das técnicas que possibilita a avaliação de desempenho das redes é a simulação.

As fases de desenvolvimento de uma simulação não são simples de serem realizadas. É necessário abstrair as características do sistema em um modelo, transcrever esse modelo em um programa de simulação e analisar os resultados de modo que as variáveis aleatórias não interfiram nos resultados. Atualmente, diversas ferramentas que têm como objetivo facilitar o desenvolvimento de uma simulação podem ser encontradas na literatura. Exemplos de ferramentas que auxiliam no desenvolvimento de simulações de redes de computadores são *OMNeT++* (Varga, 2004) e *Network Simulator* (NS-2, 2005).

Este relatório técnico descreve a ferramenta NS-2 (seção 2), incluindo sua instalação nos ambientes Windows e Linux (seções 3 e 4), seu funcionamento (seções 5, 6 e 7), a maneira como os resultados da simulação podem ser analisados (seção 8) e as considerações finais sobre a ferramenta (seção 9).

2. A ferramenta NS-2

A ferramenta NS *(Network Simulator)* (NS-2, 2005) iniciou-se como uma variante do *REAL Network Simulator* (Real, 2005) no ano de 1989. O *REAL Network Simulator* é dedicado ao estudo do comportamento dinâmico de fluxo de dados e controle de congestionamento e provê aproximadamente 30 módulos (escritos em C) que simula vários protocolos de controle de fluxo (como o TCP). Em 1995, o desenvolvimento do *NS* passou a ser financiado pelo *DARPA (Defense Advanced Research Projects Agency*, órgão do governo norte-americano ligado ao *DoD (Department of Defense))* através do projeto *VINT* (VINT, 2005), cujo propósito é a construção de um simulador de redes de computadores e é um projeto colaborativo envolvendo várias entidades de pesquisa: USC/ISI *(University of Southern Califórnia – Information Sciences Institute)*, Xerox PARC (Palo Alto Research Center), *LBNL (Lawrence Berkeley National Laboratory)* e *UC Berkeley (University of Califórnia in Berkeley).*

Atualmente, o projeto é ainda financiado pelo *DARPA*, através do projeto *SAMAN (Simulation Augmented by Measurement and Analysis for Networks)* (SAMAN, 2005) e também pelo *NSF (Nacional Science Foundation)* através do projeto *CONSER (Collaborative Simulation for Education and Research)* (Conser, 2005). O projeto *SAMAN* pesquisa o problema de como tornar os protocolos e as operações existentes em redes de computadores mais robustos às falhas. O objetivo deste projeto é entender, detectar e evitar condições de falha com o uso de ferramentas como o *NS*-2. Já o projeto *CONSER* tem o objetivo de apoiar a pesquisa na

avaliação e desenvolvimento de protocolos e também apoiar o ensino de conceitos sobre protocolos existentes e novos.

A versão atual da ferramenta é a 2.28 e a transição da versão 1.0 para a atual ocorreu no ano de 1996. As novas versões lançadas são oriundas da própria inclusão de novas funcionalidades à ferramenta, além da correção dos erros reportados pelos usuários. Através do endereço eletrônico *http://www.isi.edu/nsnam/ns/ns-lists.html* é possível participar de listas de discussão sobre diversos assuntos relacionados à ferramenta, como instalação e funcionamento.

Na próxima seção, serão mostrados os passos para a instalação no *NS-2*.

3. Instalação da ferramenta no ambiente *Linux*

A instalação do *NS-2* pode ser feita tanto sistema operacional *Linux* quanto no sistema operacional *Windows*. A instalação no *Linux* requer o arquivo *ns-allinone-2.28.tar.gz* (que pode ser obtido em *http://www.isi.edu/nsnam/dist/ns-allinone-2.28.tar.gz*), que contém vários pacotes necessários e opcionais ao funcionamento da ferramenta. O arquivo contém os seguintes pacotes:

- *Tcl/Tk (Tool Command Language)* versão 8.4.5 → Linguagem de comando com a qual as simulações no NS-2 são montadas. (Necessário)
- *Otcl (MIT Object Tcl)* versão 1.9 → Extensão do *Tcl/Tk* para programação orientada a objetos. (Necessário)
- *TclCL (Tcl with Classes)* versão 1.16 → Interface *Tcl/C++* usada pelo *NS* (Necessário)
- *NS-2* versão 2.28 → A ferramenta em si (Necessário)
- *Nam (Network Animator)* versão1.11 → visualizador gráfico do comportamento das simulações. (Opcional)
- *Xgraph* versão 12.1 → Software que gera gráficos com o arquivo de saída das simulações do *NS-2*. (Opcional)
- *Cweb* versão 3.4g, *SGB* versão 1.0, *gt-itm* e *sgb2ns* versão 1.1 e *Zlib* versão 1.1.4 → programas de conversão de tipo de arquivos (Opcionais)

Descompactando-se o arquivo *ns-allinone-2.28.tar.gz*, o diretório *ns-allinone-2.28* será criado com os seguintes subdiretórios: *Cweb, Gt-itm, Nam-1.11, Ns-2.28, Otcl-1.9, Sgb, Tcl8.4.5, Tclcl1.16, Tk8.4.5, Xgraph-12.1, zlib-1.1.4.*

Para a instalação do pacote inteiro basta digitar o comando `./install` no diretório *ns-allinone-2.28* e assim todos os componentes irão ser instalados e caso ocorra algum erro, o mesmo é reportado. No entanto, para a utilização apenas do *NS-2*, é necessário que no diretório *../ns-allinone-2.28/ns-2.28* se digite os comandos `./configure`, depois `./make` e finalmente `./validate`.

4. Instalação da ferramenta no ambiente *Windows*

A ferramenta *NS-2* também pode ser instalada no sistema operacional *Windows*. Neste ambiente, a instalação também não é complicada quando utilizado um arquivo compilado do

NS-2, que pode ser encontrado no endereço *http://www.isi.edu/nsnam/dist/binary/ns-2.1b9a.exe.* Além deste arquivo, é necessário o arquivo *tcl830.exe* (*ftp://ftp.scriptics.com/pub/tcl/tcl8_3/*), que instalará a linguagem *tcl* no computador.

Como opcional, pode ser obtido o arquivo *nam-1.0a11-win32.exe (http://www.isi.edu/nsnam/dist/binary/ nam-1.0a11-win32.exe)*, que é o arquivo do programa de animação *NAM*, também compilado para o *Windows*.

Com estes arquivos, a instalação pode ser feita da seguinte maneira:

- Clicar duas vezes sobre o arquivo *tcl830.exe*. Por padrão, o programa será instalado no diretório *C:\Arquivos de Programas\tcl*;
- Renomear os arquivos *ns-2.1b9a.exe* e *nam-1.0a11-win32.exe* para *ns.exe* e *nam.exe*;
- Copiar os arquivos *ns.exe* e *nam.exe* para o diretório *C:\Arquivos de Programas\tcl\bin.*

Concluídos estes procedimentos, o programa está pronto para uso, bastando clicar duas vezes sobre o ícone *ns.exe* do diretório *C:\Arquivos de Programas\tcl\bin.*

Os procedimentos descritos funcionaram em um computador com o *Windows 2000 ME* instalado e também em outro com o *Windows XP* instalado.

A forma de uso da ferramenta, assim como suas estruturas de funcionamento são expostas nas próximas seções.

5. Arquitetura da ferramenta

O funcionamento da ferramenta *NS-2* é estruturada com uma especificação inicial da topologia a ser simulada (incluindo dados sobre *links*, tamanho de pacotes, protocolos, etc) e a sua real simulação de eventos discretos é realizada em linguagem *C++*. A ferramenta usa uma lista de eventos para as simulações e utiliza um modelo simples de controle, com uma única *thread*, ou seja, sem mecanismos de *locking* ou *race conditions*, simplificando o funcionamento. Nas próximas seções, são mostrados os aspectos das diferentes linguagens de programação usadas pela ferramenta e o funcionamento dos chamados nós, que simulam o comportamento de um computador em uma rede de computadores.

5.1. Interação entre as linguagens

Para a especificação da simulação, é usada a linguagem *Tcl*, que conecta as especificações da simulação com as respectivas classes do núcleo do programa. Na figura 1, é possível visualizar esta conexão entre as camadas.

Como pode ser observado, o núcleo da ferramenta, que são as classes *C++*, recebe os comandos escritos em *Tcl* e então conecta os mesmos com as respectivas classes. Na mesma figura, observa-se como primeira camada o cenário da simulação. Esse cenário pode ser feito graficamente (desenhando-se com nós, *links,* etc), através da ferramenta *Nam*, que oferece este recurso. Porém, as especificações mais detalhadas dos ambientes, como a configuração de algumas variáveis de alguns protocolos, não podem ser alteradas. Desta maneira, é recomendável, para simulações complexas, a utilização da linguagem *Tcl*, por oferecer todos os recursos possíveis de configuração que a ferramenta pode oferecer.

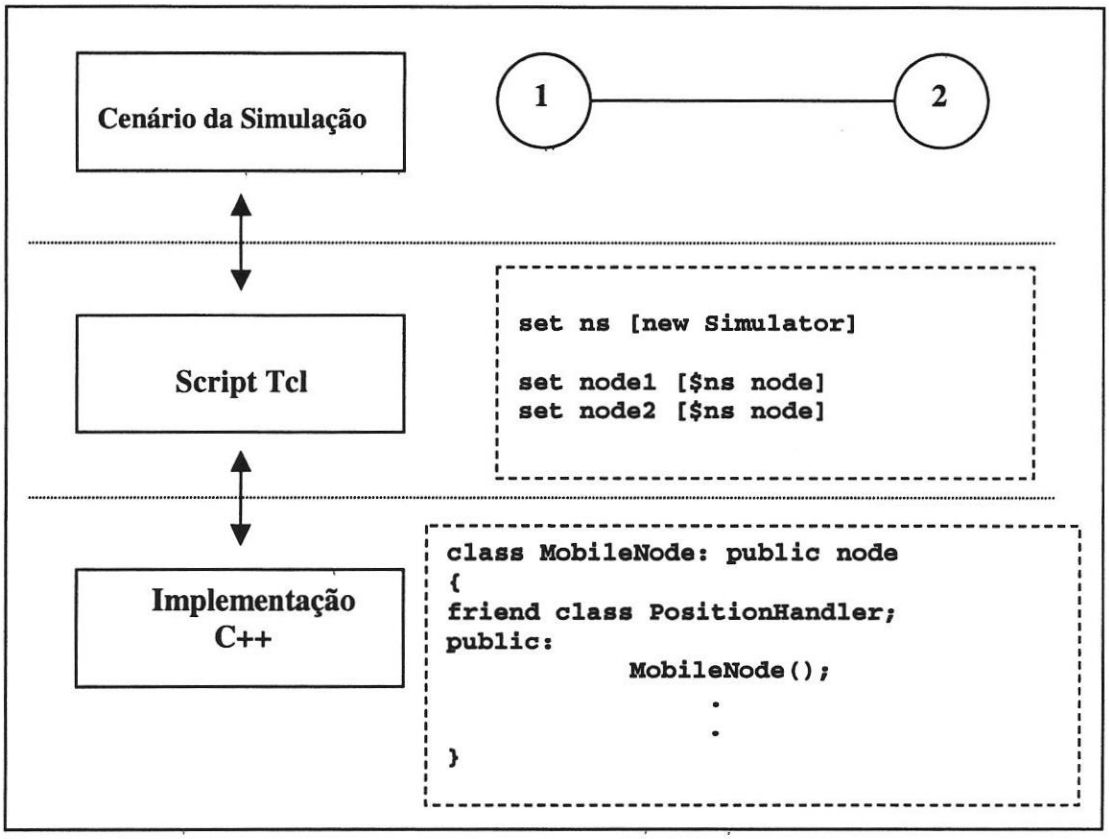

Figura 1. Arquitetura de Funcionamento do *NS-2*

Para a parte de dados, processamento de pacotes, detalhamento e outras atribuições de simulação, o núcleo *C++* é o responsável. Já para a parte de configuração de cenários de simulação, manipulação de objetos C++, a responsabilidade fica por conta da linguagem *Tcl* e suas variantes *OTcl*, *Tk* e *TclCL*.

Uma possível visão que se pode ter da correspondência entre as classes C++ e os componentes *OTcl* é exposto na figura 2.

Com isso, os objetos *OTcl* obtêm correspondências com os objetos C++ do núcleo da ferramenta. Esse aspecto facilita a construção de simulações, pois se faz uso de uma linguagem de comandos (*Tcl*) de fácil aprendizagem em vez de uma linguagem mais complexa que é a *C++*.

5.2. Protocolos de comunicação

No âmbito geral, a ferramenta conta com vários modelos de tráfegos para simulação (como *FTP, Telnet, Constant Bit Rate*), protocolos de transporte (como TCP, UDP), roteamento (como roteamento *ad-hoc*) e também simulação de meios físicos como redes *Wireless.* Na figura 3, são mostrados organogramas com as hierarquias de diversos tipos de classes que o programa possui para a simulação de protocolos.

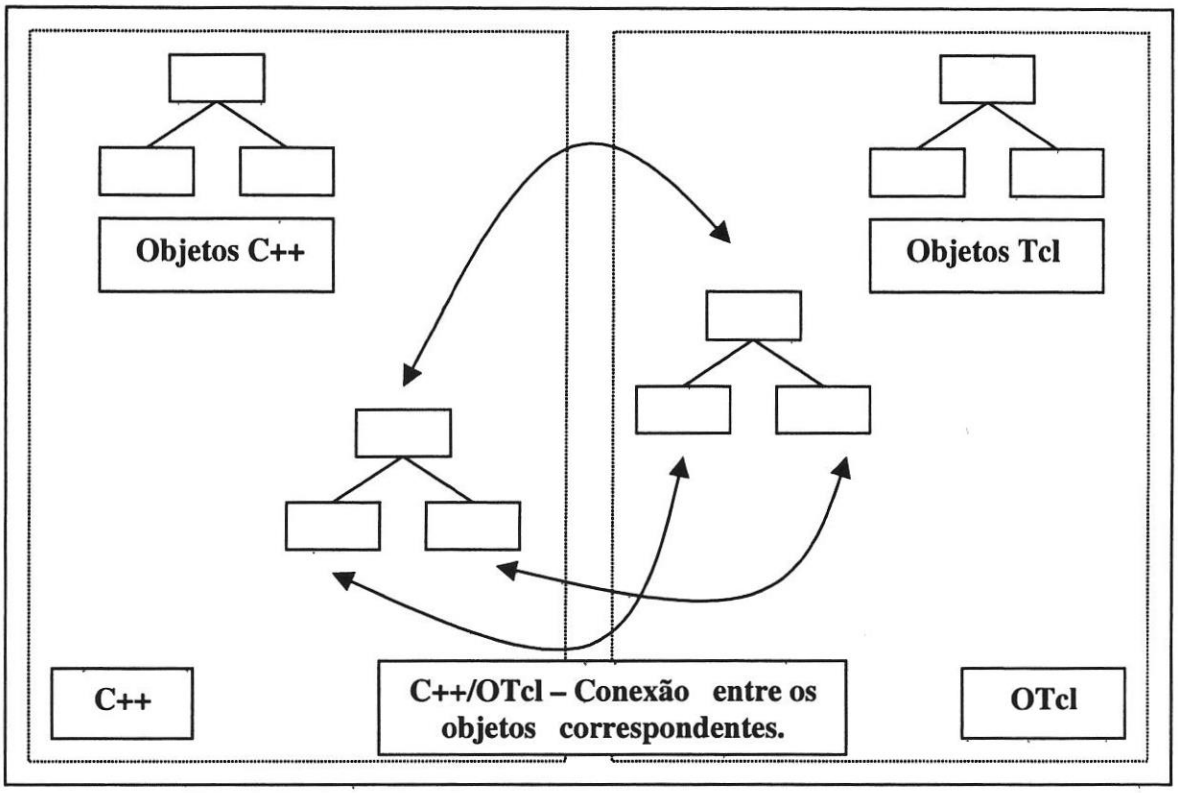

Figura 2. Correspondência entre os objetos das linguagens

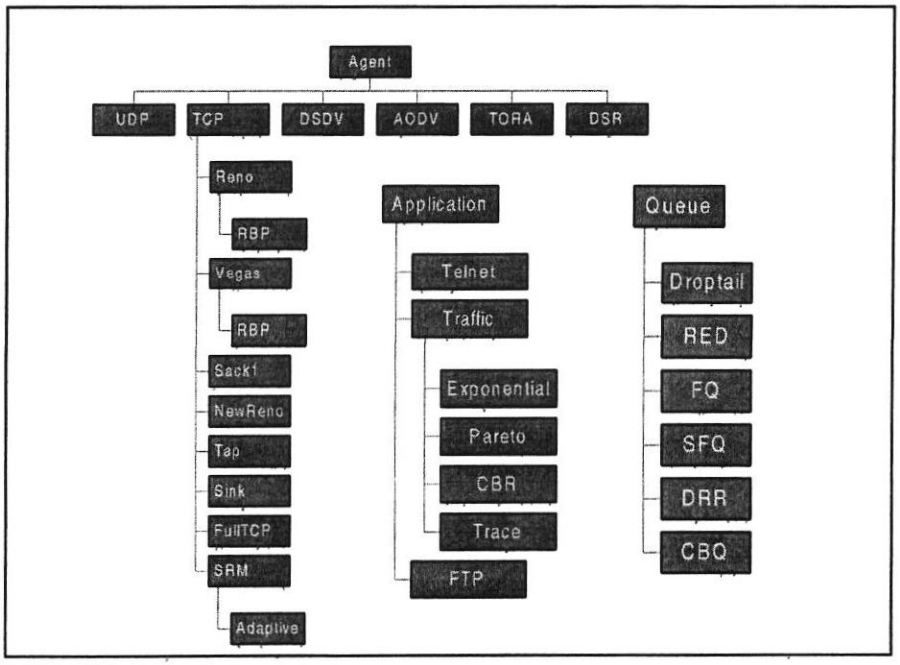

Figura 3. Protocolos, aplicações e tipos de filas simulados no *NS-2*

Nesta figura é possível perceber que o programa contém várias variantes do protocolo *TCP*, como *TCP Reno, Vegas*, etc. Para as aplicações, pode-se ter *Telnet, FTP* ou algum tráfego que pode ser de distribuição exponencial ou *pareto*, por exemplo. Para as filas, podemos ter, por exemplo, a do tipo *DropTail*, que implementa uma fila *fifo* ou do tipo DRR, que implementa o escalonamento *deficit round robin*.

5.3. Funcionamento dos nós

Todas os protocolos, aplicações e tipos de filas são especificados na declaração da simulação na linguagem *Tcl*, assim como toda a topologia da rede a ser simulada. Cada ponto na rede (que pode ser entendido como um computador) é chamado de nó e a conexão entre dois deles, com um agente (que pode ser *TCP, UDP*, etc) e uma aplicação em cada nó, pode ser abstraído pelas figuras 4 e 5.

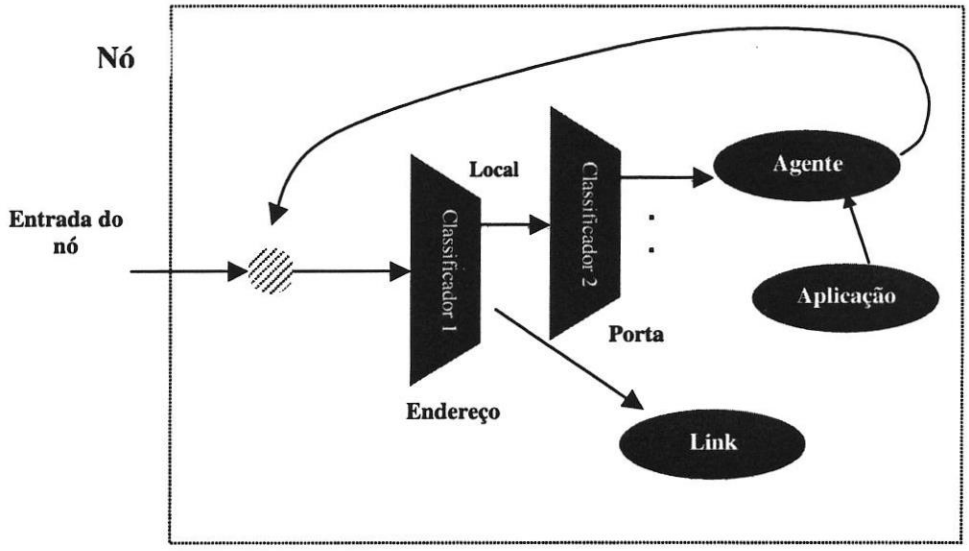

Figura 4. Arquitetura de um nó na rede

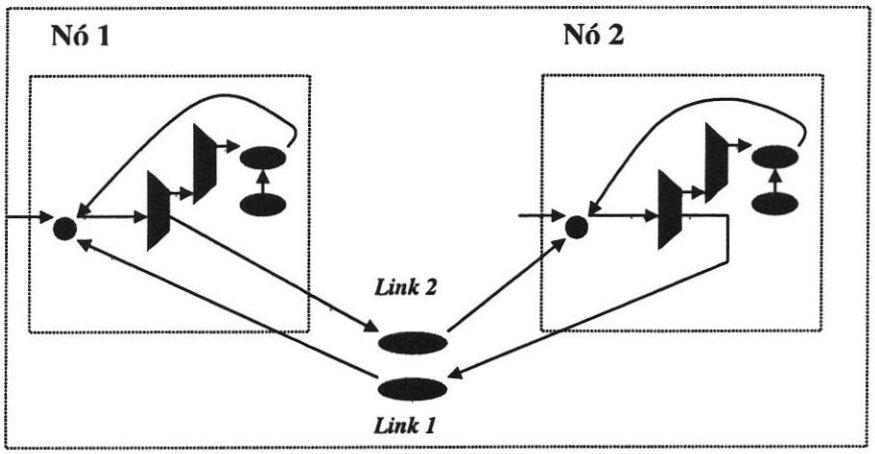

Figura 5. Conexão entre dois nós através de um *link*

Na figura 4 vemos a entrada do nó, dois classificadores, um agente, uma aplicação e um *link* (o numero de agentes, aplicações e *links* podem ser mais do que um), representando a estrutura de um nó. Através da entrada do nó (representado pelo ponto tracejado) é feita a conexão a outros nós e os dois classificadores existentes servem para classificar os pacotes que chegam ou que são gerados pelo nó. Com isso, o classificador 1 (que pode ser observado na figura 4) verifica se o pacote tem como destino o nó local (neste caso, o classificador 1 manda o pacote para o classificador 2) ou algum *link* que está conectado a ele (passando o pacote, então, para algum *link*). O agente presente no nó (que pode ser *UDP, TCP, etc*) gera pacotes e os manda para a entrada (que pode ser visualizado pelas setas nas figuras 4) e assim, o classificador 1 os envia aos *links* correspondentes. Pode-se ter o caso da chegada de um pacote gerado por um outro nó e que é enviado a outro, mas precisa passar por um ou mais nós intermediários. Temos então o outro caso de uso do classificador 1, que irá transferir o pacote de chegada para o *link* ao qual precisa ser enviado.

Na figura 5 é representada a conexão entre dois nós e como a conexão tem duas vias, temos dois *links* que os interligam. O *link* 1 conecta o classificador do nó 2 com a entrada do nó 1, enquanto o *link* 2 conecta o classificador do nó 1 com a entrada do nó 2. Com isso, um pacote enviado para o nó 2 oriundo do nó 1 irá passar pelo *link* 2 chegando à entrada do nó 2. Como o pacote destina-se ao nó 2, o classificador 1 repassará o mesmo para o classificador 2, que por sua vez vai repassa-lo ao agente. O agente então processará os dados e então vai gerar um pacote destinado ao nó 1 (neste caso, podemos pensar em um protocolo *TCP* gerando um pacote *Ack).* Este pacote gerado voltará à entrada do nó 2 e o classificador 1 o repassará ao *link* 1, que conecta o nó 2 com o nó 1. Chegando à entrada do nó 1, este pacote vai percorrer o classificador 1, que então o encaminhará para o classificador 2, finalmente chegado ao agente do nó 1.

6. Desenvolvimento das simulações

O desenvolvimento da simulação efetuada via linguagem *Tcl* provê a possibilidade da especificação de vários aspectos relativos ao ambiente idealizado e que se deseja simular. Além disso, é possível a montagem de cenários de simulações através da ferramenta *Nam (Network Animator).* Na próxima seção vão ser expostos alguns aspectos básicos desta ferramenta e na seção 5.2, um exemplo de rede de computadores que servirá de guia para a abordagem da montagem de simulações utilizando a linguagem *Tcl.*

6.1. A ferramenta Nam (Network Animator)

A ferramenta *Nam* pode ser executada através do duplo clique no ícone *nam.exe* no diretório *C:\Arquivos de Programas\tcl\bin* no ambiente *Windows* ou com o comando `./nam` no diretório *../ns-allinone-2.28/nam-1.10* no ambiente *Linux.*

Para a montagem do cenário, é necessário acessar o comando *New Nam Editor*, localizado no menu *File* na janela do programa. Feito isso, uma nova janela é aberta com as opções de se colocar nós (os computadores da rede), *links*, agentes, aplicações (que aparecem como geradores de tráfego) e modelos de perdas, que simulam conexões ruins entre os nós. Uma visão da ferramenta pode ser observada na figura 6.

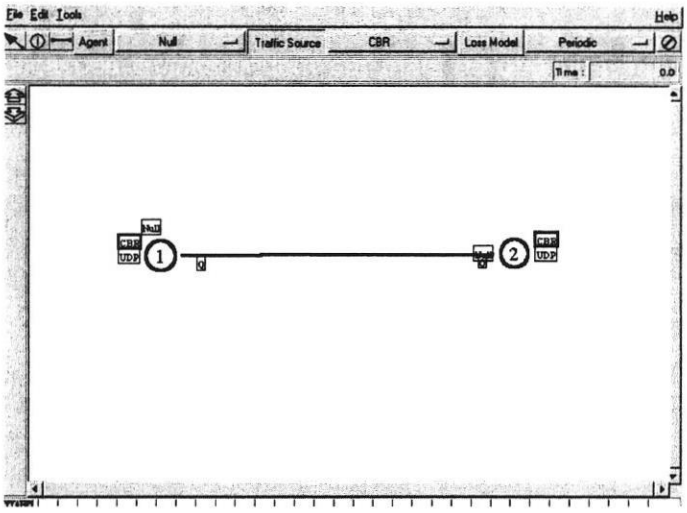

Figura 6. Ferramenta *Nam* com uma rede de dois nós.

A figura 6 mostra um modelo de rede com dois nós e um link existente entre os mesmos. Tanto o nó 1 quanto nó 2 possuem um agente *UDP,* um agente *null* e um gerador de tráfego (que simula alguma aplicação) *CBR (Constant Bit Rate).* Para o *link*, é necessário que se escolha o terceiro ícone da esquerda para direita no lado superior da janela e que se conecte os nós e é possível especificar alguns parâmetros clicando com o botão direito do *mouse* sobre ele. Esses parâmetros são a cor na qual se deseja que apareça o tráfego na hora da execução da simulação, a velocidade do *link* (em *MegaBits* por segundo) e o *delay* (em microssegundos).

Clicando-se também com o botão direito do *mouse* sobre a letra *Q* que aparece do lado de cada nó, é possível escolher o tipo de tipo de fila que se deseja assim como o tamanho máximo da mesma.

Para os protocolos e aplicações, o procedimento para se configurar os parâmetros segue o mesmo padrão. Para protocolos, por exemplo, pode-se escolher o tipo (*TCP, UDP*) no lado superior da janela e clicar sobre o ícone *Agent.* Desta maneira, basta então clicar sobre o nó desejado para associá-lo ao mesmo. Estando com todos os agentes associados aos nós, é necessária ainda a conexão entre os protocolos, que pode ser feita utilizando novamente o botão destinado à configuração dos *links.*

No menu *Edit*, na opção *Simulator Properties*, define-se o tempo de simulação, em segundos. Para a execução, há a opção *Run ns*, no menu *File.* Este comando abre uma janela para que se salve um arquivo com extensão *ns.* Depois de salvo, o mesmo deve ser executado pelo programa *NS-2.* Este arquivo, na realidade, é um arquivo de configuração em linguagem *Tcl* e que pode sofrer modificações. Para executa-lo, digita-se `./ns nome_do_arquivo` no *prompt* da ferramenta.

6.2. Exemplo de configuração através da linguagem *Tcl*

Como explicado anteriormente, a linguagem *Tcl* é utilizada para especificar a topologia da rede a ser simulada, assim como todas outras especificações. Com este tipo de configuração, pode-se configurar vários aspectos específicos da simulação, que a ferramenta *Nam*, por exemplo, não faz, por suportar apenas parâmetros básicos.

Para a configuração de qualquer simulação via linguagem *Tcl*, basta utilizar um editor simples de texto, como o *pico* no ambiente *Linux* ou o bloco de notas, no ambiente *Windows*. Depois de concluído, basta salvar o arquivo com a extensão *.tcl* e executa-lo no *NS-2*.

Um exemplo será utilizado com o objetivo de ilustrar a maneira de se especificar uma simulação no *NS-2*. Este exemplo é composto por 4 nós e dois destes geram tráfego de maneira constante (utilizando o gerador *CBR (Constant Bit Rate)*), e o link entre os nós 2 e 3 é composto por uma política de fila SQF (*Stochastic Fair Queueing)*, conforme mostra a figura 7.

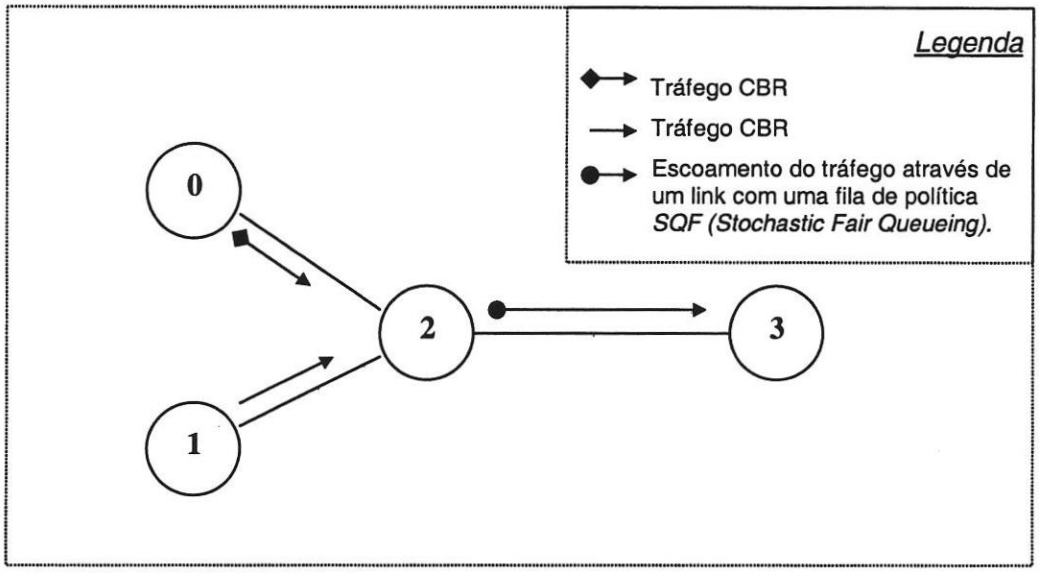

Figura 7. Topologia do exemplo

Toda simulação no NS-2 inicia com a construção de um objeto do simulador, que é feito com o seguinte comando:

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

Com o objeto *ns* criado, podemos criar um arquivo para os dados usadas pelo visualizador *NAM* (isso é opcional).

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

Estes dois commandos criam o arquivo *out.nam* no diretório em que se estiver usando o simulador *NS-2*. A seguir, é criado um arquivo com o nome *out.tr*, que guarda todas as especificações dos pacotes enviados pela simulação em um formato próprio. Este arquivo pode ser utilizado em ferramentas para a análise da simulação, gerando gráficos e alguns valores. Estas ferramentas, assim como o formato do arquivo são mostrados na próxima seção.

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

As linhas seguintes definem duas cores diferentes para os fluxos de dados e serão usados na hora da especificação dos protocolos.

```
set cor 1 Blue
set cor 2 Red
```

Para o encerramento da simulação, faz-se necessário um procedimento, que pode ser observado na figura 8.

```
proc finish {} {
      global ns arq_nam arq_trace
      $ns flush-trace

      #Fechando o arquivo do visualizdor NAM
      close $arq_nam
      #Fechando o arquivo da saída de dados
      close $arq_trace

      #Executando o programa Nam automaticamente
      exec nam out.nam &
      exit 0
}
```

Figura 8. Código para finalização da simulação

Neste procedimento descrito, é importante ressaltar que na linha 2 (`global ns`) são colocados os nomes de todos os arquivos criados, ou caso seja algum outro tipo de procedimento, todas as variáveis usadas dentro do mesmo. Esse fato é importante pois o esquecimento de um nome pode causar erros na execução da simulação. Outro fato a ser considerado é que a linha *`exec nam out.nam &`* pode ser retirada, caso não se queira a execução do programa *NAM* de imediato.

A próxima etapa é a criação dos nós, que no caso são quatro.

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

Os *links* podem ser de dois tipos: o *simplex*, de uma única via, ou o *duplex*, com duas vias (ida e volta). A forma de especificação é a seguinte: *$ns tipo_link(simplex ou duplex) nó_origem nó_destino velocidade(megabits/seg) atraso(em ms) tipo_fila.* Os *links* para o exemplo são:

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
```

Os três *links* são do tipo *duplex*, com uma velocidade de 1 *Megabit* por segundo, com um *delay* de 10 ms e dois destes utilizam a política *DropTail* e um que utiliza a política *SQF.*

Para a orientação quanto ao tráfego em relação a topologia a ser representada pela ferramenta *NAM*, é interessante especificar as orientações dos fluxos que serão mostrados na animação, como mostrado a seguir.

```
$ns duplex-link-op $n0 $n2 orient right-down
```

```
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

A fila implementada pelo *link* entre os nós 2 e 3 pode ser visualizada através da ferramenta *NAM* e para isso é necessário que se use o seguinte comando:

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

A partedo comando `queuePos 0.5` o tempo de 0.5 segundo para o início da visualização.

A partir de agora, são necessárias as especificações dos protocolos e geradores de tráfego que vão ser simulados neste exemplo. O próximo comando cria um agente *UDP*.

```
set udp0 [new Agent/UDP]
$udp1 set class_ 1
$ns attach-agent $n0 $udp0
```

Nos comandos acima, a segunda linha define, para a classe de tráfego do agente *udp1*, o valor 1, que é a cor azul especificada no começo do exemplo e a terceira linha acopla o agente *udp0* ao nó 0 (`n0`).

Para a geração de tráfego, é criado um gerador *CBR (Constant Bit Rate)* como o que se segue.

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 600
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

A primeira linha cria um objeto do tipo *CBR* chamado *cbr0*, enquanto a segunda linha especifica o tamanho do pacote (600 *bytes*), a terceira o intervalo entre os envios dos pacotes e a quarta finalmente acopla o gerador *cbr0* ao agente *udp0*.

Para o nó 1, é colocado um agente *UDP* e um gerador de tráfego do tipo *CBR*, ou seja, com uma configuração igual à colocada para o nó 0, apenas com algumas mudanças quanto ao tamanho do pacote e intervalo de envio dos mesmos.

```
set udp1 [new Agent/UDP]        #Agente UDP
$udp1 set class_ 2              #Comando para a cor do tráfego
$ns attach-agent $n1 $udp1      #Acopla o nó 1 ao agente udp1


set cbr1 [new Application/Traffic/CBR]  #Tráfego CBR
$cbr1 set packetSize_ 500               #Tamanho do pacote
$cbr1 set interval_ 0.008               #Intervalo de envios
$cbr1 attach-agent $udp1                #Acopla cbr1 com udp1
```

No nó 3, é colocado um tipo de agente chamado *Null*, que apenas recebe os pacotes que chegam.

```
set null0 [new Agent/Null]      #Cria agente null0
$ns attach-agent $n3 $null0     #Acopla com o nó 3
```

Depois de todas essas configurações, são necessárias as conexões entre os agentes e o agente *null0*, localizado no nó 3.

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

Com a intenção de escalonar o início e o fim do funcionamento dos geradores de tráfego, fazemos:

```
$ns at 0.5 "$cbr0 start"      #Inicia o tráfego cbr0
$ns at 1.0 "$cbr1 start"      #Inicia o tráfego cbr1
$ns at 4.0 "$cbr1 stop"       #Pára o tráfego cbr1
$ns at 4.5 "$cbr0 stop"       #Pára o tráfego cbr0

$ns at 5.0 "finish"

$ns run
```

Finalmente, com os dois últimos comandos, o procedimento *finish* é chamado no tempo de 5 segundos e a simulação é executada.

Para a execução deste exemplo, basta salva-lo com a extensão *tcl* abrir o arquivo *ns.exe*, no caso do ambiente *Windows*, e digitar */.ns nome_arquivo.tcl*. No caso do ambiente *Linux*, também é necessário digitar o comando */.ns nome_arquivo.tcl*.

Na execução da simulação, é aberta a janela do software *NAM*, com a topologia especificada, conforme pode ser visualizada na figura 9.

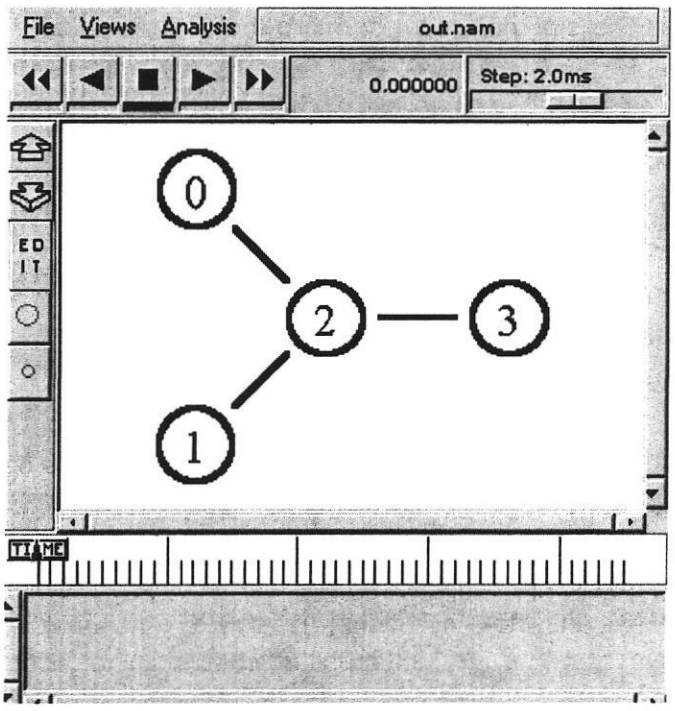

Figura 9. Topologia do exemplo descrito

Para o início da simulação, basta clicar no botão que inicia a simulação, que é o quarto da esquerda para a direita na parte superior da janela. Na mesma região, pode se encontrada uma opção chamada *step*, ou seja, o tamanho dos intervalos de tempo que devem ser transpassados até o fim da simulação. As outras opções são de retrocesso total, retrocesso parcial, parada, adiantamento parcial e total.

Durante a simulação, é possível verificar os fluxos de dados sendo enviados de um nó à outro da rede. Estes fluxos, neste exemplo, possuem as cores vermelha e azul e podem ser observados na figura 10 como tons de cinza.

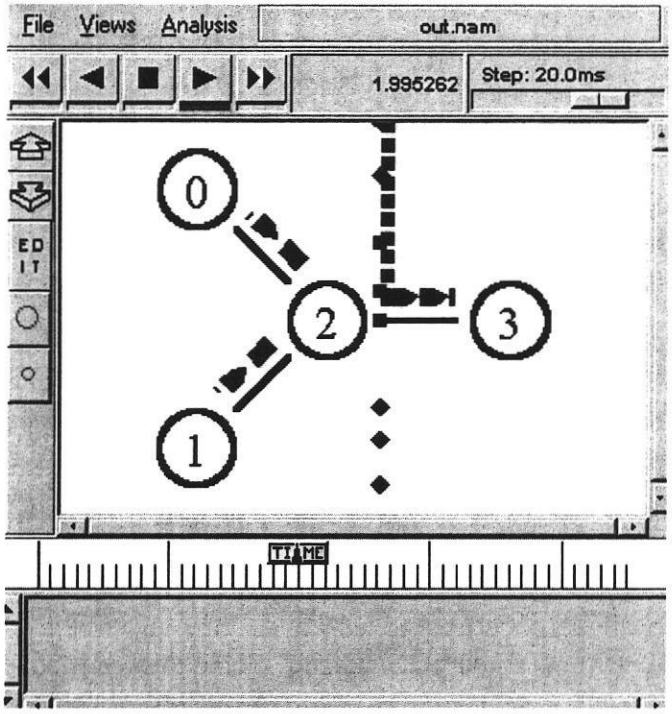

Figura 10. Fluxos de dados na rede de exemplo

Entre os nós 2 e 3, mostrados na figura 10, alguns pacotes são descartados (quadrados caindo na parte mais abaixo do nó 2) e este comportamento deve-se à política de fila especificada para o *link*.

Nas próximas seções são mostrados o formato do arquivo de saída e também algumas ferramentas para a análise deste tipo de arquivo.

7. Formato do arquivo de monitoramento

A ferramenta *NS-2* fornece a possibilidade de usar um arquivo de extensão *tr* para armazenar as informações de cada pacote que foi enviado e recebido pelos nós da rede. No exemplo acima, o nome do arquivo é *arq_trace*, que foi especificado logo no começo do *script tcl*. O arquivo de saída tem o seguinte formato:

1	2	3	4	5	6	7	8	9	10	11	12

1. Tipo do evento
2. Tempo que o evento ocorreu
3. Nó origem
4. Nó destino
5. Tipo de pacote
6. Tamanho do pacote
7. *Flags* (não são sempre usados)
8. Identificação de fluxo
9. Endereço de origem
10. Endereço de destino
11. Número de seqüência do pacote (dado pelo protocolo usado)
12. Identificação única do pacote (em relação aos *links*)

O primeiro campo, tipo do evento, pode conter três tipos de símbolos que dizem o que ocorreu em determinado *link*: o símbolo **+**, que significa que o pacote foi adicionado à fila; o símbolo **-** , que mostra que o pacote foi retirado da fila; o símbolo ***r*** (received), mostrando que o pacote foi recebido na saída do *link*, e o símbolo ***d*** (dropped), expondo que o pacote foi descartado. Um exemplo de saída deste tipo de arquivo pode ser visto na figura 11.

1	**+**	**0.5**	**0**	**2**	**cbr**	**600**	**-------**	**1**	**0.0**	**3.0**	**0**	**0**
2	**-**	**0.5**	**0**	**2**	**cbr**	**600**	**-------**	**1**	**0.0**	**3.0**	**0**	**0**
3	+	0.505	0	2	cbr	600	-------	1	0.0	3.0	1	1
4	-	0.505	0	2	cbr	600	-------	1	0.0	3.0	1	1
5	+	0.51	0	2	cbr	600	-------	1	0.0	3.0	2	2
6	-	0.51	0	2	cbr	600	-------	1	0.0	3.0	2	2
7	**r**	**0.5148**	**0**	**2**	**cbr**	**600**	**-------**	**1**	**0.0**	**3.0**	**0**	**0**
8	+	0.5148	2	3	cbr	600	-------	1	0.0	3.0	0	0
9	-	0.5148	2	3	cbr	600	-------	1	0.0	3.0	0	0
	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.

Figura 11. Amostra da saída do exemplo dado na última seção (arquivo *arq_trace.tr*)

Pela figura acima, é possível perceber os eventos na fila nos *links.* No *link* entre os nós 0 e 2, por exemplo, temos, no tempo de 0.5 segundo, na linha 1, a adição do pacote número 0 à fila, simbolizada pelo símbolo **+**. Também no tempo 0.5 segundo, na linha 2, este mesmo pacote saiu da fila e em 0.5148 (linha 7), o mesmo foi recebido na saída do *link* (símbolo ***r***), ou seja, pelo nó destino. A trajetória deste pacote está representada na figura pelas linhas em destaque.

Na próxima seção, são expostas algumas ferramentas para a análise deste tipo de arquivo.

8. Softwares para a análise dos arquivos de monitoramento

Para a análise dos arquivos de monitoramento (*trace files*), que guardam as informações sobre os pacotes enviados e recebidos pelos nós, existem ferramentas (tipo *open source*) que

realizam esta tarefa. Uma das ferramentas chama-se *Xgraph*, que vem inclusa no arquivo *ns-allinone-2.28.tar.gz* e é para a utilização no ambiente *Linux*. Mais detalhes sobre a ferramenta podem ser obtidos no endereço eletrônico (*http://www.isi.edu/nsnam/xgraph/index.html*).

Uma outra ferramenta que pode se usada chama-se *Trace Graph* e seu uso é tanto para o ambiente *Windows* quanto para o *Linux*. Ela pode ser obtida através do endereço eletrônico (*http://www.geocities.com/tracegraph/*) e também é gratuita. Sua instalação no ambiente *Windows* requer os seguintes componentes:

- *MatLab 6.5* (disponível em *http://diament.ists.pwr.wroc.pl/~tracegr/mglinstaller.exe*)
- *Trace Graph 2.02* para *Windows* (disponível em http://diament.ists.pwr.wroc.pl/~tracegr/tracegraph202.zip)

Primeiramente, é necessário executar o arquivo *mglinstaller.exe* para a instalação do *MatLab*. Depois desta instalação, é necessário o comando `SET PATH=c:\mgl\bin\win32` no console *DOS* do *Windows* (caso o diretório de instalação do *MatLab* seja *mgl*). Logo em seguida, basta descompactar o arquivo *tracegraph202.zip* para o subdiretório `c:\mgl\bin\win32` e para a execução do programa, basta um duplo clique no ícone *tgraph*, este também localizado no subdiretório `c:\mgl\bin\win32`.

A ferramenta *Trace Graph* possui várias alternativas de geração de gráficos e visualização de estatísticas. Entre os gráficos podemos ter somas acumulativas de pacotes, *throughput* em relação aos nós, e muitos outros, além de geração de gráficos *3D* para o número de pacotes gerados em cada nó, pacotes perdidos e outros. Também tem-se a possibilidade da geração de histogramas. A figura 12 mostra um gráfico *3D* sobre os pacotes gerados por cada nó do exemplo utilizado na seção 5.2.

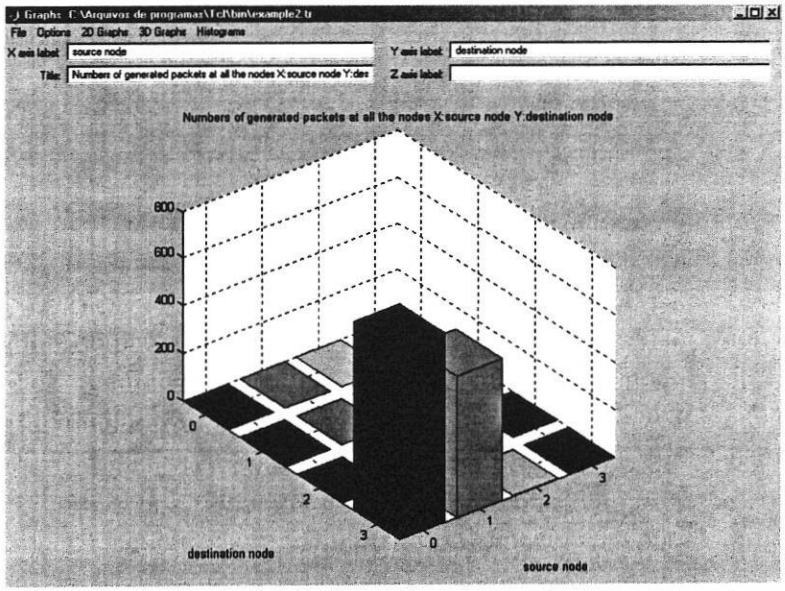

Figura 12. Gráfico 3D gerado pela ferramenta *Trace Graph*

Pela figura 12 é possível perceber que apenas os nós 0 e 1 geraram pacotes para o nó 3. Também é possível perceber que há uma diferença na quantidade de pacotes gerados por

cada um. Este fato é o reflexo dos intervalos de tempo diferentes entre os envios colocados para os agentes *UDP* localizados nos dois nós. Como no nó 0 temos um intervalo de tempo menor (0.005) que o do nó 1 (0.008), o número de pacotes gerados pelo primeiro tende a ser maior.

9. Considerações sobre a ferramenta

A ferramenta *NS-2* mostra-se coerente com sua proposta de proporcionar um ambiente favorável para a construção de simulações de redes de computadores. Seu uso não é de difícil compreensão e se mostra como um grande aliado como forma de aprendizado no ensino de redes de computadores.

Além da simulação de redes mostradas neste relatório técnico, o programa contém suporte para a simulação de outros tipos de redes, como *Wireless* e redes que dependem de conexão com satélites.

Referências Bibliográficas

(Conser, 2005) "*Collaborative Simulation for Education and Research*", Página HTML: http://www.isi.edu/conser/index.html , acessada em Janeiro/2005.

(Ns-2, 2005) "*The Network Simulator HomePage*", Página HTML: http://www.isi.edu/nsnam/ns/, acessada em Janeiro/2005.

(REAL, 2005) "*REAL Overview*", Página HTML: http://www.cs.cornell.edu/skeshav/real/overview.html , acessada em Janeiro/2005.

(SAMAN, 2005) "*SAMAN*", Página HTML: http://www.isi.edu/saman/index.html , acessada em Janeiro/2005.

(VINT, 2005) "VINT Project", Página HTML: http://www.isi.edu/nsnam/vint/index.html , acessada em Janeiro/2005.

(Vargas, 2004) "OMNeT++ User Manual", http://www.omnetpp.org/external/doc/html/usman.php, acessada em Outubro/2004.

Bibliografia Consultada

DEARHAM, N., "*Network Simulator V2 (NS-2)*", Notas didáticas, Arquivo Formato Power Point: http://www.ee.und.ac.za/pcs/files/Seminar2002 - Network Simulator final.ppt, acessada em Janeiro/2005.

GHASSEMIAN, M., "*Network Simulator 2*", Notas didáticas, Arquivo formato Power Point: www.ctr.kcl.ac.uk/teams/ns2/slides/ns2-mona.pdf, acessada em Janeiro/2005.

NANDAKUMAR, K., "*The Network Simulator – NS-2*", Notas didáticas, Arquivo formato PDF: *www.cse.msu.edu/~nandakum/ns2presentation.ppt* , acessada em Janeiro/2005.