**UNIVERSIDADE FEDERAL DE VIÇOSA**
**DEPARTAMENTO DE INFORMÁTICA**
**INF 651 – REDES DE COMPUTADORES**

Lista de Exercícios sobre simulações de redes usando NS2-2

**1-) Argumente sobre as situações em que é conveniente ou indispensável a utilização de um simulador como o NS2. Comente sobre as principais funcionalidades do simulador NS2.**

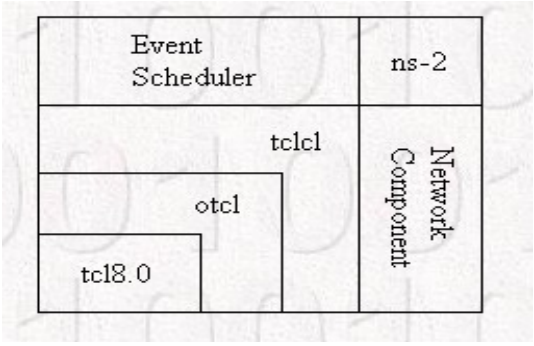
Uma das situações que a utilização do NS2 seria conveniente é na área acadêmica para ensino de Redes e seus conceitos, partindo do princípio que colaboram muito com a união da teoria com a simulação do funcionamento da rede, ou seja, visualizar um conceito é extremamente eficaz, didaticamente, para que alunos de diversos níveis de conhecimento possam compreender realmente como funcionam as Rede de Computadores.

A sua utilização tem mais força na área científica, pois proporciona uma boa infra-estrutura para desenvolver programas com uma grande quantidades de protocolos e tecnologias existentes com a intenção e oportunidade para estudar interações de protocolos em ambiente controlado o que seria ótimo para redução de custos de implementações de ambientes reais.

As principais funcionalidades do NS2 são:

Implementa grande parte da funcionalidade existentena internet, como os protocolos IP, TCP, UDP, TP e HTTP, para protocolos de roteamento como o Session, DV, LS e multicast também implementa abordagens para QoS como, IntServ, DiffServ, MPLS e QoS Routing. Tipos de filas (roteadores) DropTail, CBQ, SFQ, WFQ, DRR e RED. Para comunicação sem fio tipo LAN sem fio, comunicação por satélite, GPRS, etc. dentre outras.

**2-) Pesquise e disserte sobre a arquitetura do NS2? Oriente-se pela figura a seguir.**




Nesta figura um usuário qualquer (não um desenvolvedor do NS) pode estar no canto mais à esquerda e abaixo, planejando e rodando simulações em Tcl utilizando os objetos de simulação das bibliotecas OTcl. Os schedulers de evento e a maioria das componentes de rede são implementados em C++ e disponíveis em OTcl pela ligação que é implementada utilizando tclcl.

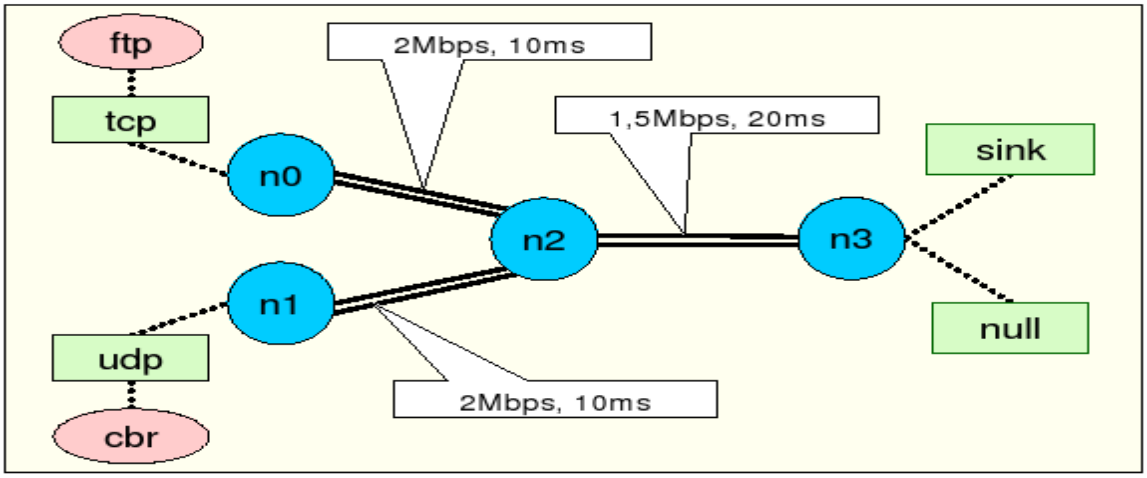
**3-) Quais são os principais componentes do NS2? Comente sobre cada um.**

O interpretador OTCL (Object-oriented Tool Command Language) para uso como frontend. OTCL é uma linguagem interpretada, desenvolvida pelo MIT. Nela serão efetivamente escritas as simulações. O uso da linguagem OTCL, que é interpretada, evita o desgaste por parte do usuário, pois há uma simplificação no processo interativo de mudar e re-executar o modelo.

A linguagem de programação C++, para a estrutura básica (protocolos, agentes, etc) que é uma linguagem compilada e de uso tradicional, mostrou-se a ferramenta mais eficaz.

Um outro componente do NS reside nos objetos de rede do scheduler de evento. Um evento no NS tem um ID de pacote, que é único para o pacote, com um tempo de escalonamento e um ponteiro para o objeto que controla o evento.

Os componentes de redes também fazem parte dos componentes do NS onde a comunicação destes componentes de rede se dá por passagem de pacotes, de forma a não consomir tempo de simulação real.




**4-) O front-end utilizado pelo NS2 é o interpretador tcl8.x. Ou seja, as simulações em NS2 são desenvolvidas a partir de scripts NS2 que retratam o cenário modelado. Dessa forma, pesquise sobre a linguagem de programação tcl8.x e faça um PEQUENO resumo esquemático apontando as principais informações sobre a linguagem, tais como: variáveis, constantes, entrada e saída de dados, estruturas de dados, estruturas de condição e estruturas de repetição.**

1. **Introdução**

- Quando instalares Tcl, o programa que irás chamar pra executar coisas é tclsh. Por exemplo, se escreveste algum codigo em um ficheiro com o nome "hello.tcl", e quiseres executá-lo, tu executarias deste modo: tclsh hello.tcl.
- O ponto classico de um tutorial é o clássico programa "Hello, World". Quando conseguires imprimir uma string prá tela, tás no bom caminho pra usar Tcl para brincar e prófito!
- O comando para mostrar uma string em Tcl é o puts. Um pedaço de texto depois do comando puts irá ser enviado para o stdout. É padrão de enviar um caracter de nova linha ("return") no fim do texto.
- Muitos comandos em Tcl (incluindo o puts) podem aceitar multiplos argumentos. Se uma string não é fechada em quotas ou braçadeiras, o interpretador Tcl vai considerar cada palavra como um argumento separado, e passar cada um individualmente ao comando puts. O comando puts irá tentar avaliar as palavras como argumentos opcionais. O que irá provavelmente causar um erro.
- Um comando em Tcl é uma lista de palavras terminado por uma nova linha ou ponto e virgula ;. Comentários em tcl são iniciados por # no inicio da linha, ou depois de fechar um comando com ponto e virgula;

① Exemplo

```
puts "Hello, World - Em quotas"     ;# Isto é um comentário a seguir ao comando.
# Isto é um comentário no inicio de uma linha
puts {Hello, World - Em braçadeiras}
puts {Mau exemplo de um comentário}# *Erro* - não existe ponto e virgula!
puts "Isto é a linha 1"; puts "Isto é a linha 2"
puts "Hello, World; - Com ponto e virgula dentro das quotas"
# Palavras não precisam de quotas a não ser que contenham espaços entre elas:
puts HelloWorld
```

1. **Variáveis**

- Em Tcl, tudo pode ser representado como uma string, mas internamente representado como uma lista, integer, double, ou outro tipo, em ordem de fazer a linguagem rápida.
  O comando para atribuir valores é o set.

  Quando o set é chamado com 2 argumento tal como:

  **set fruta ananás**

  põe o segundo argumento (ananás) no espaço de memória referente ao primeiro argumento (fruta). Set sempre retorna o conteudo da variavel usada no primeiro argumento. Então, quando set é chamado com 2 argumentos, ele poe o segundo argumento na memória referente ao primeiro argumento e depois retorna o segundo argumento. No exemplo acima, por exemplo, iria retornar "ananás", sem as quotas.

  O primeiro argumento do comando set pode ser uma unica palavra, tipo fruta ou pi, ou pode ser o membro de um array. Arrays iram ser discutidos em maior detalhe mais tarde, para já apenas lembra-te que muita data pode ser posta em uma unica variavel, e que dados individuais podem ser acessados pelo seu index dentro daquele array. Criando indexes em arrays em Tcl é feito usando parenteses depois do nome da variavel.

- **Exemplo**

```
set X "Isto é uma string"

set Y 1.24

puts $X
puts $Y

puts "............................."

set label "O valor em Y é: "
puts "$label $Y"
```

1. **Expressões Matemática**

   O comando do Tcl para fazer o tipo operações do math é expr. A seguinte discussão do comando do expr é extraída e adaptada da página do homem do **expr**.

   O **expr** faz exame de todos seus argumentos (“2 + 2” por exemplo) e avalía o resultado como um Tcl “expressão” (melhor que um comando normal), e retornos o valor. Os operadores permitiram em expressões do Tcl incluem todas as funções do math do padrão, operadores lógicos, bitwise operadores, as well as funções do **math** como a **rand(), sqrt(), cosh()** e assim por diante. As expressões rendem quase sempre resultados numéricos (inteiro ou valores de ponto flutuante).

   **Ponta do desempenho:** incluir os argumentos ao expr em cintas curly resultará em um código mais rápido. Fazer assim o expr {$i * 10} em vez simplesmente do expr $i * 10

2. **Operadores e Operandos**

   Uma expressão do Tcl consiste em uma combinação dos operandos, dos operadores, e dos parênteses. O espaço branco pode ser usado entre operandos, operadores e parênteses; é ignorado pelo processador da expressão. Onde possível, os operandos são interpretados como valores do

inteiro. Os valores do inteiro podem ser especificados no decimal (o caso normal), em octal (se o primeiro caráter do operando é 0), ou no hexadecimal (se os primeiros dois caráteres do operando são 0x).

Nota-se que a conversão octal e hexadecimal ocorre diferentemente no comando do **expr** do que na fase da substituição do Tcl. Na fase da substituição, a **\ x32** seriam convertidos a um ascii "2", quando o expr converteria **0x32 a uns** 50 decimais.

Se um operando não tiver um dos formatos do inteiro dados acima, está tratado então como um número floating-point, se aquele for possível. Os números Floating-point podem ser especificados em algumas das maneiras aceitadas por um compilador de C ANSI-compliant. Por exemplo, todo o seguir é números floating-point válidos:

2.1 - 3.6E4 7.91e+16 .000001
Se nenhuma interpretação numérica for possível, a seguir um operando está deixado porque uma corda (e somente um jogo limitado dos operadores pode lhe ser aplicado). Nota-se entretanto, isso que não suporta números dos seguintes formulários:

2.1 - uma vírgula decimal, em vez de um ponto decimal 2.100 - um separador dos milhares

É possível tratar dos números que o formulário, mas você terá que converter estes "amarra" aos números no formulário padrão primeiramente.

3. **Operadores**

Os operadores válidos são alistados abaixo, agrupado na ordem diminuindo da precedência:

**- + ~!** Menos, Mais, negação, lógico NÃO. Nenhuns destes operadores podem ser aplicados aos operandos da corda, e bit-wise NÃO podem ser aplicados somente aos inteiros.

****** Exponenciação (trabalhos em ambos os números e inteiros floating-point)

***/%** Multiplicar, dividir, restante. Nenhuns destes operadores podem ser aplicados aos operandos da corda, e o restante pode ser aplicado somente aos inteiros. O restante terá sempre o mesmo sinal que o divisor e um valor absoluto menores do que o divisor.

**+ -** Adicionar e subtrair. Válido para alguns operandos numéricos.

**<< >>** Deslocamento esquerdo e direito. Válido para operandos do inteiro somente.

**< > >= do <=** Operadores relacionais: menos, mais grande, menos do que ou igual, e mais grande do que ou igual. Cada operador produz 1 se a circunstância for verdadeira, 0 de outra maneira. Estes operadores podem ser aplicados aos operandos numéricos as well as as cordas, em que a comparação da corda do caso é usada.

**ne do eq no ni**
Comparar duas cordas para a igualdade (eq) ou o desigualdade (ne). e dois operadores para verificar se uma corda estiver contida em uma lista (dentro) ou não (ne). Estes operadores todos retornam 1 (verdadeiro) ou 0 (falso). Usar estes operadores assegura-se de que os operandos estejam considerados exclusivamente como cordas (e listas), não como números possíveis:

**&** Bit-wise E. Válido para operandos do inteiro somente.

**^** Bit-wise exclusive OU. Válido para operandos do inteiro somente.

**|** Bit-wise OU. Válido para operandos do inteiro somente.

**&&** Lógico E. Produz um 1 resultado se ambos os operandos forem non-zero, 0 de outra maneira. Válido para operandos numéricos somente (inteiros ou floating-point).

**||** Lógico OU. Produz uns 0 resultados se ambos os operandos forem zero, 1 de outra maneira. Válido para operandos numéricos somente (inteiros ou floating-point).

**x? y: z** Se-então-outro. Se x avaliar a non-zero, então o resultado é o valor do Y. Se não o resultado é o valor do Z. O operando de x deve ter um valor numérico.

### 4. Estrutura Condicional While

Como a maioria de línguas, o Tcl suporta se comando. A sintaxe é:

- if expr1 ?then? body1 elseif expr2 ?then? body2 elseif ... ?else? ?bodyN? Se a expressão do teste avaliar a falso, então a palavra depois que body1 será examinado. Se a palavra seguinte for elseif, a seguir a expressão seguinte do teste estará testada como uma circunstância. Se a palavra seguinte for outra então o corpo final estará avaliado como um comando.

  A expressão do teste depois da palavra se for avaliado na mesma maneira que no comando do **expr**.

  A expressão do teste que seguem se puder ser incluido dentro das citações, ou cintas. Se for incluido dentro das cintas, será avaliado dentro do se o comando, e se incluído dentro das citações ele for avaliado durante a fase da substituição, e um outro círculo das substituições será feito então dentro do se comando.

- **Exemplo**

```
set x 1

if {$x == 2} {puts "$x is 2"} else {puts "$x is not 2"}

if {$x != 1} {
    puts "$x is != 1"
} else {
    puts "$x is 1"
}

if $x==1 {puts "GOT 1"}

#
# Be careful, this is just an example
# Usually you should avoid such constructs,
# it is less than clear what is going on and it can be dangerous
#
set y x
if "$$y != 1" {
    puts "$$y is != 1"
} else {
    puts "$$y is 1"
}

#
# A dangerous example: due to the extra round of substitution,
```

```
# the script stops
#
set y {[exit]}
if "$$y != 1" {
    puts "$$y is != 1"
} else {
    puts "$$y is 1"
}
```

O comando **switch** permite que você escolha uma de diversas opções em seu código. É similar ao **switch** em C, exceto que é mais flexível, porque você pode ligar strings, em vez dos inteiros justos. A string será comparada a um jogo dos testes padrões, e quando um teste padrão combina a string, o código associado com esse teste padrão será avaliado.

É uma idéia boa usar o comando **switch** quando você quer combinar uma variável de encontro a diversos valores possíveis, e não quer fazer uma série longa de if… indicações do elseif do elseif….

A sintaxe do comando é:

- switch string pattern1 body1 ?pattern2 body2? ... ?patternN bodyN?

- ou -

- switch string { pattern1 body1 ?pattern2 body2?...?patternN bodyN? }

- **Exemplo**

```
set x "ONE"
set y 1
set z ONE

# This is probably the easiest and cleanest form of the command
# to remember:
switch $x {
    "$z" {
        set y1 [expr $y+1]
        puts "MATCH \$z. $y + $z is $y1"
    }
    ONE {
        set y1 [expr $y+1]
        puts "MATCH ONE. $y + one is $y1"
    }
    TWO {
        set y1 [expr $y+2]
        puts "MATCH TWO. $y + two is $y1"
    }
    THREE {
        set y1 [expr $y+3]
        puts "MATCH THREE. $y + three is $y1"
    }
    default {
        puts "$x is NOT A MATCH"
    }
}
```

## 5. Laços de Repetição

O Tcl inclui dois comandos para dar laços, o quando e para comandos. Como se a indicação, eles tem a função de avaliar seu teste a mesma maneira que o **expr**. Nesta lição nós discutirmos o comando **while**, e na lição seguinte, o comando **for**. Na maioria das circunstâncias onde um destes comandos pode

ser usado, o outro pode ser usado também.

**while test body**

O comando **while** avalia o teste como uma expressão. Se o teste for verdadeiro, o código no corpo será executado. Depois que o código no corpo foi executado, o teste avalia outra vez.

Uma indicação **continue** dentro do corpo parará a execução do código e o teste será reavaliado. Um **break** dentro do corpo quebrará fora do laço **while**, e a execução continuará com a linha seguinte do código após o corpo

- **Exemplo**

```
set x 1

# This is a normal way to write a Tcl while loop.

while {$x < 5} {
    puts "x is $x"
    set x [expr $x + 1]
}

puts "exited first loop with X equal to $x\n"

# The next example shows the difference between ".." and {...}
# How many times does the following loop run?  Why does it not
# print on each pass?

set x 0
while "$x < 5" {
    set x [expr $x + 1]
    if {$x > 7} break
    if "$x > 3" continue
    puts "x is $x"
}

puts "exited second loop with X equal to $x"
```

6. **Estrutura Condicional for**

O Tcl suporta uma construção iterada do laço similar ao laço **for** no C. Para o comando no Tcl faz exame de quatro argumentos; uma iniciação, um teste, um incremento, e o corpo do código a avaliar em cada passagem através do laço. A sintaxe para para o comando é:

**for start test next body**

- **Exemplo**

```
 for {set i 0} {$i < 10} {incr i} {
    puts "I inside first loop: $i"
}
```

7. Variável Global e upvar

O comando global fará com que uma variável em um espaço local (dentro de um procedimento) consulte à variável global desse nome.

O comando upvar é similar. “Amarra” o nome de uma variável no espaço atual a uma variável em um espaço diferente. Isto é usado geralmente

simular a pass-por-referência aos  procs.

Normalmente, o Tcl usa um tipo do “de referência chamada da coleção lixo” que conta a fim limpar automaticamente acima das variáveis quando não são usados mais, como quando vão “fora do espaço” no fim de um procedimento, de modo que você não tenha que se manter a par d você mesmo. É também possível ao explicity unset os com nomeado aptly unset o comando.

A sintaxe para upvar é:

**upvar ?level? otherVar1 myVar1 ?otherVar2 myVar2? ... ?otherVarN myVarN?**

- **Exemplo**

```
  proc SetPositive {variable value } {
    upvar $variable myvar
    if {$value < 0} {
        set myvar [expr -$value]
    } else {
        set myvar $value
    }
    return $myvar
}

SetPositive x 5
SetPositive y -5

puts "X : $x    Y: $y\n"

proc two {y} {
    upvar 1 $y z                ;# tie the calling value to variable z
    upvar 2 x a                 ;# Tie variable x two levels up  to a
    puts "two: Z: $z A: $a"     ;# Output the values, just to confirm
    set z 1                     ;# Set z, the passed variable to 1;
    set a 2                     ;# Set x, two layers up to 2;
}

proc one {y} {
    upvar $y z                  ;# This ties the calling value to variable z
    puts "one: Z: $z"           ;# Output that value, to check it is 5
    two z                       ;# call proc two, which will change the value
}

one y                           ;# Call one, and output X and Y after the call.
puts "\nX: $x  Y: $y"


proc existence {variable} {
    upvar $variable testVar
    if { [info exists testVar] } {
        puts "$variable Exists"
    } else {
        puts "$variable Does Not Exist"
    }
}

set x 1
set y 2
for {set i 0} {$i < 5} {incr i} {
    set a($i) $i;
}

puts "\ntesting unsetting a simple variable"
```

```
# Confirm that x exists.
existence x
# Unset x
puts "x has been unset"
unset x
# Confirm that x no longer exists.
existence x
```

**5-) Quais são os passos para desenvolver uma simulação com o NS2? Dê um exemplo.**

Para simplificar o processo de criação de uma simulação, existem alguns passos básicos para se chegar a bons resultados.

O primeiro passo é o Planejamento da Simulação, que significa planejar em forma de rascunho toda a estrutura da simulação, desde os agentes de transporte, formulando as aplicações e todo o mapeamento físico da topologia precisa ser representado. A noção de tempo no NS é obtida através de unidades de simulação que podem ser associadas em segundos, para melhor compreensão. Após definir o esquema do que se pretende colocar na simulação, formular e desenhar tudo se torna mais simples para entender o andamento do projeto.

O segundo passo é definir os nós. Serão descritos os comandos necessários para a definição dos nós. Para se criar um Nó utiliza-se o comando:

```
set <nome do nó> [$ns node]
set no1 [$ns node].
```

Também é possível criar um laço de nós e gerar automaticamente uma quantia necessária para a simulação.

Na seqüência é necessária a definição dos enlaces da rede, ou seja, criar as ligações entre o nós, ou links como são mais conhecidos. Para a criação de um enlace será apresentado um exemplo com as explicações comentadas com #, em negrito sentença do comando [CMM]:

```
# Tipo do link Simplex ou Duplex
$ns duplex-link
# Nó de origem e nó de destino, velocidade do nó em megabits por segundo, atraso do link em milisegundos, política de fila
$no1 $no2       1Mb 10ms DropTail
```

O próximo passo é definir o tráfego entre os nós. A definição do tráfego no NS é algo um pouco mais trabalhoso, no entanto, para que se possa concluir essa definição dois itens são requeridos no planejamento:

a-) O agente ou protocolo de transporte que irá conduzir os pacotes – os mais utilizados são o TCP e o UDP, mas o NS também oferece suporte ao RTP.

b-) O tipo de aplicação que fará o uso desse transporte – como exemplo de aplicações que farão o uso do transporte pode-se citar as aplicações de vídeo contínuo que são do tipo CBR (Constant Bit Rate), as aplicações em FTP (File Transfer Protocol) que são utilizadas em Download e HTTP ( Hyper Text Transfer Protocol), para navegação em WEB[TAS97].

Para exemplificar a definição de tráfego serão exibidos a seguir os comandos [CMM], em forma de comentário está a explicação:

```
# Definição da cor que representará o tráfego
$ns color 7 orange
  # Valor atribuído à instância de transporte, e tipo de protocolo de transporte
Set tcp1   [new Agent/TCP/Newreno]
# Atribuição do fluxo FID (Flow Id Identification)
$tcp1 set class_7
 $tcp1 set fid_1
# Definição do tamanho da janela do protocolo TCP
$tcp1 set windows_2000
# Criação do agente sink que receberá os dados
 set sink1   [new Agent/TCPSink]
# Atribuição do nó onde será conectado o fluxo para efeito de transmissão
 $ns attach-agent $transmissor1       $tcp1
# Nó onde será recebido o fluxo
```

```
$ns attach-agent $receptor1         $sink1
# Definição da conexão do protocolo com o nó de recepção
$ns connect $tcp1      $sink1
# Tipo de aplicação
Set ftp1 [$tcp1 attach-source FTP]
```

**6-) Pesquise e disserte sobre a ferramenta NAM. Quais são as principais funcionalidades e objetivos de uso.**

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.

A ferramenta NAM executa uma animação do sistema sendo simulado, apresentando graficamente uma ilustração gráfica da simulação, mostrando pacotes sendo enviados nos canais, crescimento das filas, perdas de pacotes e monitoramento de pacotes.

Também faz uso de Arquivos de *Trace como,* Resultados de simulação, Dados de redes reais e Arquivo construído manualmente, faz parte da distribuição básica do NS com o principal objetivo de apenas mostrar o resultado de simulação já realizada.

**7-) Pesquise e comente sobre a ferramenta Xgrapf e gnuplot. Como o Xgrapf e gnuplot pode contribuir para analisar estatísticas do NS2.**

O Xgraph e o gnuplot podem contribuir para analisar as estatísticas do NS2 pois são ferramentas de geração de gráficos que vem junto com a distribuição do NS com o objetivo de Plotagem de gráficos e visualização interativa para gerar a animação.

Trabalham em cima de arquivos com formato colunado, ou seja, uma série de linhas com pares (x, y).

O GNUPLOT é um aplicativo de domínio público, destinado à construção de gráficos e superfícies. É uma poderosa ferramenta, usada por uma série de organizações por todo o mundo, como pode-se ver a partir dos links disponíveis na página oficial.

Uma característica importante deste aplicativo é o fato de se ter arquivos binários para diferentes sistemas operacionais, possibilitando que um arquivo *script* seja executado em diferentes plataformas.

**8-) Os arquivos de trace são os dados resultados da execução da simulação de um cenário, que são guardados em arquivos. O formato básico de um arquivo de trace é:**


**É praticamente impossível visualizar o arquivo e identificar alguma informação interessante. Dessa forma, é necessário utilizar ferramentas auxiliares como perl, shell, awk o próprio tcl para retirar informações interessantes do arquivo. Um exemplo seria a seguinte instrução em shell: cat out.tr | grep ^d | while read a b c d e f g;do echo $a $b $e $f; done >> saida.txt**

**Este comando lê do arquivo trace out.tr que contém todos os dados de uma simulação qualquer, a informação de quais pacotes foram dropados, em que momento, qual era o protocolo e o tamanho do pacote. Faça um teste! Em anexo a lista de exercícios, existe um arquivo com o nome de out.tr.**

**Execute a instrução shell citada anteriormente sobre o out.tr, basta você está posicionado no mesmo diretório onde o arquivo out.tr se encontra. Veja como as informações são filtradas. Pratique: Utilize a ferramenta awk para consultas de algumas informações sobre o arquivo out.tr:**

**a-) No intervalo de tempo de 3 a 4 segundos, liste em que fração do intervalo de tempo os pacotes foram enfileirados.**
**R:** {if ($1 == "+") {if ($2 >= 3 && $2 <= 5) print $1 $2}} o problema é que eu não entendi se são os tempos 3 e 4 ou se o tempo 4 não entra. Se não entra então fica assim:
**R:** {if ($1 == "+") {if ($2 >= 3 && $2 < 4) print $1 $2}}

**b-) Liste quantos pacotes foram recebidos.**
{if ($1 == "r") total = total + 1} END {print total}
**R:** 15430 pacotes

**c-) Informe o tamanho e o intervalo de tempo de pacotes desenfileirados.**
**R:** {if ($1 == "-") {print $6 $2}}

**d-) Faça um somatório do tamanho dos pacotes que foram dropados.**
**R: Total**: 485008
{if ($1 == "d") total = total + $6} END {print total}

os scrips foram gerados em arquivos separados.

**9- CRIATIVIDADE: Crie um cenário de rede para ser modelado, que envolva hosts(nodos), links(canal de ligação entre os nodos), tipos de enlace(com fio e sem fio), defina a largura de banda dos links, o delay, o tipo de enfileiramento, protocolos de transporte, fontes, destinos, aplicações e período de simulação. O objetivo deste exercício é tentar modelar um típico cenário do mundo real em um ambiente de simulação utilizando o NS2.**

**Faça um esboço em papel inicialmente com todas as definições, em seguida escreva o script tcl de simulação, observe a simulação e faça comentários. Procure monitorar o enfileiramento, a banda passante e atrasos e o que mais julgar necessário. Utilize o NAM e o XGRAPF para acompanhar a simulação e verificar as estatísticas.**

Este exemplo trata-se dois nós enviando dados através de um agente UDP do nó n0 para o n1. O nó n0 envia dados para o nó n1 através da rede, neste script o tráfego foi projetado para que os fluxos de dados sejam de cores diferentes para o que sai de n0 seja azul, o resultado da simulação é apresentada na Figura 1.1.

Não há problemas neste script, pois trata-se de uma aplicação simples, com isso, não existe perda de pacotes nem congestionameto da rede, será descrito a seguir ao mostrar as figuras da simulação do mesmo, linhas com # comentadas para entendimento do programa. O arquivo com o script consta nos anexos do trabalho.

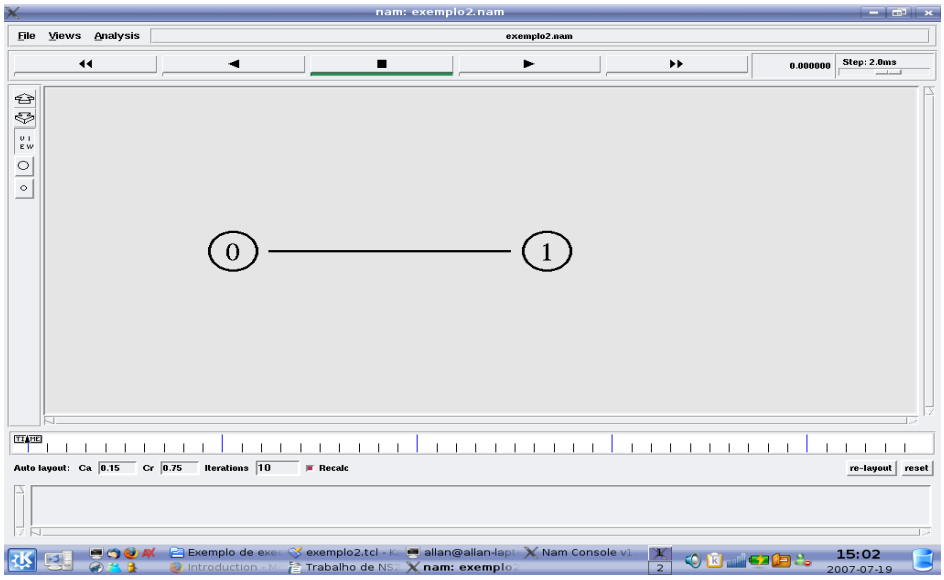



Figura 1.1 – topologia da rede.

A figura 1.2 mostra como o tráfego de pacotes do protocolo de transporte UDP.

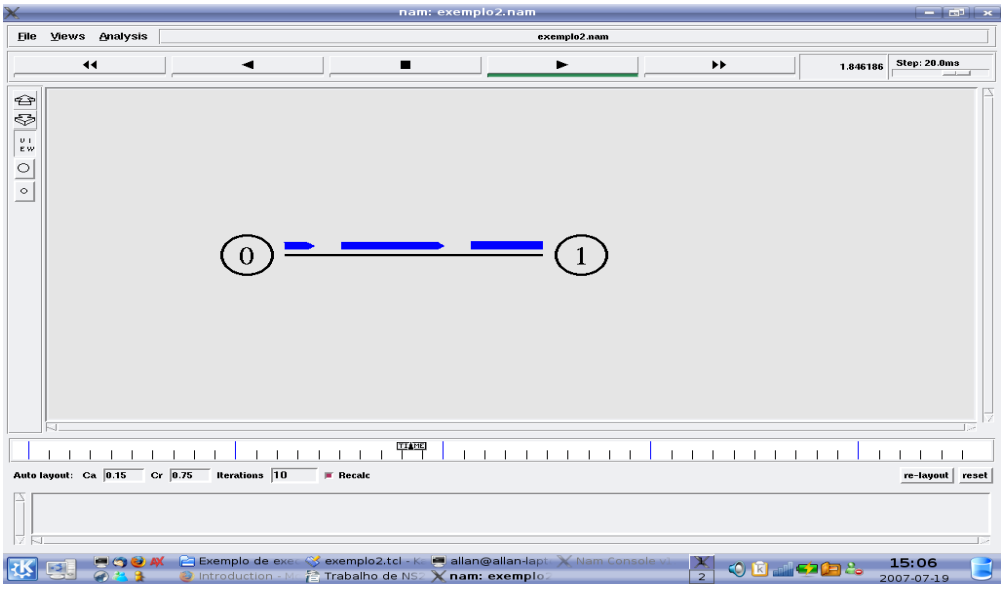



Figura 1.2 – trafego UDP.

plotagem do gráfico com o xgraph.

Figura 1.3 Plotagem do gráfico

1. Trabalho prático sobre simulações de redes usando NS2-2

O objetivo deste trabalho é utilizar o simulador para uma rede simples e analisar o seu comportamento em face de mudanças nas fontes de dados.

Sempre que necessário recorra ao tutorial e ao manual do ns-2 (que podem ser obtidos a partir da página www.isi.edu/nsnam/ns).

**Cenários:**

- Crie um script TCL para a realização de uma simulação no ns-2 com a seguinte especificação:

**Cenário 1**

- 6 nós, n0, n1, n2, n3, n4, n5;
- 6ligações: (n0, n1), (n1, n2), (n1, n3), (n1, n4), (n4, n5) e (n2,n5);
- Utilize recursos para que a monitoração da simulação visualize o cenário da seguinte forma no Nam:
- As ligações entre os links deverão ter as seguintes diretrizes:

  - largura de banda:
    - (n0, n1): 2Mb;
    - (n1, n2): 1Mb;
    - (n1, n3): 2Mb;
    - (n1, n4): 1Mb;
    - (n4, n5): 2Mb;
    - (n2,n5): 2Mb

  Atrasos:
    - (n0, n1): 10ms;
    - (n1, n2): 20ms;
    - (n1, n3): 10ms;
    - (n1, n4): 20ms;
    - (n4, n5): 10ms;
    - (n2, n5): 10ms;

- Todos os link a priori devem ter disciplina de serviço Drop Tail.
- Crie uma ligação TCP com a fonte no nó n0 e o destino no nó n5;
- Associe uma fonte de tráfego FTP ao agente TCP no nó n0 com destino no nó n5;
- Inicie a fonte FTP no instante 0,5s e desligue a fonte no instante 9s;
- Termine a simulação no instante 10seg;
- Monitore a fila de espera no nó n1 para a ligação (n1, n2);
- Crie traces de saída do tipo trace-all e namtrace-all
- Execute a simulação e acompanhe a visualização da mesma através do Nam;

- Manipule o arquivo de saída com dados tipo trace-all de tal forma que você consiga saber **quantos pacotes foram dropados e em que momento da execução da simulação**. Utilize ferramentas do tipo shel script ou awk para manipulação do arquivo de saída;
- Trace um gráfico das **larguras de banda ocupadas** pelas fontes, plote o gráfico com xgrapf ou gnuplot. Produza um arquivo de relatório da simulação deste cenário, coloque no relatório, textos argumentando sobre a simulação do cenário em questão, coloque screenshots da simulação com o Nam e da plotagem das larguras de banda com Xgraph ou gnuplot.

Manipulando o arquivo de trace-all e acompanhando o progresso da simulação pelo NAM nota-se que não há perda de pacotes ou seja, não existe arquivos dropados.

Este exemplo trata-se seis nós enviando dados através de um agente TCP com uma aplicação FTP para um único nó entretanto faz uso da rede para alcançar esse determinado nó o envio é do nó n0 para o o nó n5. O nó n0 envia dados para o nó n5 através dos de n1 e n2, neste script o tráfego foi projetado para que os fluxos de dados sejam de cores diferentes para o que sai de n0 azul e a resposta no caso o sink também seja azul, o resultado da simulação é apresentada na Figura 1.1. Existe um problema neste script que acarreta em congestionamento da rede, que será descrito a seguir ao mostrar as figuras da simulação do mesmo, linhas com # comentadas para entendimento do programa.

Após a criação do script, foi executada a simulação cuja topologia é apresentada na Figura 2.1.

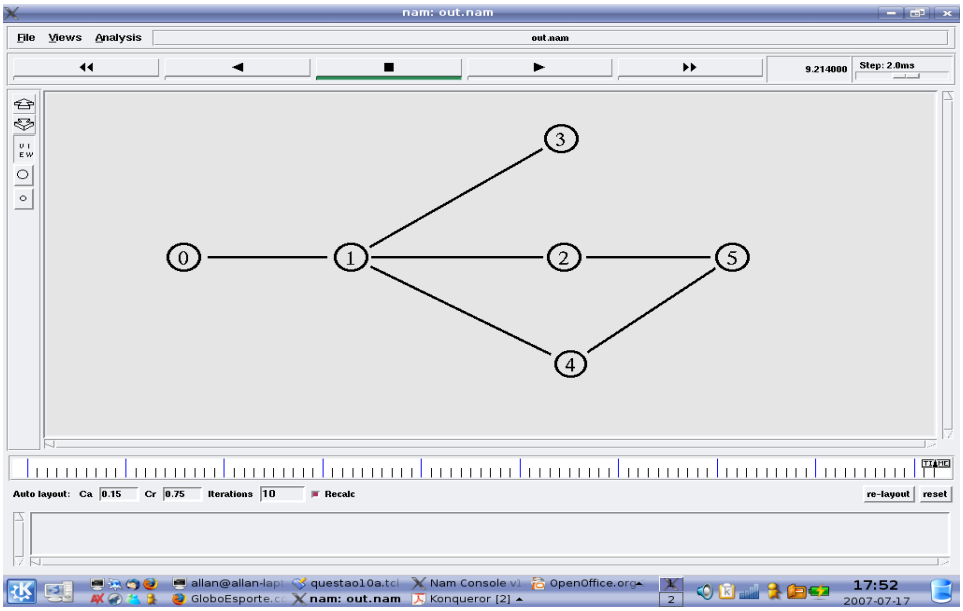



Figura 2.1- Topologia dos Nodos

Esta simulação mostra um problema de sobrecarga no nó 1. Nota-se que os links criados são de dimensionamentos diferentes. Possuem largura de banda, atraso e política de escalonamento de filas iguais, 1Mb e 2Mb, 10ms e 20ms mas todos implementam a mesma política de filas DropTail. E o tráfego entre os dois nodos é variado pois, do n0 p/ n1 a largura de banda é de 2Mb e o atraso é de 10ms mas quando sai do n1 p/ n2 a largura de banda diminui sendo de 1Mb e o tempo de atraso aumenta para 20ms é neste momento o grande problema uma solução seria aumentar a largura de banda entre esses nodos e diminuir o atraso, quando o trafego passa por esses nodos alcança o no n2 / n5 note que neste momento não há sobrecarga na rede. A figura 3.2 mostra este problema.

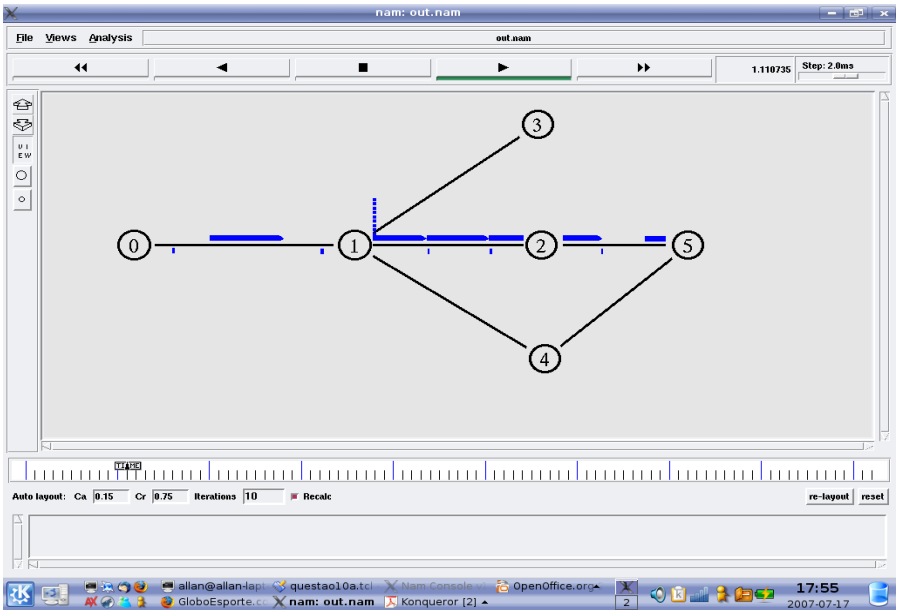


Figura 2.2 – Congestionamento no nó n1

Figura 2.3 – plotagem sobre política de fila DropTail

**Cenário 2**

Faça expansão do arquivo que contém o script do cenário 1, vamos alterar algumas configurações e monitora-las durante a simulação.

Troque a disciplina de serviços para:

- (n0, n1): Drop Tail;
- (n1, n2): SFQ;
- (n1, n3): Drop Tail;
- (n1, n4): SFQ
- (n4, n5): Drop Tail;
- (n2, n5): Drop Tail;

Adicione cores diferentes para cada tráfego criado;

1. Adicione um agente UDP ao nó n0 e uma fonte CBR (Taxa de transmissão = 2Mbps, tamanho do pacote = 512 bytes); o sink correspondente deverá ser colocado no nó n5;
2. Inicie esta fonte CBR no instante 3s e desligue-a no instante 7s;

3. Adicione um agente UDP e uma fonte CBR (Taxa de transmissão = 2Mbps, tamanho do pacote = 512 bytes) ao nó n3; o sink correspondente deverá ser colocado no nó n5;
4. Inicie esta fonte CBR no instante 4s e desligue-a no instante 7s;
5. Monitore a fila de espera no nó n1 para a ligação (n1, n2);
6. Execute a simulação e acompanhe a visualização da mesma através do Nam;

- Manipule o arquivo de saída com dados tipo trace-all de tal forma que você consiga saber quantos pacotes foram dropados e em que momento da execução da simulação. Utilize ferramentas do tipo shel script ou awk para manipulação do arquivo de saída;
- Trace um gráfico das larguras de banda ocupadas pelas fontes, plote o gráfico com xgrapf ou gnuplot.
- Produza um arquivo de relatório da simulação deste cenário, coloque no relatório, textos argumentando sobre a simulação do cenário em questão, coloque screenshots da simulação com o Nam e da plotagem das larguras de banda com Xgraph ou gnuplot.

Foram dropados 2484 pacotes.

Este cenário trata-se seis nós enviando dados através de um agente TCP do nó no para o n5 com uma aplicação FTP atachada e também implementa dois agentes de transporte UDP o primeiro do n0 para o n5 e o segundo do n0 para o n3. A topologia foi projetada para que o trafego esteja com cores diferentes nó n0 envia dados para o nó n5 sendo o agente TCP a cor é azul do mesmos nodos mas com o agente UDP a cor está setada para vermelha e dos nodos n0 para o n3 a cor está setada para verde, o resultado da simulação é apresentada na Figura 1.1. Existem vários problemas nesta simulação que acarreta em congestionamento da rede e perda de pacotes, que será descrito a seguir ao mostrar as figuras da simulação do mesmo, linhas com # comentadas para entendimento do programa.

Após a criação do script, foi executada a simulação cuja topologia é apresentada na Figura 3.1.

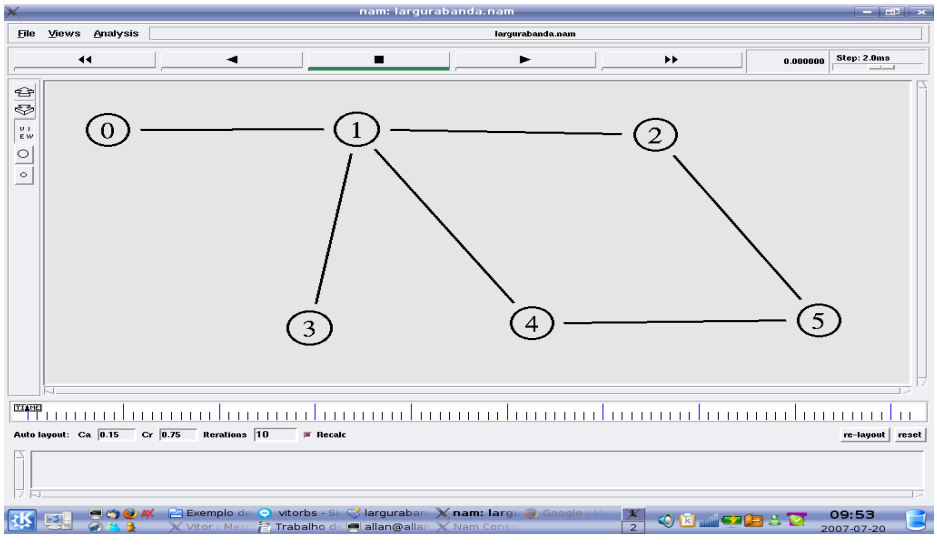
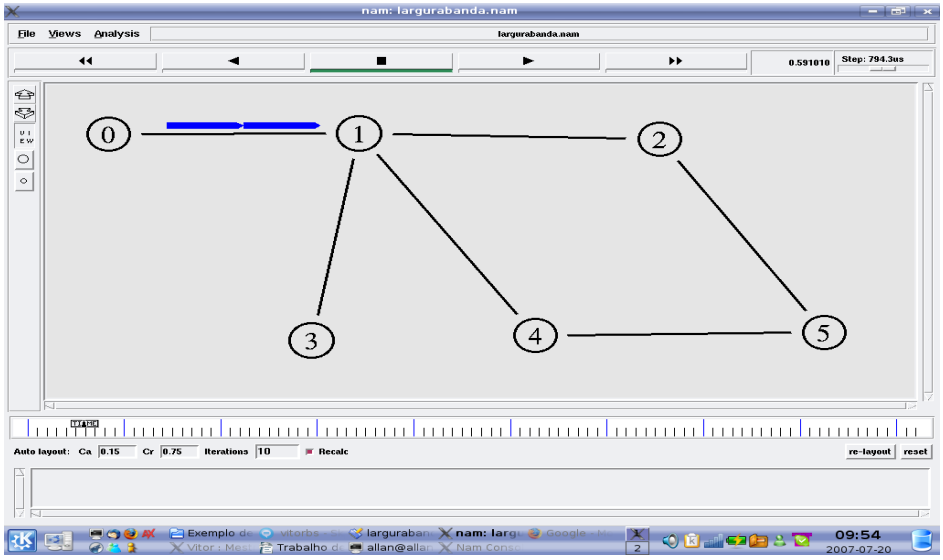



Figura 3.1 – Mostra a Topologia da rede.

A próxima figura mostra a execução no tempo 0.5 o começo do tráfego TCP na rede que vai do nó n0 até o n5.




Fifura 3.2 – Início do agente TCP

Note na figura 3.2 acima o real funcionamento do início dos envios de pacotes TCP na rede sem congestionamento, veja como ele primeiro manda logo após a confirmação apenas dois pacotes mas depois a janela de envio cresce para 4, 8, 16, 32 e 64kb que é o limite de crescimento da janela TCP.

A próxima figura mostra o gargalo que existe entre as conexões já que entre os nodos n0 e n1 a largura de banda máxima é de 2Mb e o atraso é de 10ms e do n1 para o n2 a largura de banda é de 10Mb e o atraso é de 20ms por isso é gerado um gargalo na transmissão causando um congestionamento mas sem perda de pacotes.

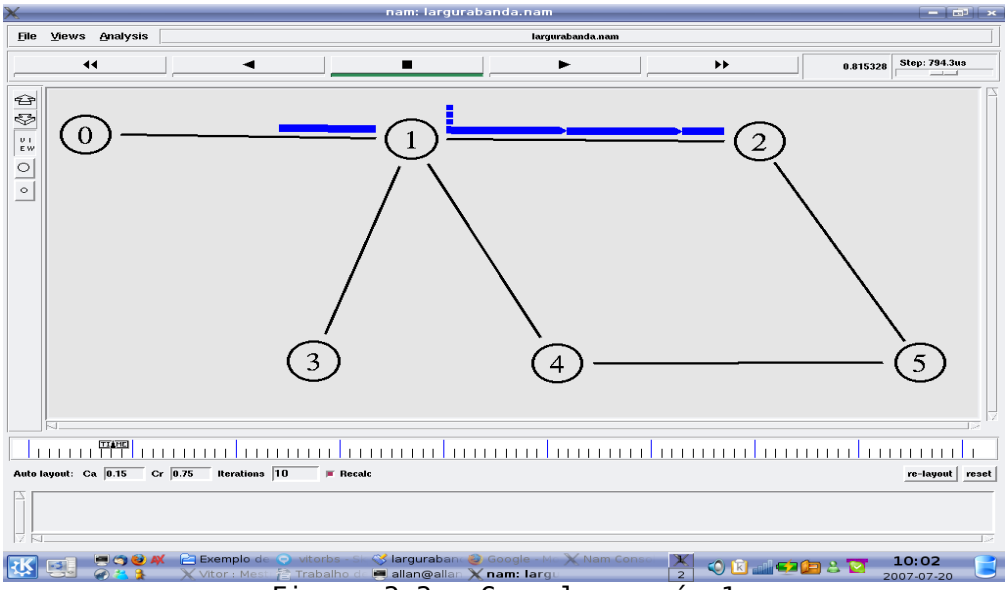



Figura 3.3 – Gargalo no nó n1

A próxima figura 3.4 mostra a execução no tempo 0.7, pois existe um atraso de 2 segundos de um agente UDP em cor preta com o objetivo de monitorar o trafego na rede que vai do n5 até o n0 para obter a largura de banda e gerar as estatísticas e a continuação do tráfego TCP na rede que vai do nó n0 até o n5.

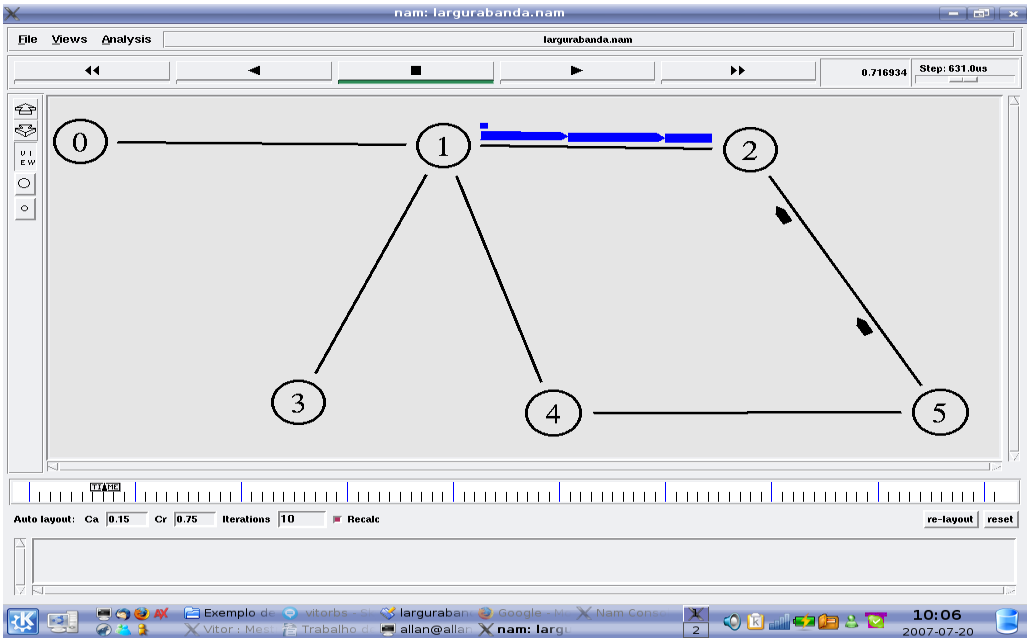



Figura 3.4 – Início do monitoramento da rede

No mesmo tempo 0.7 mas com um pequeno atraso é lançada a rede o outro agente UDP de monitoramento da rede com o intuíto de capturar a largura de banda entre as fontes agora saindo do n3 para o n0. Como mostra a figura 3.5.

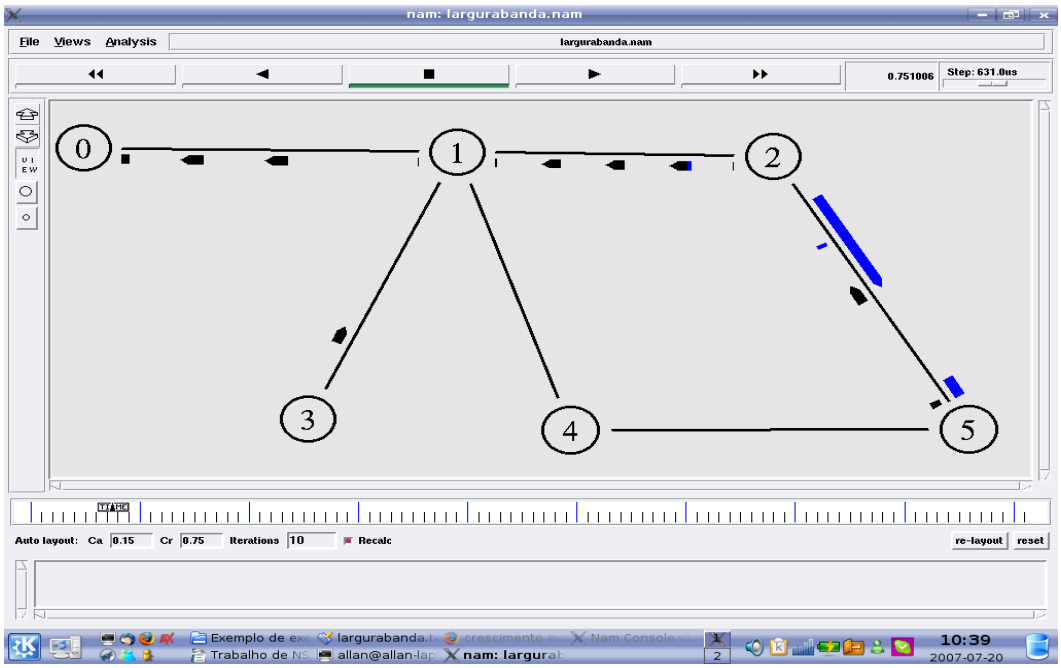



Figura 3.5 – Disparo de outro agente UDP do n3

Note nesta figura 3.6 que é disparado o agente UDP em vermelho do nó n0 para o n5 e também outro agente UDP de monitoramento do nó n4 para o n0, no momento não existe a perda de pacotes, somente o gargalo no nodo n1.

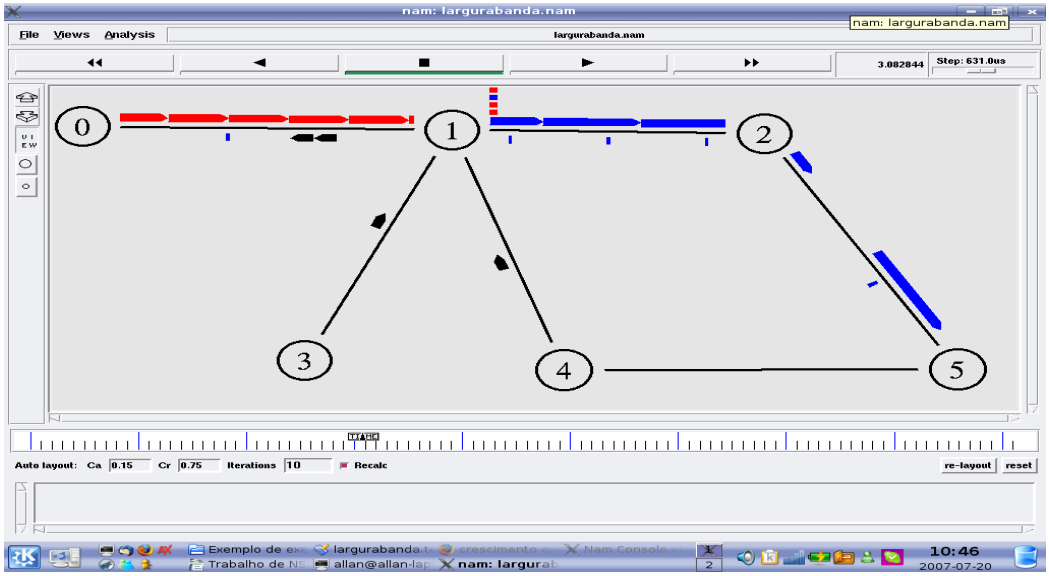



Figura 3.6 – Início do agente UDP em vermelho na rede

Nesta próxima figura 3.7 começa a perda dos pacotes em função do alto tráfego gerado entre o n0 e o n1 pois o n0 está enviando tanto pacotes utilizando o protocolo TCP como também o UDP e como a largura de banda é menor e o atraso maior entre esses dois nodos acarreta em perda de pacotes.

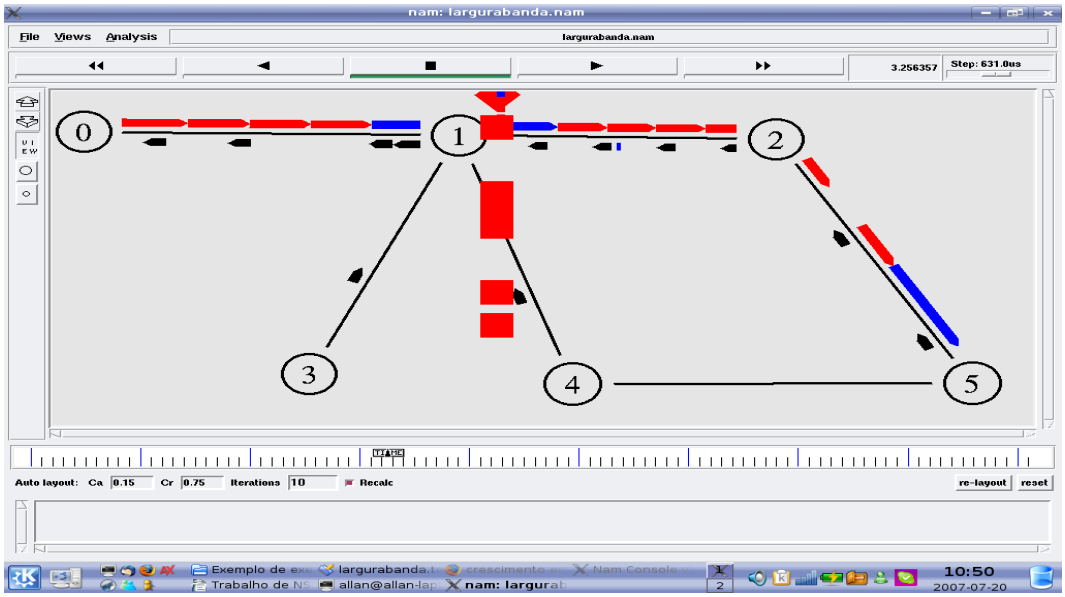



Figura 3.7 – Perda de pacotes

Já na figura 3.8 a perda de pacotes não é só no nodo n1 como também no nodo n0, como há perda de pacotes tanto TCPs como UDPs no caso do TCP que estava no início com janela de 64kb, volta a ser uma janela de 2kb crescendo caso obtenha uma resposta do n5 mas o crescimento diminui pela metade a partir de 32kb e assim crescendo de 2kb em 2kb se não houver perda até atingir uma janela de 64kb.

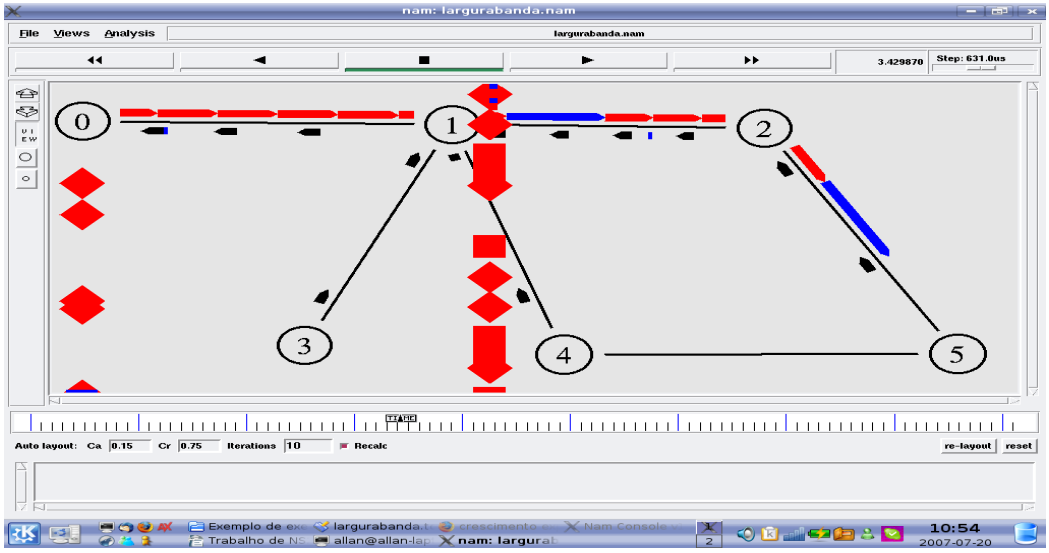



Figura 3.8 – Pacotes dropados no nodo n0

Nesta Última figura 3.9 o agente UDP em verde é disparado no tempo 4s que sai do n3 até o n5.

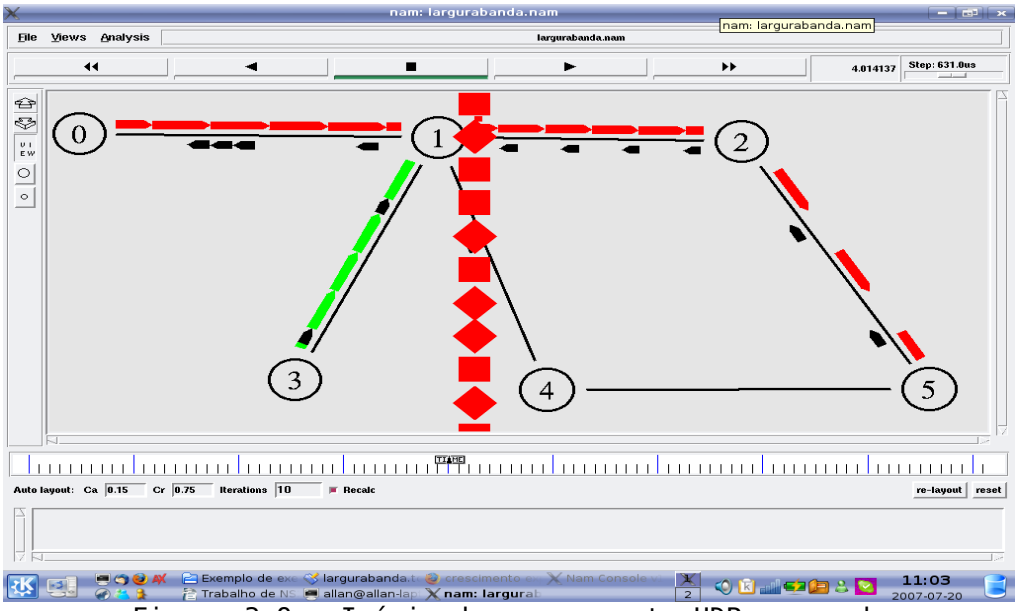



Figura 3.9 – Início do novo agente UDP em verde

Esta figura 4.0 mostra o descarte de pacotes UDP do n3.

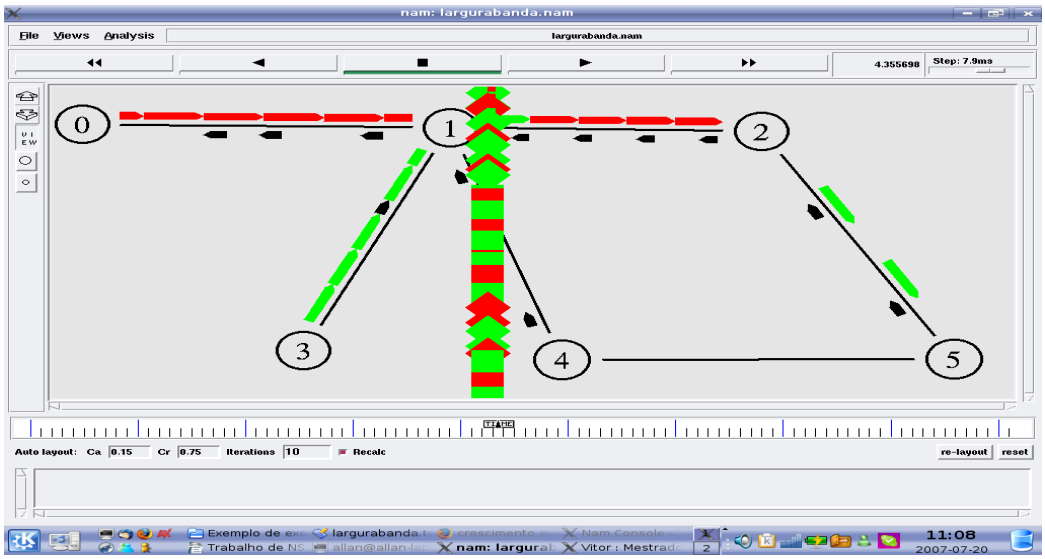



Figura 4.0 – Descarte de Pacotes.

A próxima figura 4.1 mostra a plotagem do gráfico no xgraph amostrado a largura de banda no cenário 2.

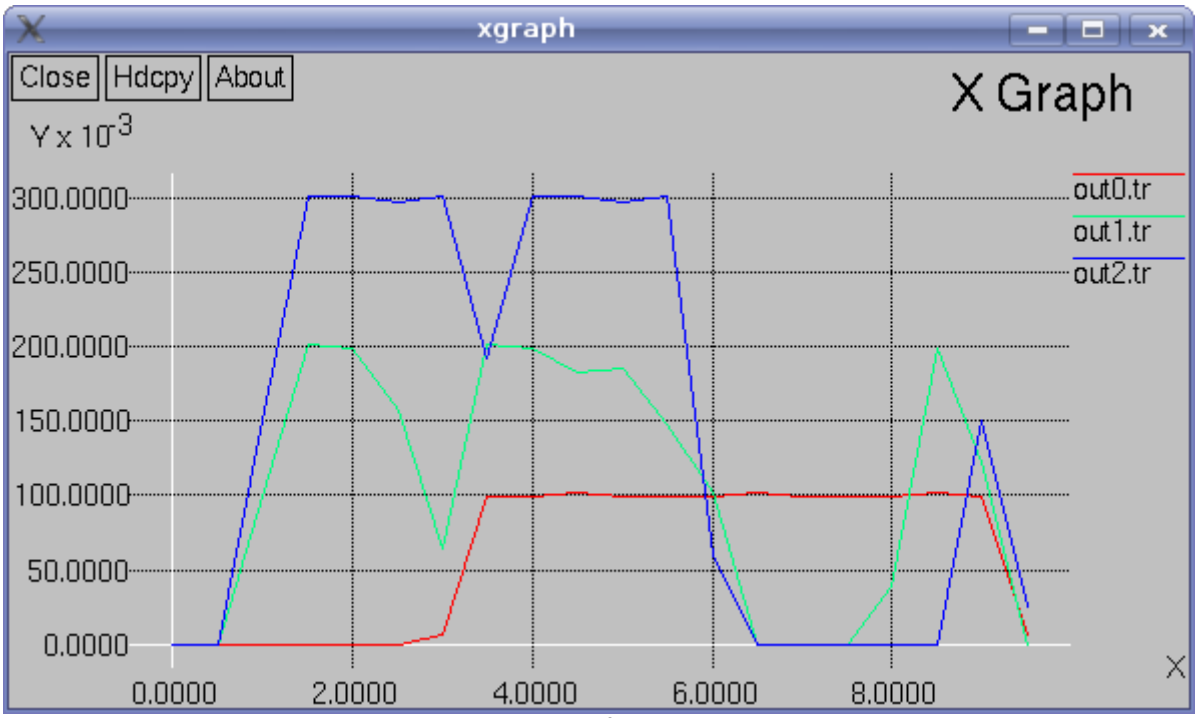



Figura 4.1 Plotagem do gráfico da largura de banda

**Cenário 3**

- Estenda agora o script do cenário 2 e acrescente:
- Associe 'loss monitors' às fontes de tráfego CBR;
- Trace um gráfico das larguras de banda ocupadas pelas fontes;
- Altere a disciplina de serviço na ligação (n1, n2) para SFQ; observe o comportamento no nam; Trace novamente os gráficos da largura de banda e compare com os que obteve anteriormente; Comente os

resultados;

- Repita o ponto anterior, mas com uma disciplina de serviço RED.
- Produza um arquivo de relatório da simulação deste cenário, coloque no relatório, textos argumentando sobre a simulação do cenário em questão, coloque screenshots da simulação com o Nam e da plotagem das larguras de banda com Xgraph ou gnuplot.

No cenário 3 os graficos gerados pelo nam são semelhantes ao do cenário 2 mas colocarei aqui os gráficos gerados pelo xgraph para as políticas de fila diferentes perceba que não há diferença do gráfico anterior.

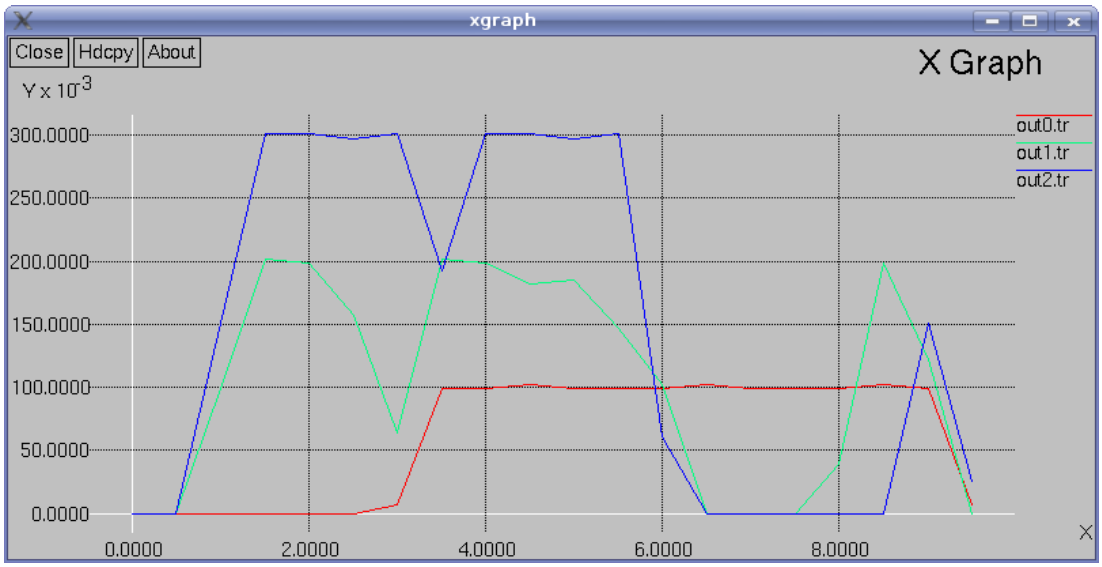



Figura 5.0 – Plotagem do cenario 3

Agora o gráfico gerado com a política de filas RED, neste cenário a mudança é expressiva.

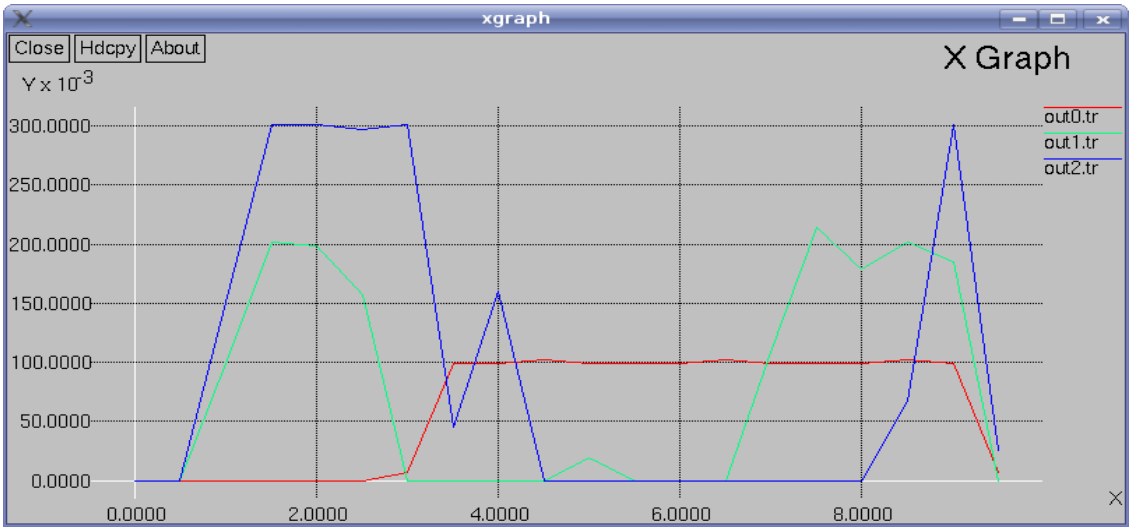



Figura 5.1 – Plotagem com política RED

A política de fila RED fez toda a diferença na hora de gerar o gráfico com o xgraph.

Hebert Luiz Amaral Costa
hebert.amaral@gmail.com