UNIVERSIDADE FEDERAL DE VIÇOSA
DEPARTAMENTO DE INFORMÁTICA
Mestrado em Ciência da Computação

INF651 – Redes de Computadores
Prof.: Carlos de Castro Goulart
Aluno: Marcelo Daibert
Data: 20.07.2007 – Trabalho NS2

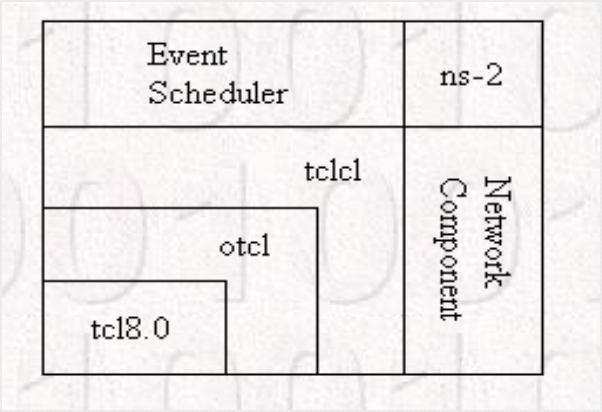
1. Argumente sobre as situações em que é conveniente ou indispensável a utilização de um simulador como o NS2. Comente sobre as principais funcionalidades do simulador NS2.

O NS2 (Network Simulator) é uma ferramenta de simulação de redes com o objetivo de auxiliar os trabalhos de pesquisa que envolvem redes de computadores. Ele é um simulador de eventos discreto resultante de um projeto conhecido como VINT (Virtual InterNetwork Testbed). Esse projeto possui colaboração da DARPA, USC/ISI, Xerox PARC, LBNL, e a universidade de Berkeley. É um simulador totalmente gratuito e com código fonte aberto, o que permite ao usuário proceder com 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 simulação é uma técnica útil para análise de desempenho de sistemas computacionais, principalmente quando os mesmos não estão disponíveis ou simplesmente seja necessário realizar algum teste simulando um ambiente real, que na prática haveria alguma dificuldade de realização. Com a simulação é facilidade as comparações com uma maior variedade de cargas e de ambientes. Simular em um ambiente real, principalmente com grandes possibilidades de se dar errado é muito custoso. Imagine a quantidades de testes necessários para a implementação de um novo protocolo ou algoritmo de roteamento, por exemplo. Em um ambiente real, seriam necessários vários ativos de redes re-configurados e na simulação, basta programar.

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. Nela serão efetivamente escritas as 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 freqüência. 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. *Pesquise e disserte sobre a arquitetura do NS2? Oriente-se pela figura a seguir.*



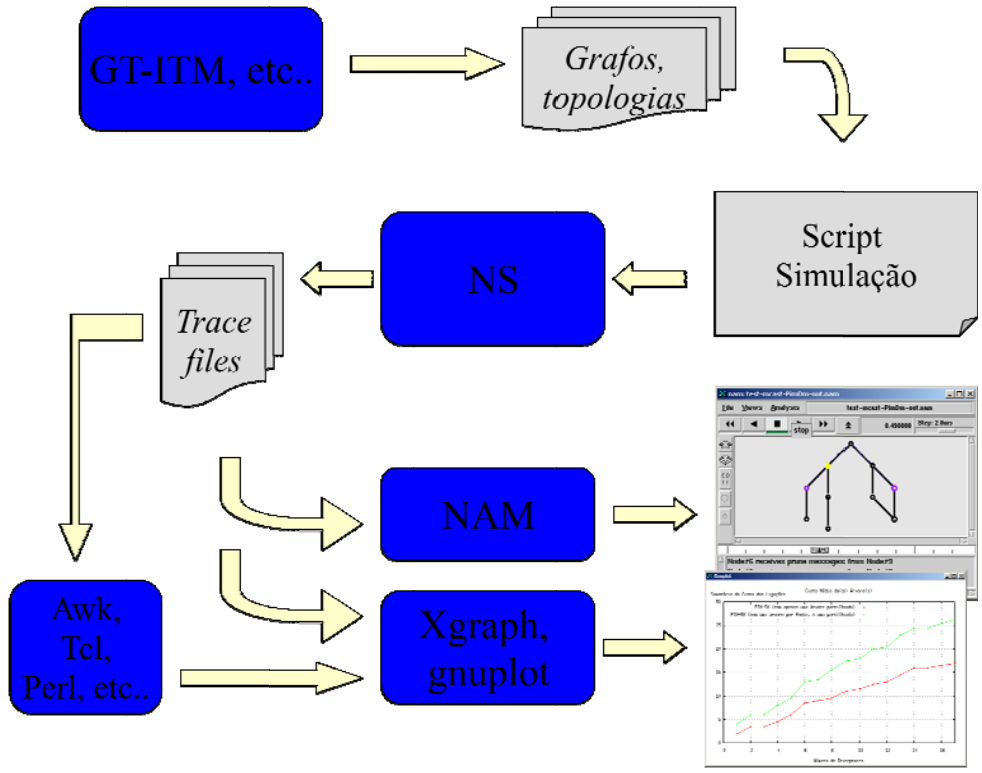
Um usuário qualquer (não um desenvolvedor do NS) pode estar no canto mais à esquerda e abaixo, planejando e rodando simulações em Tcl utilizando os objetos de simulação das bibliotecas OTcl. Os *schedulers* de evento (*Event Scheduler*) e a maioria das componentes de rede são implementados em C++ e disponíveis em OTcl pela ligação que é implementada utilizando tclcl. O TCL é uma linguagem script, fracamente tipada e tem como objetivo definir todo o ambiente de simulação. O OTCL e a TCL orientada a objetos. O TCLCL (TCLCLASS) é o TCL com suporte à "Object Oriented Split-Level Programming".

Quando uma simulação é feita, o NS produz um ou mais arquivos de saída baseadas em texto que contém dados da simulação detalhados. Os dados podem ser utilizados para análise de simulações ou como entrada para uma ferramenta de simulação gráfica chamada Network Animator (NAM) que é desenvolvido com uma parte do projeto VINT. O NAM tem uma interface de usuário gráfica bastante amigável e tem um controlador de velocidade. Além do mais, isto pode graficamente apresentar informações presentes como throughput e números de pacotes dropados por cada link, contudo as informações gráficas não podem ser utilizadas para análise de simulações acuradas.

O NS não é apenas escrito em OTcl, mas também em C++. Por razões de eficiência, o NS separa as implementações de caminho de dados das implementações de caminho de controle. Para reduzir os pacotes e o tempo de processamento do evento (que não são tempos de simulação), o scheduler de evento e os objetos de componentes de rede básicos nos caminhos dos dados são escritos e compilados utilizando C++. Estes objetos compilados são disponibilizados para o interpretador de OTcl por uma ligação OTcl que cria uma correspondência do objeto OTcl para cada um dos objetos C++. Também fazem com que as funções de controle e as variáveis de configuração especificadas pelo objeto C++ ajam como funções de membros e variáveis de membros de objetos OTcl correspondentes. Desta forma, os controles de objetos C++ são dados pelo OTcl. Isto também é possível para adicionar funções de membros e variáveis para o C++ ligados aos objetos OTcl. O objeto em C++ que não é necessário ser controlado na simulação ou internamente utilizado por um outro objeto, não precisa ser ligado ao OTcl.

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

A figura abaixo apresenta os principais componentes do NS2:



O primeiro componente é a própria ferramenta de simulação NS. Esta responsável com executar as simulações e gerar os devidos relatórios destas simulações – arquivos trace. O NAM (Network Animator) que é o visualizador/animador dos resultados do NS (e de outros simuladores também). As versões mais recentes do NAM permitem editar topologias e cenários de simulação. Como pré-processamento os componentes geradores de tráfego, e de topologias (INET, GT-ITM, TIERS, BRITE) são muito importantes principalmente para projetar a simulação, gerando grafos e obtendo a topologia do ambiente.

Após a execução da simulação, é muito importante o pós processamento, como o próprio NAM, para visualizar as simulações. São importantes os analisadores do arquivo de trace contendo o relatório da simulação gerado pelo NS. Entre eles os scripts Shell, AWK, Perl ou até mesmo TCL. Para então obter também uma visualização gráfica, onde se destacam as ferramentas Xgraph, gnuplot, tracegraph (necessita do matlab versão 6 acima para execução), Curve.

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.

Tcl, sigla de Tool Command Languag, é uma linguagem script simples, de fácil manutenção, razoavelmente independente do sistema operacional e fácil de extender usando C/C++, permitindo assim o suporte ao Split-Level Programming. Foi criada no final dos anos 80 por John Ousterhout. Em Tcl, uma variável deverá receber um valor de atribuição através do comando set ao contrário das outros linguagens que utilizam o sinal de "=" ou ":=" para fazer atribuições. Exemplo:

```
set vNome "Marcelo Daibert"
set vIdade 23
```

Para atribuição de uma variável em outra, deve-se também utilizar o comando set, mas passando entre colchetes a variável já existente com o símbolo $ na sua frente:

```
set vNome2 [$vNome]
```

Para exibição do texto no terminal, o comando puts é utilizado:

```
puts "Eu sou $vNome"
```

Para atribuição do conteúdo de algum arquivo em disco, o comando deve ser:

```
set vRetorno [open "c:\texto.txt" w]
```

A linguagem apresenta uma vasta gama de funções matemáticas: abs, cosh, log, sqrt, acos, double, log10, srand, asin, exp, pow, tan, atan, floor, rand, tanh, atan2, fmod, round, wide, ceil, hypot, sin, cos, int, sinh.

Uma estrutura de dados muito importante para desenvolvimento é as listas e matrizes que o tcl implementa como no exemplo abaixo:

```
set idade(1) 10
set idade(2) 16
set idade(3) 15
```

ou apenas: `array set idade {1 "10" 2 "16" 3 "15"}`

Para definição de matrizes o procedimento é semelhante, mas com o cuidado de adição das dimensões da matriz. Segue abaixo um exemplo de uma matriz 2x2:

```
set matriz(0,0) "0-0"
set matriz(0,1) "0-1"
set matriz(0,0) "1-0"
```

```
set matriz(0,0) “1-1”
```

Para manipulação de matrizes e vetores, a tcl apresenta alguns comandos para isso.

Saber se um vetor existe

```
array exists NomedoArray
```

Retorna 1 se NomedoArray é um array e 0 se não há nenhuma variável com aquele nome ou se é uma variável scalar (variável comum).

Exemplos:
1)Testando uma variável scalar

```
set v1 0
array exists v1
Retorna:
0
```

2)Testando um array

```
array set boLa {1 b 2 o 3 L }
array exists boLa
Retorna:
1
```

Saber a quantidade de elementos de um Vetor

```
array size NomedoArray
```

Retorna uma string decimal contendo a quantidade de elementos do array. Se NomedoArray não for um array será retornado 0.

Exemplo:

```
array set valor {1 “Rio” 2 “São Paulo” 3 “Minas”}
array size valor
```

O valor de retorno será 3

O comando FOR é utilizado para fazer uma contagem de um valor inicial definido pelo usuário até um valor final também definido pelo usuário. Sintaxe:

```
for {inicialização} {condição} {incremento} {corpo}
```

Onde:
{incicialização} é a inicialização da variável utilizada pelo contador.

{condição} será a condição verificada a cada loop para continuar ou parar o laço de repetição.

{incremento} é o incremento ou decremento sobre a variável do contador.

{corpo} é o que deverá ser repetido várias vezes.

Exemplo:

Contar de 1 até 10,

```
for {set conta 1} {$conta < 11} {incr 1} {
    puts $conta
}
```

O comando FOREACH permite ao programador definir os elementos que irão ser contados durante o loop. Estes elementos poderão ser números ou strings. Suponha que você está criando uma rotina para desenhar um MENU, é possível utilizar o foreach para exibir cada comando do menu, veja a sintaxe e um exemplo do foreach. Sintaxe:

```
foreach variável {lista} {corpo}
```

Onde:

variável, é a variável que receberá cada um dos elementos da lista.

{lista}, é a lista que será utilizada para passar os valores para a estrutura.

{corpo}, são os comandos que deverão ser repetidos a cada loop.

Exemplo:

```
foreach opcoes {“Novo” “Abrir” “Salvar”} {
    puts $opcoes
}
```

A cada passada do loop, um valor da lista será passado para a variável até que se chegue ao último valor.

O comando while irá executar uma série de comandos enquanto uma condição for verdadeira, caso na primeira verificação já for verificado que a condição é falsa nenhum comando do corpo será executado nenhuma vez. Sintaxe:

```
while {condição} {corpo}
```

Onde:

{condição}, será a condição verificada a cada loop para continuar ou parar o laço de repetição.

{corpo}, são os comandos que deverão ser repetidos a cada loop

Exemplo:

```
set conta 0
while {$conta < 10} {
```

```
    puts $conta
    incr conta
}
```

Em TCL existe uma outra estrutura de dados que chamamos de LISTA, assim como com as variáveis e array's , as LISTAS são criadas utilizando o comando SET. A diferença de uma lista para uma variável é que a variável guarda um único valor enquanto uma LISTA guarda vários valores como um array. É importante deixar bem claro que LISTA não é o mesmo que VETOR ou MATRIZ. Como uma LISTA possui vários elementos, temos que nos referir numericamente a cada um deles. O primeiro elemento é o de índice 0 e o último é o END.

Para criamos uma LISTA estática, deve-se por os valores que irão compo-la entre chaves "{}", exemplo:

```
set lista_frutas {"banana" "laranja" "morango"}
```

Para criarmos uma LISTA dinamicamente utilizaremos o comando lappend que irá re-tornar uma LISTA contendo os argumentos recebidos, veja a sintaxe deste comando e em seguida um exemplo:

```
lappend nome_lista valor1 valor2 valorN
```

Exemplo:

```
lappend lista_cidades "Rio de Janeiro" "São Paulo" "Recife"
```

ou

```
gets stdin vcidades
lappend lista_cidades $vcidades
```

Para acessar os valores dentro da LISTA utilize o comando lindex o qual retorna o elemento da lista que está em uma posição passada como parâmetro, veja a sintaxe e um exemplo. Sintaxe:

```
lindex $lista posição
```

Exemplo:

```
for { set conta 0 } { $conta < 2 } { incr conta } {
    set varquivo [lindex $lista_arquivos $conta]
    puts "O [expr $conta + 1]º arquivo é $varquivo"
}
```

Lembre-se que TODOS os comandos que retornam algo deverá estar entre colchetes " [] " para que o valor retornado possa ser recebido por alguém. O comando lreplace cria uma nova LISTA com o valor entre um dado intervalo substituído na LISTA criada, mantendo a lista original com o mesmo valor, veja a sintaxe e um exemplo:

```
lreplace $lista min max novo_valor
```

Onde:
min é o primeiro índice
max é o segundo índice
Imagine a seguinte lista { "Banana" "Pera" "Uva" "Laranja" "Jabuticaba" }
A Banana é o elemento de índice 0,
A Jabuticaba é o elemento de índice END,
Logo se queremos substituir o elemento de índice 2, que é a Uva, indicamos no comando lreplace 1 como índice menor e 3 como índice maior ou 2 para o índice menor e maior veja os exemplos:

Indicando indices diferente:

```
lreplace $lista 1 3 "Jaca"
```

Indicando índices iguais:

```
lreplace $lista 2 2 "Jaca"
```

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

Passos para Criar uma Simulação:

1. Planejar a simulação;
2. Criar o escalonador de eventos;
3. Ativar as opções de trace para produzir os outputs desejados;
4. Criar a topologia da rede;
5. Ativar o encaminhamento (unicast e multicast);
6. Criar conexões de transporte (TCP e/ou UDP);
7. Transmitir dados entre as aplicações – executar a simulação;

Exemplo:

```
#Instanciando o Objeto de Simulação
set ns [new Simulator]

#Definindo as Cores
$ns color 1 Blue
$ns color 2 Green

#Abrindo o arquivo de trace para o nam
set nf [open out.nam w]
$ns namtrace-all $nf

#Definindo o Procedimento Finish
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

#Criando quatro nós
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Criando os links entre os nós
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Configurando o tempo de espera entre o link para 10
```

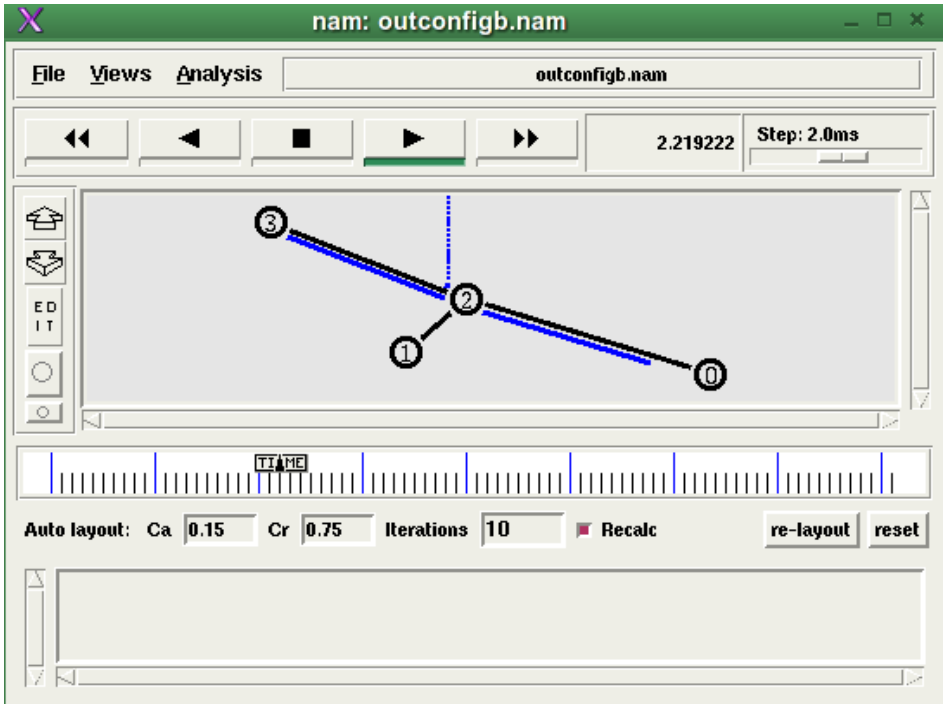
```
$ns queue-limit $n2 $n3 10

#Configurando uma Conexão TCP
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Configurando uma aplicação FTP para o TCP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


#Configurando uma conexão UDP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Configurando o CBR para o UDP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false


#Agendamento de inicio e fim das conexões
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Chama o método finish aos 5 segundos
$ns at 5.0 "finish"

#Executa a Simulação
$ns run
```

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, entre outros.

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. É este arquivo de trace que o NAM irá se basear para gerar as animações. Nele contém todo o histórico da simulação.

O NAM tem uma interface de usuário gráfica bastante amigável e tem um controlador de velocidade. Pode ainda apresentar, graficamente, informações presentes como vazão e números de pacotes emitidos para cada link. Contudo as informações gráficas não podem ser utilizadas para análise profunda de simulações. O NAM pode ainda ser utilizado como gerador de scripts, através do NAM Editor. Segue abaixo uma figura do NAM apresentando o resultado de uma simulação.



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

A ilustração de um estudo comparativo, através de gráficos, constitui-se em uma das principais contribuições do NS. Isso é possível através das ferramentas xGraph e gnuPlot que com as devidas entradas conseguem apresentar gráficos de desempenho das simulações, sendo uma importante ferramenta principalmente para comparação de desempenho.

O gnuplot é um programa para fazer gráficos 2D e 3D. Ele permite a visualização de gráficos de funções de uma ou duas variáveis, bem como de dados experimentais (lidos de algum arquivo). Todas as funções elementares estão incluídas, além de algumas especiais não estudadas em cursos básicos de cálculo. Os gráficos podem ser personalizados de muitas maneiras: cor das linhas, estilo das linhas, com ou sem borda, e muito mais. Também é possível visualizar mais de uma função no mesmo gráfico para fazer comparações. A homepage do gnuplot é http://www.gnuplot.info/.

Já o xGraph é uma ferramenta mais simples quando comparada ao gnuPlot. Esta ferramenta é um gerador de gráficos X-Y de uso geral com opções interativas de “zoom”, impressão e opções de visualização. Permite a visualização de uma ou mais curvas de tráfego no mesmo gráfico. São criados gráficos a partir de dados contidos em um ou mais arquivos, gerados pelo script TCL da simulação. O xgraph produz arquivos PostScript, PDF e MIF (Maker Interchange Format) para serem impressos, armazenados, compartilhados ou inseridos em outros arquivos. Permite configuração de cores e espessura das linhas.

Vários outros arquivos podem ser gerados por funções escritas no programa da simulação ou com a utilização de monitores colocados em pontos de interesse da topologia em estudo. Estes recursos permitem totalizar pacotes ou bytes que chegam a um nó da rede por unidade de tempo, calcular a vazão em um certo enlace, totalizar descarte ou criar dados para geração de gráficos.

8. Os arquivos de trace são os dados resultados da execução da simulação de um cenário, que são guardados em arquivos. 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. b-) Liste quantos pacotes foram recebidos. c-) Informe o tamanho e o intervalo de tempo de pacotes desenfileirados. d-) Faça um somatório do tamanho dos pacotes que foram dropados.

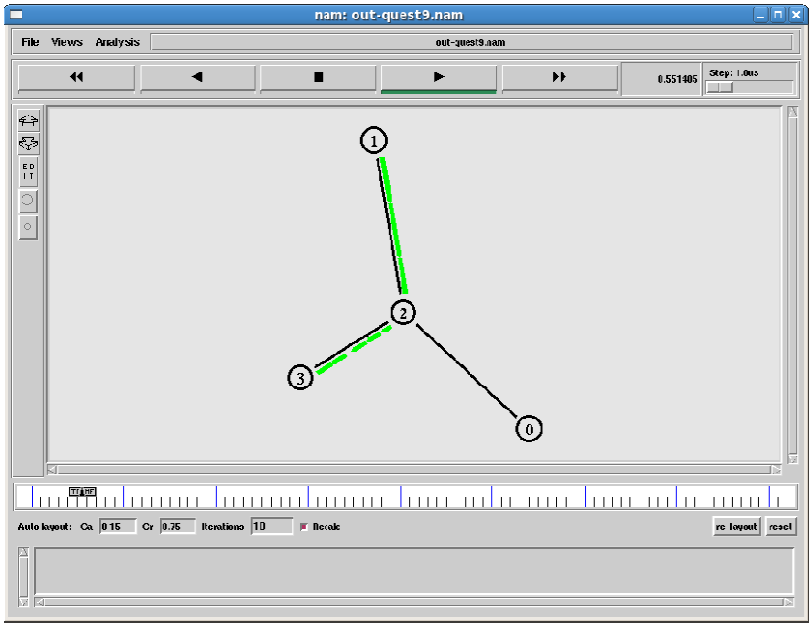
```
a)awk '{if ($2 > 3 && $2 < 4 && $1 == "+") print $2}' out.tr > final1.txt
b)awk '{if ($1 == "r") total += 1} END {print total}' out.tr > final2.txt
c)awk '{if ($1 == "-") print $6, $2}' out.tr > final3.txt
d)awk '{if ($1 == "d") total += $6} END {print total}' out.tr > final4.txt
```

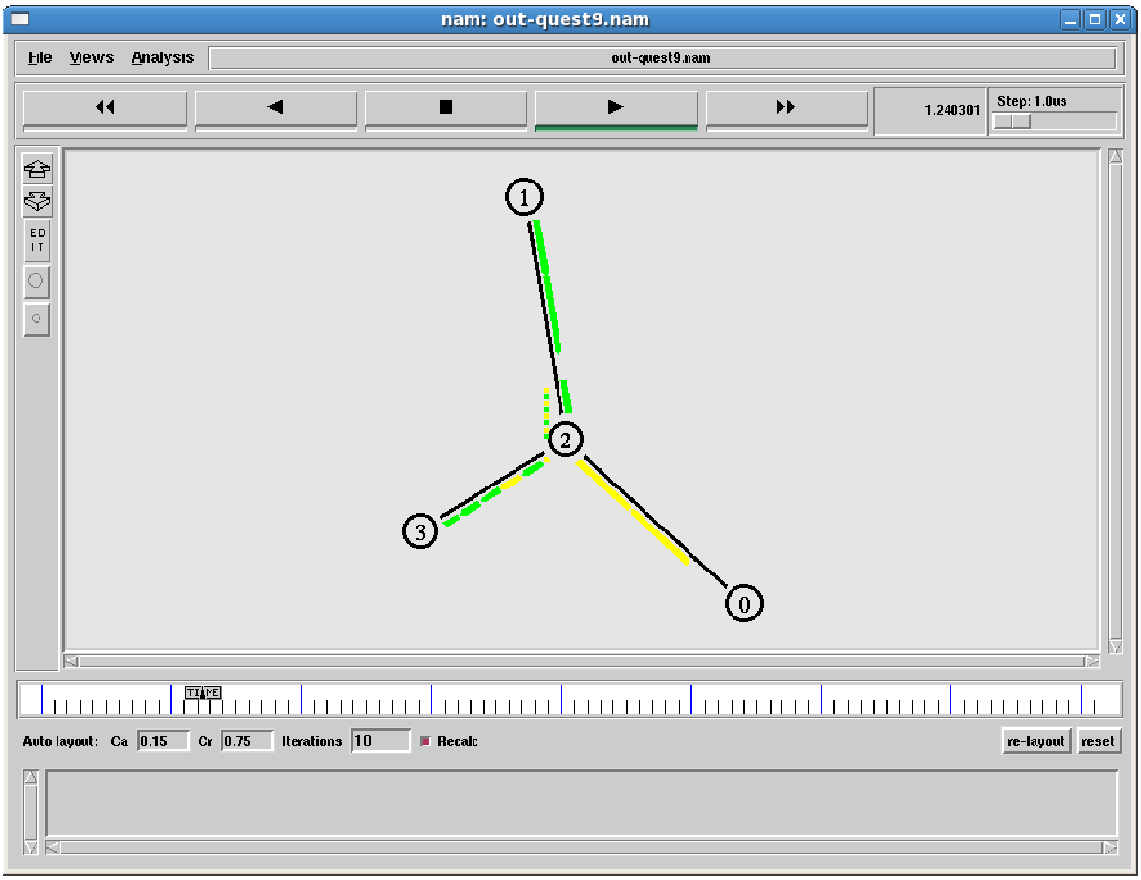
9. CRIATIVIDADE: Crie um cenário de rede para ser modelado, que envolva hosts(nodos), links(canal de ligação entre os nodos), tipos de enlace(com fio e sem fio), defina a largura de banda dos links, o delay, o tipo de enfileiramento, protocolos de transporte, fontes, destinos, aplicações e período de 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.

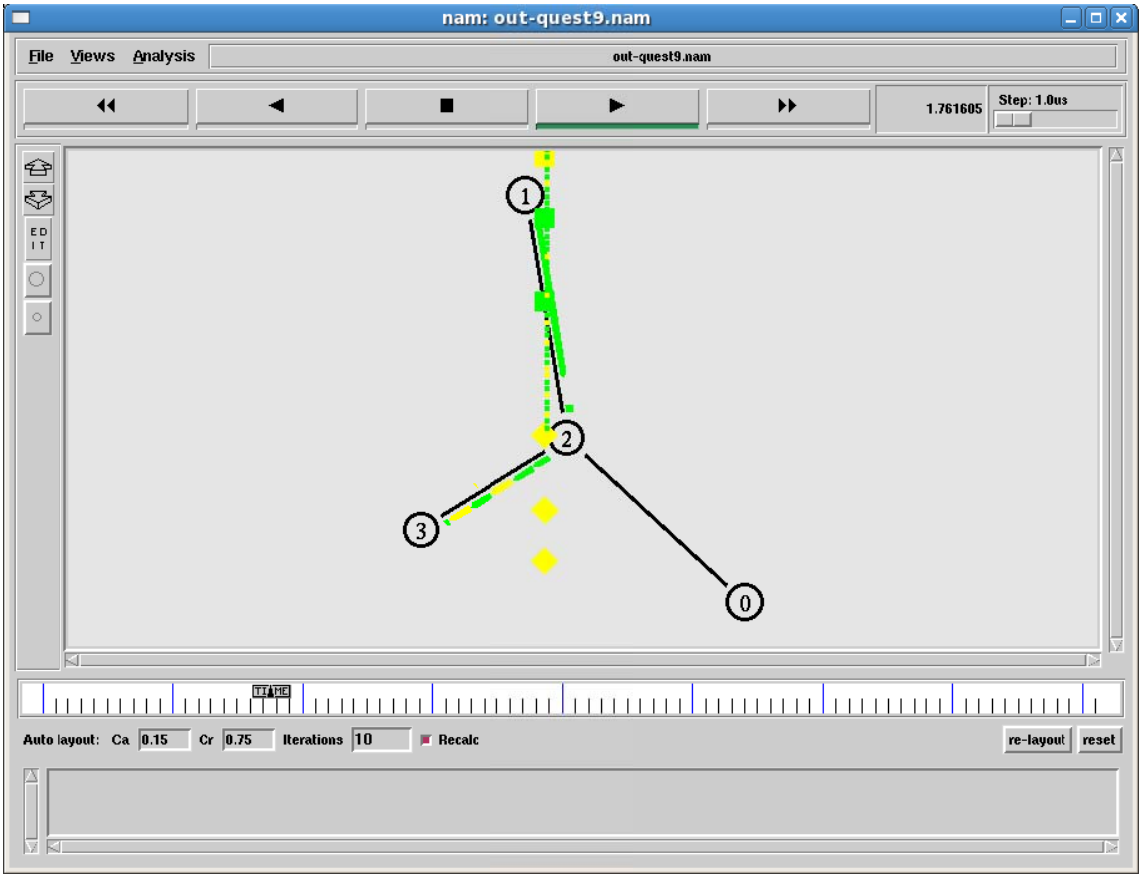
Nesta questão, eu busquei desenvolver uma simulação semelhante ao que foi apresentada pelo Hebert na sua aula sobre NS. Busquei apresentar 4 nós de rede, do 0 ao 3. Do nó 0 ao 2 e do 1 ao 2 existe um atraso de 10ms e um link de 2mb. Já do 2 ao 3 existe um link de 2mb com 20ms de atraso. Todos com a política de fila DropTail.

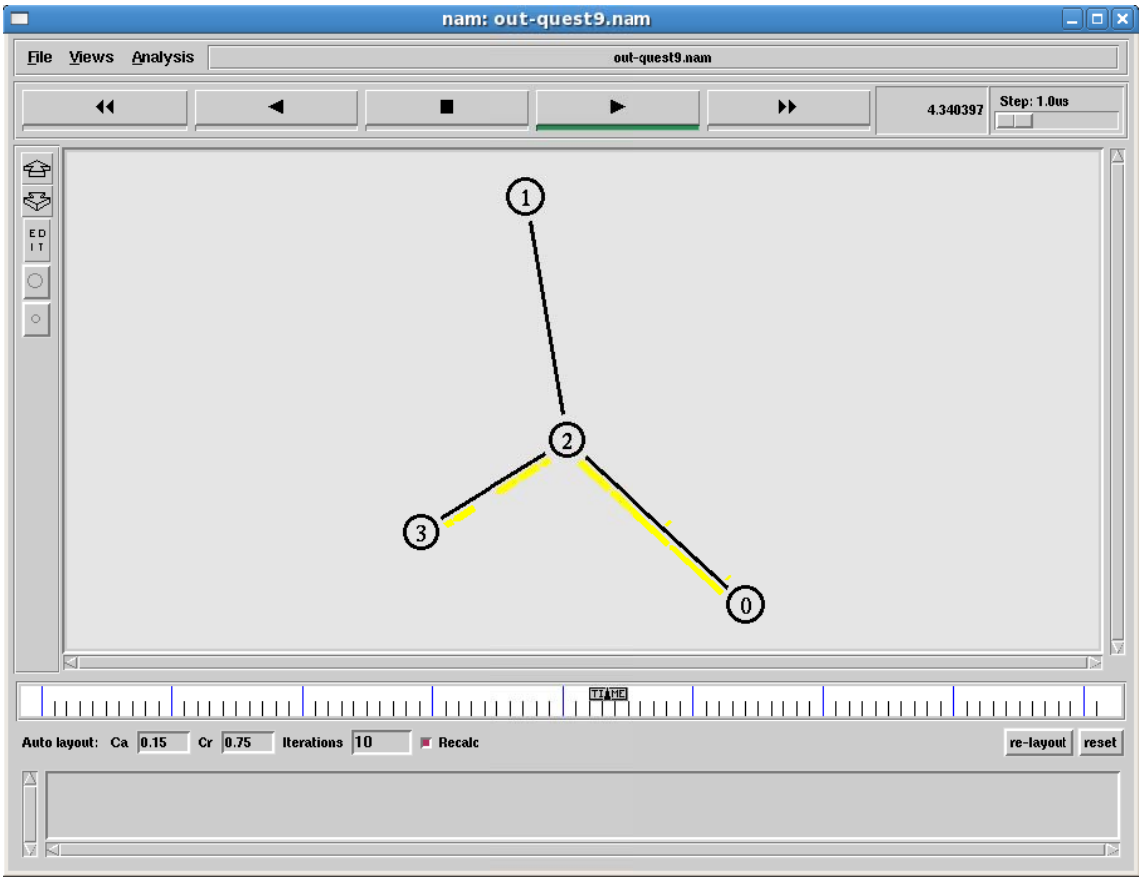
Entre o link 2 e 3, eu configurei um monitorador da fila de pacotes, para identificar uma possível visualização de gargalo que eu quero simular. Do nó 0 ao 3 eu configurei um agente TCP com o serviço de FTP rodando. Do 1 ao 3 foi configurado uma conexão UDP com um tráfego CBR com uma taxa de 1.8mb/s e pacote de tamanho 3000bytes.

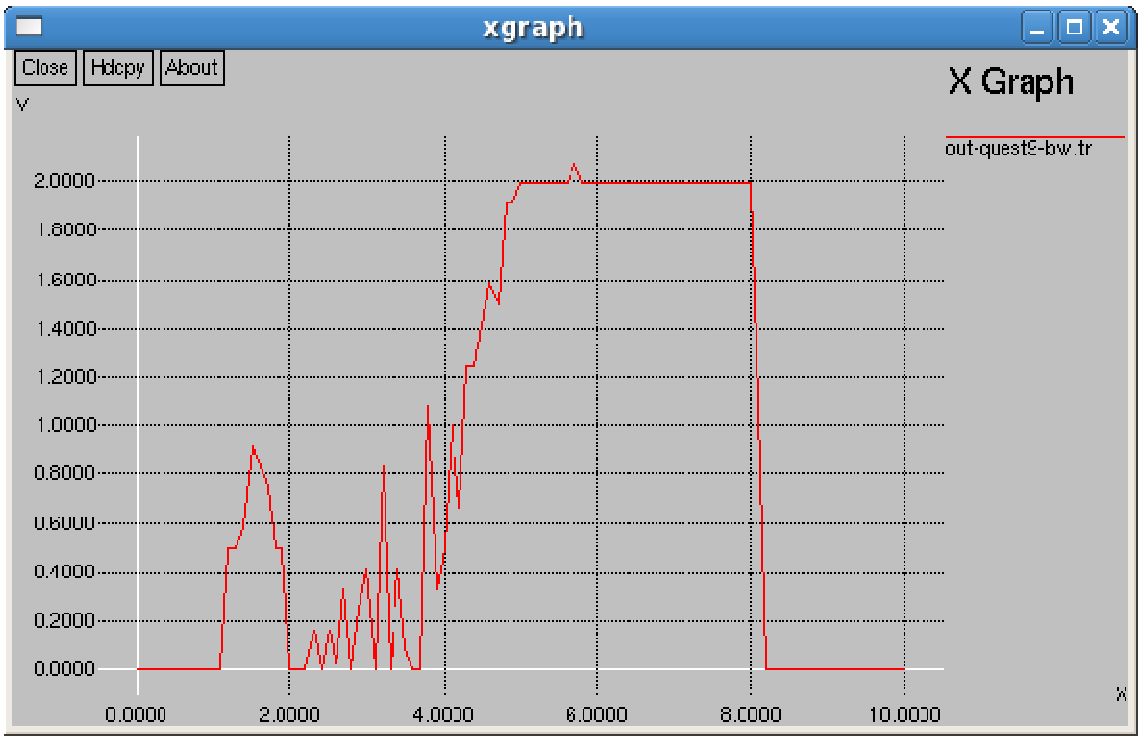
Esta simulação apresenta um total de 10 segundos. No tempo 0.1s é iniciado o tráfego CBR. Em 1.0 segundos o tráfego do FTP é iniciado. No tempo 3.5 segundos é finalizado o tráfego CBR e aos 8.0 segundos é finalizado o FTP. Aos 10.0 segundos a simulação é finalizada. Abaixo é apresentado o script tcl desta simulação. Logo abaixo são apresentadas as telas de visualização desta simulação rodando no NAM. Existe uma função no script chamada Record que é responsável por gerar um arquivo de saída com as informações de largura de banda utilizada pelo FTP. As figuras abaixo apresentam as fases da simulação vistas no NAM e o XGRAPH. Logo após as imagens é apresentado o script TCL desta simulação. É possível visualizar com esta simulação claramente o mecanismo de controle de fluxo do TCP, a janela deslizante. Assim que ocorre uma perda, existe uma diminuição clara do fluxo FTP.




nam: out-quest9.nam
File Views Analysis
out-quest9.nam
1.240301
Step: 1.0us
1
2
3
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset


nam: out-quest9.nam
File Views Analysis
out-quest9.nam
1.761605
Step: 1.0us
1
2
3
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset


nam: out-quest9.nam
File
Views
Analysis
out-quest9.nam
4.340397
Step: 1.0us
1
2
3
0
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset


xgraph
Close
Hdcpy
About
X Graph
out-quest9-bw.tr
2.0000
1.8000
1.6000
1.4000
1.2000
1.0000
0.8000
0.6000
0.4000
0.2000
0.0000
0.0000
2.0000
4.0000
6.0000
8.0000
10.0000

```
set ns [new Simulator]

$ns color 1 Yellow
$ns color 2 Green

set nf [open out-quest9.nam w]
$ns namtrace-all $nf

set nf2 [open out-quest9.tr w]
$ns trace-all $nf2

set f0 [open out-quest9-bw.tr w]

proc finish {} {
        global ns nf nf2 f0
        $ns flush-trace
        close $nf
      close $nf2
      close $f0
        exec nam out-quest9 &
      exec xgraph out-quest9-bw.tr &
        exit 0
}
proc record {} {
        global sink f0
      set ns [Simulator instance]
        set time 0.100
        set bw0 [$sink set bytes_]
        set now [$ns now]
        puts $f0 "$now [expr $bw0/$time*8/1000000]"
        $sink set bytes_ 0
        $ns at [expr $now+$time] "record"
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2Mb 20ms DropTail

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

set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
$ftp set type_ FTP

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 3000
$cbr set rate_ 1.8mb
$cbr set random_ false

$ns at 0.0 "record"
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 3.5 "$cbr stop"
$ns at 8.0 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

Trabalho Prático

Cenário 1

Este cenário apresenta seis nós, do n0 ao n5. Estes nós estão ligados entre si da seguinte forma: n0 -> n1 em um link de 2MB e 10ms de atraso, n1 -> n2 em um link de 1mb e 20ms de atraso, n1 -> n3 link de 2mb e 10ms de atraso, n1 -> n4 com link de 1mb e 20ms de atraso, n4 -> n5 em um link de 2mb e atraso de 10ms e por fim n2 -> n5 com 2mb de link de 10ms de atraso. Todos os links com a política de fila DropTail.

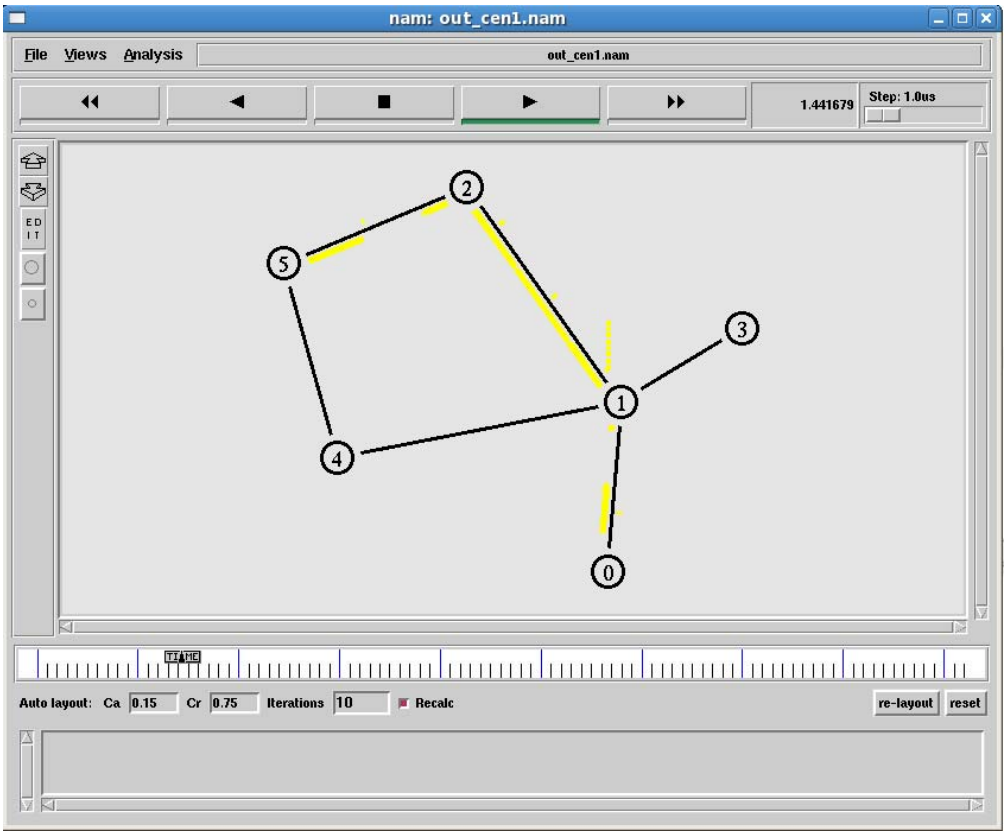
Do n0 ao n5 foi criada uma ligação com uma conexão do tipo TCP, onde foi associada com uma fonte de tráfego FTP. A simulação possui no total 10segundos. O tráfego FTP é iniciada em 0.5s e finalizada em 9.0 segundos. Foi adicionado no nó 1 um monitor de fila. Estão sendo criados 3 arquivos. Um de trace geral (trace all), outro para o trace do NAM e outro com as informações de largura de banda gerados pelo tráfego de FTP. Abaixo é apresentado o script TCL deste cenário. Após são apresentadas algumas imagens da simulação de do gráfico de largura de banda que o tráfego de FTP gera. É possível observar que o valor máximo alcançado é de 1MB, já que o entre o n1 e n2 o link é de 1MB. É possível observar inclusive a utilização da fila do n1. Neste cenário não ocorre nenhuma operação DROP de pacotes. É apresentado logo após ao script TCL os comandos AWK para visualização da quantidade dos pacotes dropados e seu instante. Mas como é possível observar, não houve nenhum ocorrência.

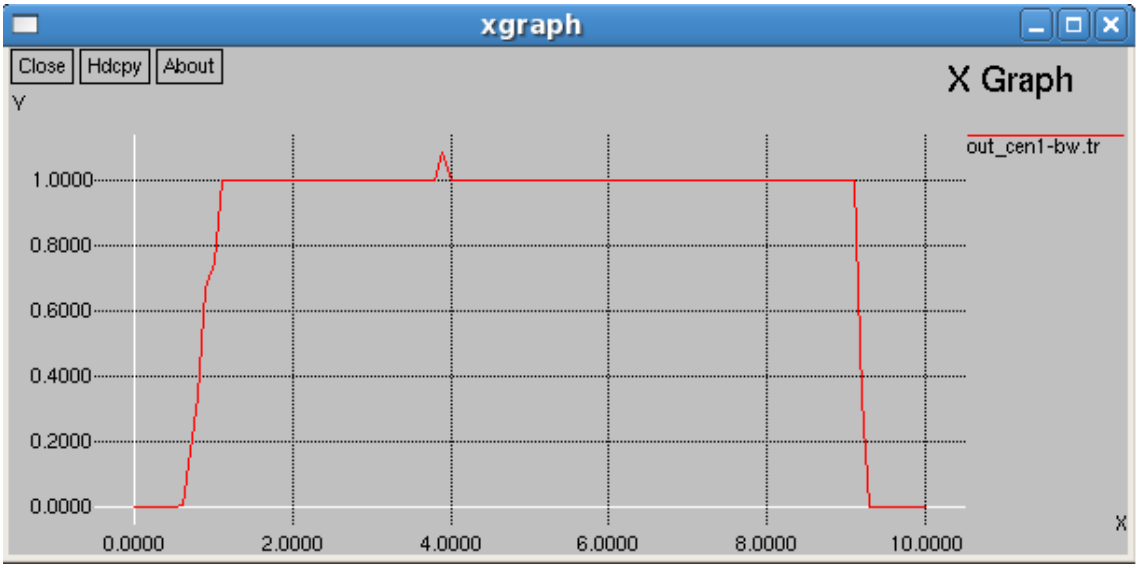
```
set ns [new Simulator]

$ns color 1 Yellow

set nf [open out_cen1.nam w]
$ns namtrace-all $nf

set nf2 [open out_cen1.tr w]
$ns trace-all $nf2

set f0 [open out_cen1-bw.tr w]

proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out_cen1 &
      exec xgraph out_cen1-bw.tr &
        exit 0
}
proc record {} {
        global sink f0
       set ns [Simulator instance]
        set time 0.100
        set bw0 [$sink set bytes_]
        set now [$ns now]
        puts $f0 "$now [expr $bw0/$time*8/1000000]"
```

```
        $sink set bytes_ 0
        $ns at [expr $now+$time] "record"
}

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

$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 20ms DropTail
$ns duplex-link $n1 $n3 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 1Mb 20ms DropTail
$ns duplex-link $n4 $n5 2Mb 10ms DropTail
$ns duplex-link $n2 $n5 2Mb 10ms DropTail

$ns duplex-link-op $n1 $n2 queuePos 0.5

set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

$ns at 0.0 "record"
$ns at 0.5 "$ftp start"
$ns at 9.0 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

AWK:

```
awk '{if($1 == "d") total += 1 } END {print "Total Pacotes Dropados:
         ",total}' out_cen1.tr

awk '{if($1 == "d") print $1,$2}' out_cen1.tr
```


nam: out_cen1.nam
File
Views
Analysis
out_cen1.nam
1.441679
Step: 1.0us
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset
0
1
2
3
4
5


xgraph
Close
Hdcpy
About
X Graph
Y
out_cen1-bw.tr
1.0000
0.8000
0.6000
0.4000
0.2000
0.0000
0.0000
2.0000
4.0000
6.0000
8.0000
10.0000
X

Cenário 2

Este cenário estende o cenário 1 com algumas modificações. Entre o n1 e n2, a política de filas foi alterada para a SFQ. A mesma modificação foi realizada no link do n1 ao n4. Foi adicionado também dois novos tráfegos na simulação: Dois tráfegos CBR foram adicionados em agentes UDP. Ambos com uma taxa de transmissão de 2mbps e com 512bytes de tamanho de pacotes. O primeiro tráfego é iniciado no instante 3 segundos e finalizado em 7 segundos, já o segundo é iniciado em 4 segundos e finalizado também em 7 segundos.

Abaixo são apresentadas algumas imagens da simulação no NAM e o gráfico de largura de banda usada por cada fonte. É possível observar na primeira imagem somente o tráfego FTP em amarelo. Como pode ser visto no gráfico de largura de banda utilizado, o tráfego de FTP chega ao máximo que o link suporta, ou seja, 1MB. Aos 3 segundos é iniciado um tráfego CBR com 2mbps de taxa de transferência. Este apresentado na segunda imagem na cor verde. Com pouco tempo este tráfego já ocupa praticamente toda a rede e ocorrem vários drops de pacotes. O tráfego do FTP é totalmente minimizado graças ao controle de tráfego de janelas deslizantes. O tráfego UDP ocupa toda a rede. Aos 4 segundos outro tráfego CBR é iniciado, como visualizado na quarta imagem em azul. Na política de filas SFQ, a banda é dividida entre os dois tráfegos CBR. Aos 7 segundos os tráfegos CBR se encerram e a rede fica livre novamente para o FTP, que aos poucos vai aumentando a velocidade de transmissão. São dropados nesta simulação 2473 pacotes.

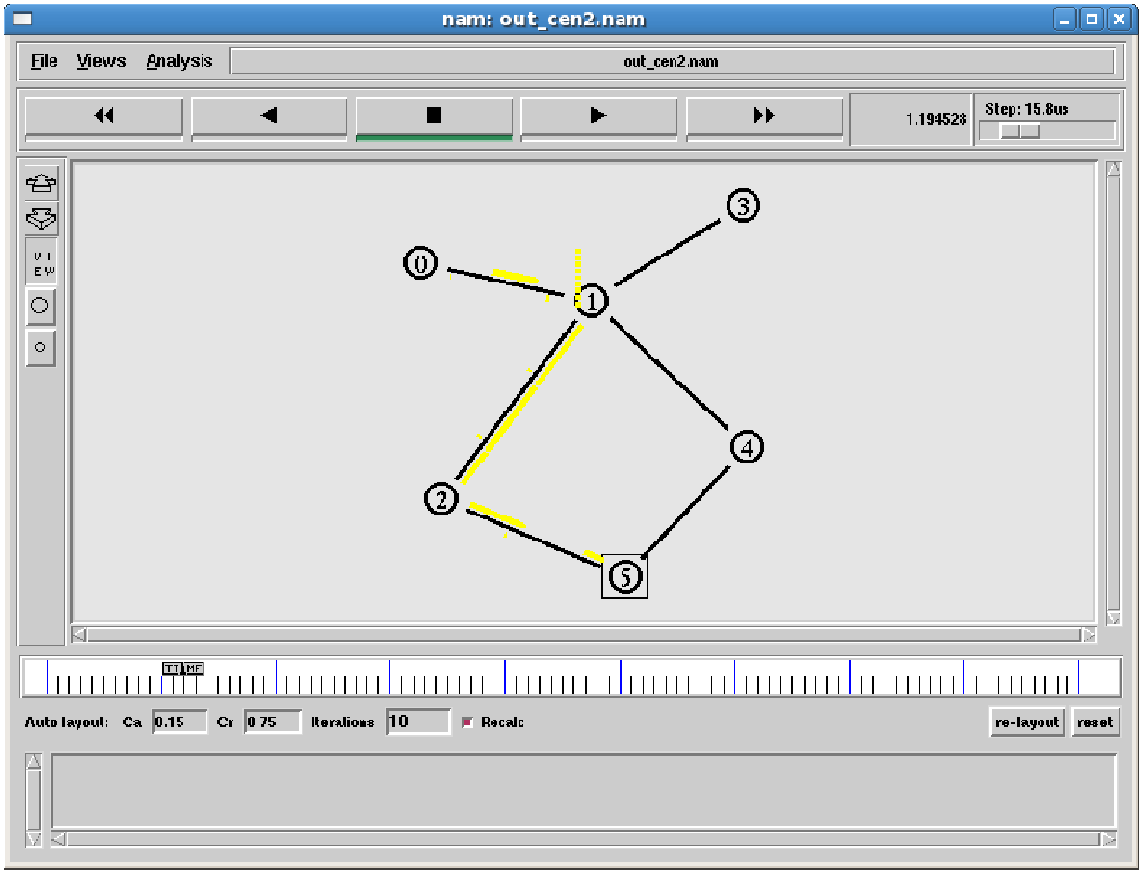
```
set ns [new Simulator]

$ns color 1 Yellow
$ns color 2 Green
$ns color 3 Blue

set nf [open out_cen2.nam w]
$ns namtrace-all $nf

set nf2 [open out_cen2.tr w]
$ns trace-all $nf2

set f0 [open out_cen2FTP-bw.tr w]
set f1 [open out_cen2CBR1-bw.tr w]
set f2 [open out_cen2CBR2-bw.tr w]

proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out_cen2 &
      exec  xgraph  out_cen2FTP-bw.tr  out_cen2CBR1-bw.tr  out_cen2CBR2-
bw.tr &
        exit 0
}
proc record {} {
        global sink f0 f1 f2 sinkudp sinkudp2
```

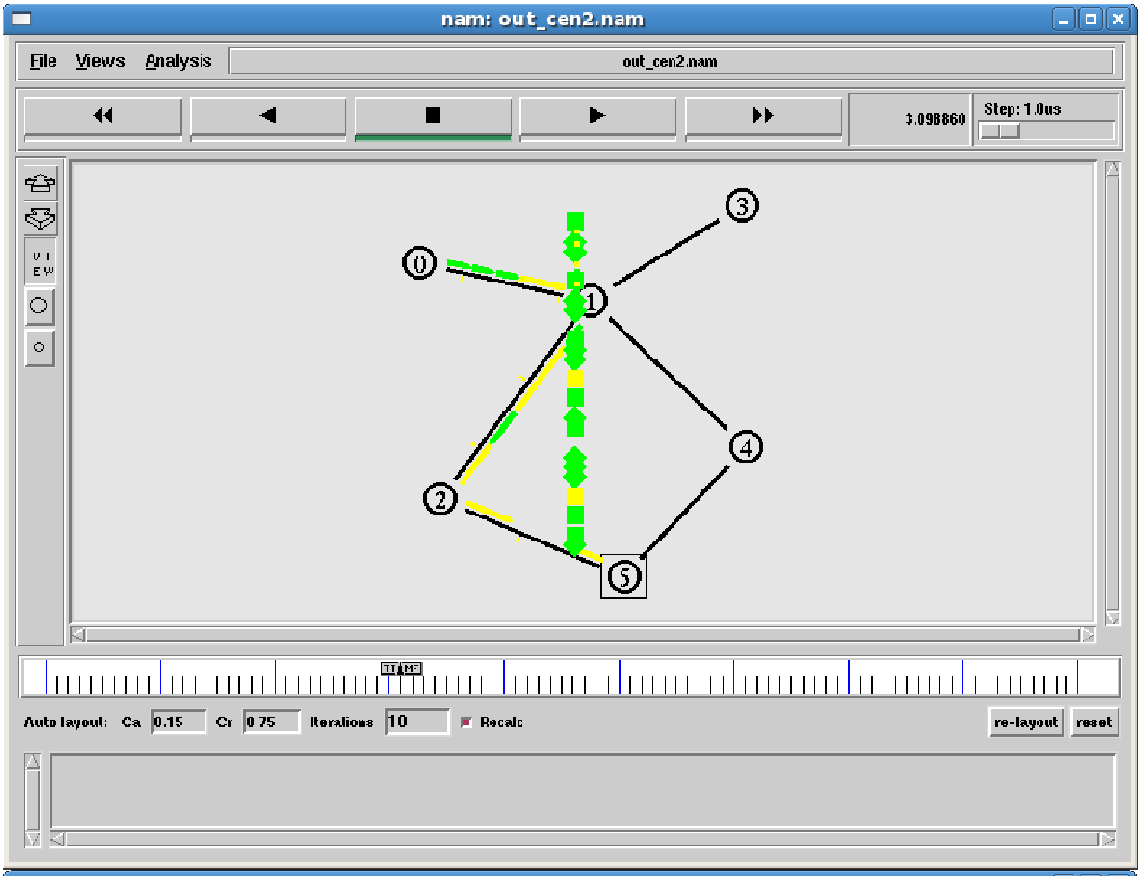
```
      set ns [Simulator instance]
        set time 0.100
        set bw0 [$sink set bytes_]
      set bw1 [$sinkudp set bytes_]
      set bw2 [$sinkudp2 set bytes_]
        set now [$ns now]
        puts $f0 "$now [expr $bw0/$time*8/1000000]"
      puts $f1 "$now [expr $bw1/$time*8/1000000]"
      puts $f2 "$now [expr $bw2/$time*8/1000000]"
        $sink set bytes_ 0
      $sinkudp set bytes_ 0
      $sinkudp2 set bytes_ 0
        $ns at [expr $now+$time] "record"
}

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

$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 20ms SFQ
$ns duplex-link $n1 $n3 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 1Mb 20ms SFQ
$ns duplex-link $n4 $n5 2Mb 10ms DropTail
$ns duplex-link $n2 $n5 2Mb 10ms DropTail

$ns duplex-link-op $n1 $n2 queuePos 0.5

set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
set sinkudp [new Agent/LossMonitor]
$ns attach-agent $n5 $sinkudp
$ns connect $udp $sinkudp
```

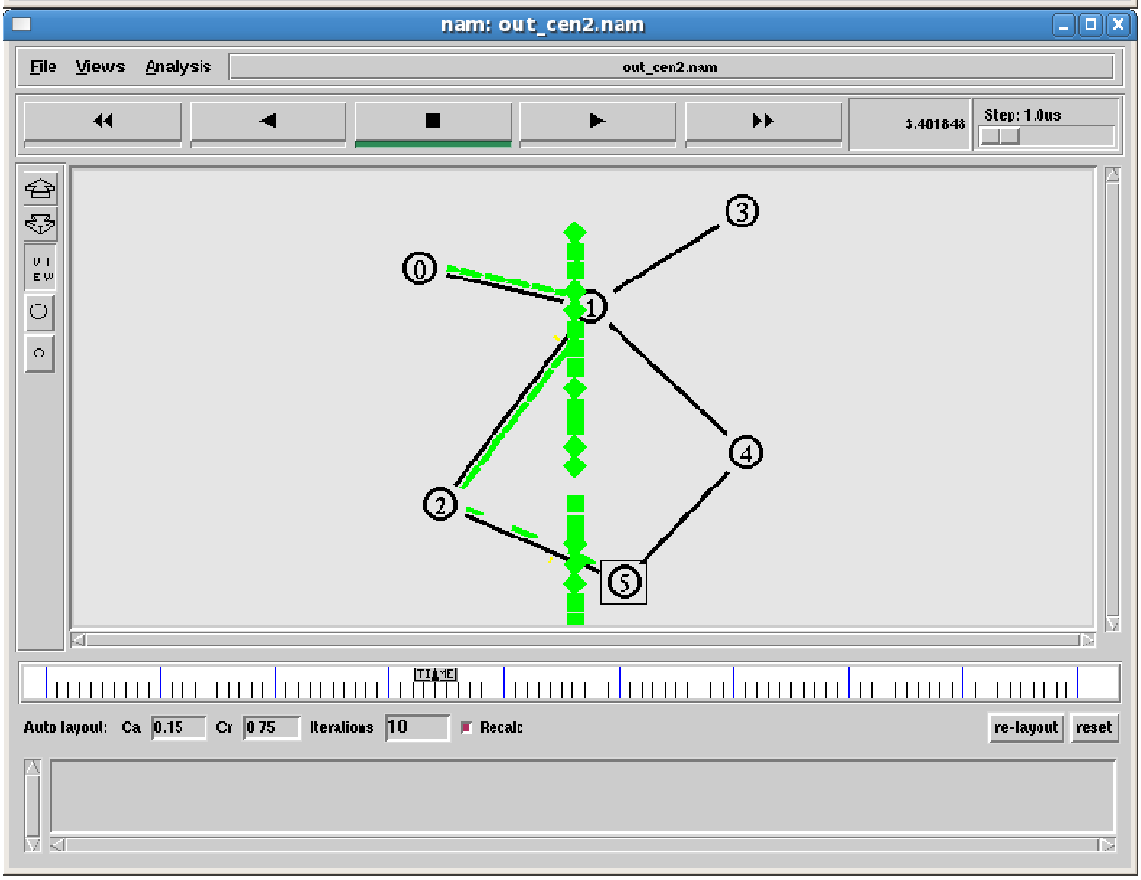
```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 512
$cbr set rate_ 2mb
$cbr set random_ false

set udp2 [new Agent/UDP]
$ns attach-agent $n3 $udp2
set null2 [new Agent/Null]
$ns attach-agent $n5 $null2
$ns connect $udp2 $null2
$udp2 set fid_ 3
set sinkudp2 [new Agent/LossMonitor]
$ns attach-agent $n5 $sinkudp2
$ns connect $udp2 $sinkudp2

set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set type_ CBR
$cbr2 set packet_size_ 512
$cbr2 set rate_ 2mb
$cbr2 set random_ false


$ns at 0.0 "record"
$ns at 0.5 "$ftp start"
$ns at 3.0 "$cbr start"
$ns at 4.0 "$cbr2 start"
$ns at 7.0 "$cbr2 stop"
$ns at 7.0 "$cbr stop"
$ns at 9.0 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

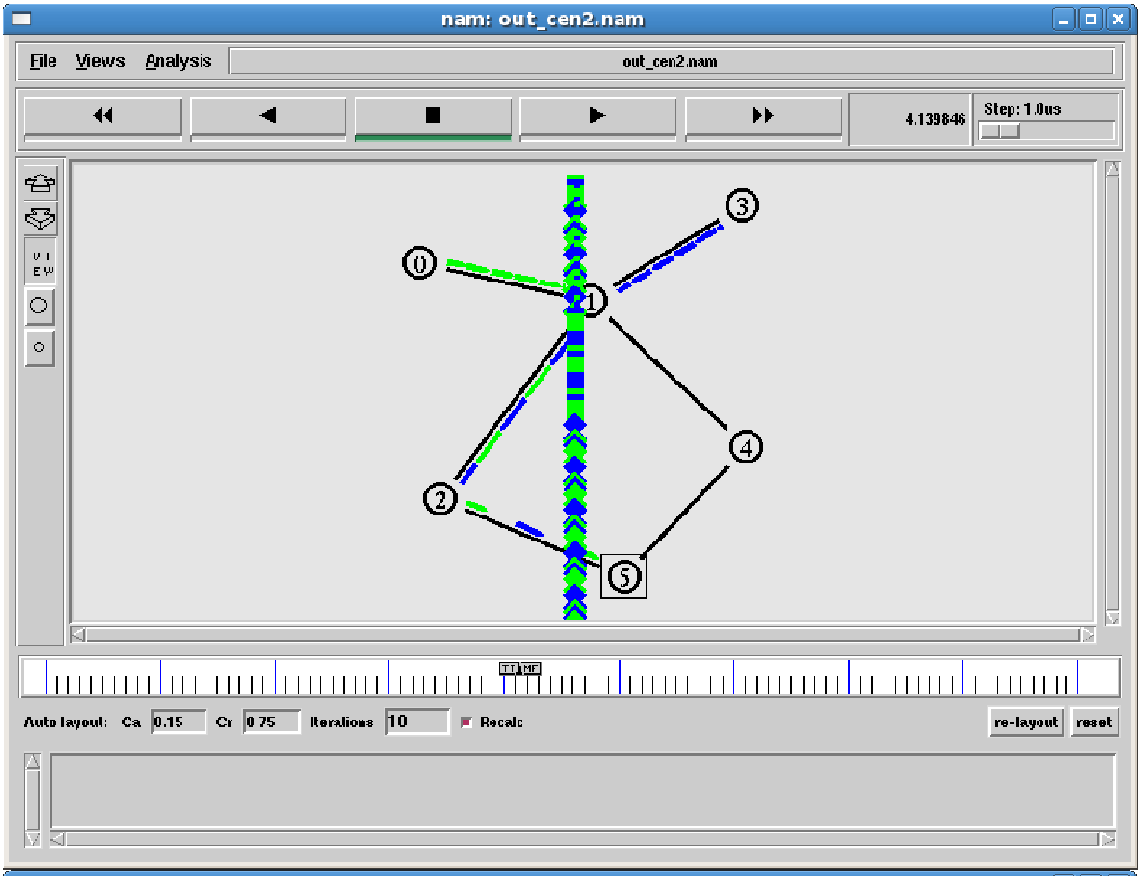
AWK:

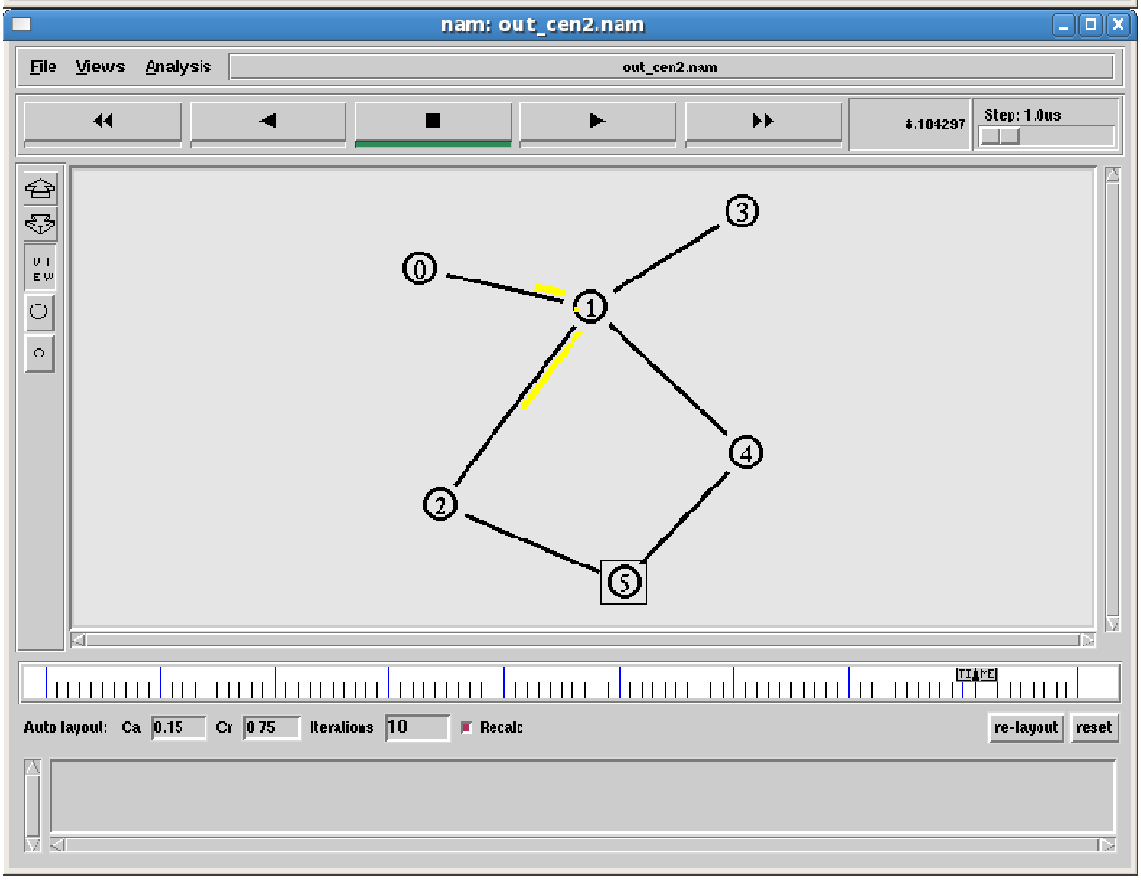
```
awk '{if($1 == "d") total += 1 } END {print "Total Pacotes Dropados:
          ",total}' out_cen1.tr

Total Pacotes Dropados:  2473

awk '{if($1 == "d") print $1,$2}' out_cen1.tr
```

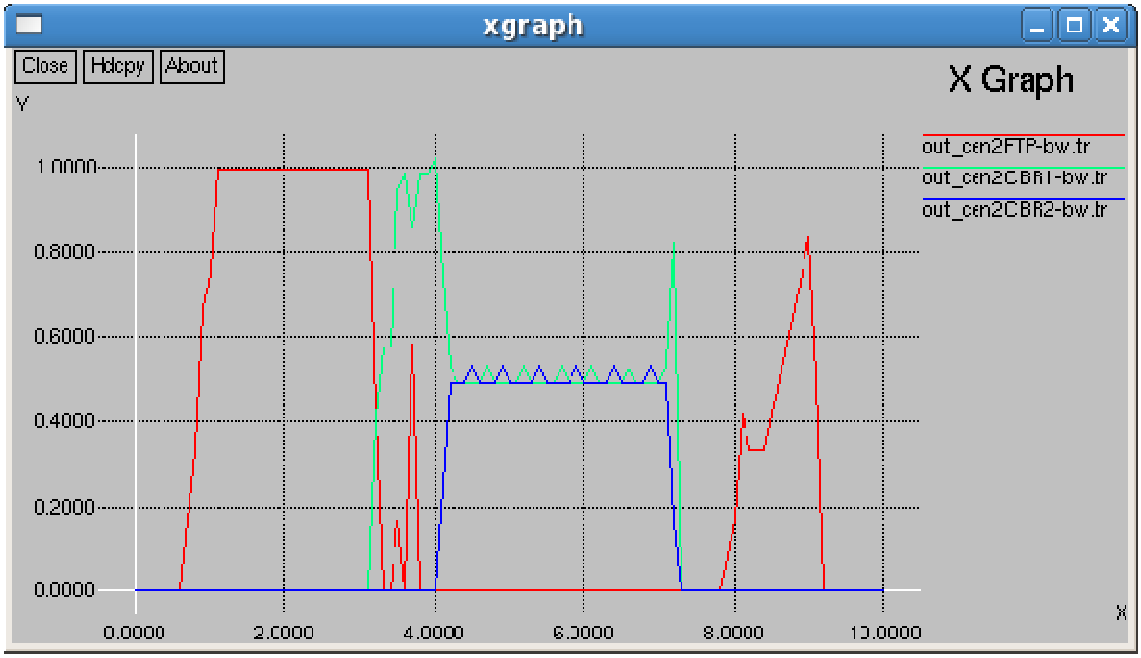

nam: out_cen2.nam
File
Views
Analysis
out_cen2.nam
1.194528
Step: 15.8us
0
1
2
3
4
5
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset


nam: out_cen2.nam
File
Views
Analysis
out_cen2.nam
3.098860
Step: 1.0us
0
1
2
3
4
5
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset


nam: out_cen2.nam
File
Views
Analysis
out_cen2.nam
3.401848
Step: 1.0us
0
1
2
3
4
5
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset


nam: out_cen2.nam
File
Views
Analysis
out_cen2.nam
4.139846
Step: 1.0us
0
1
2
3
4
5
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset

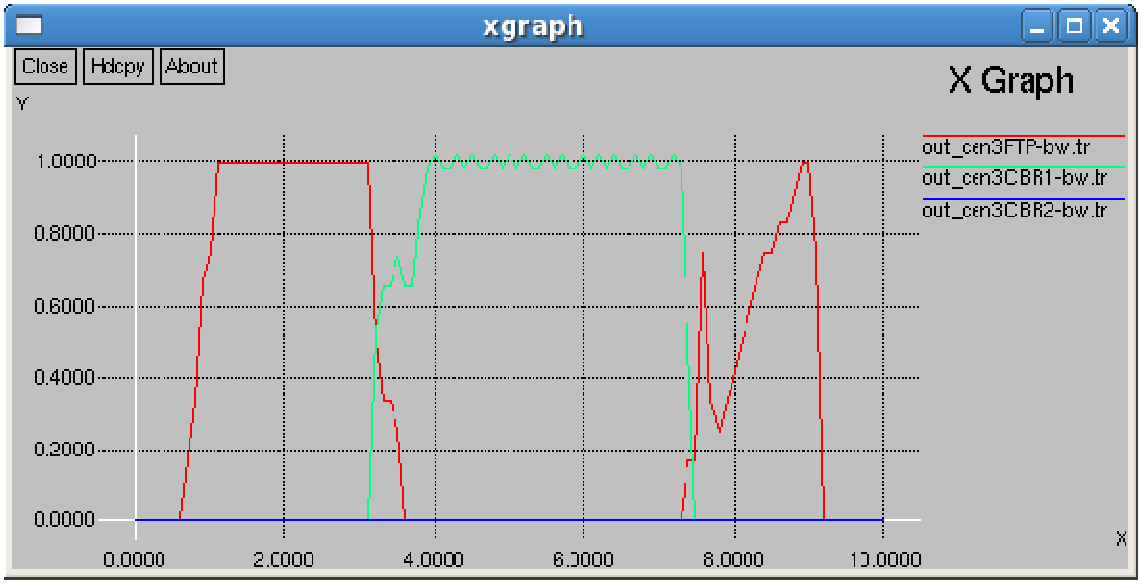

nam: out_cen2.nam
File
Views
Analysis
out_cen2.nam
4.104297
Step: 1.0us
0
1
2
3
4
5
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset


xgraph
Close
Hdcpy
About
X Graph
Y
1.0000
0.8000
0.6000
0.4000
0.2000
0.0000
0.0000
2.0000
4.0000
6.0000
8.0000
10.0000
X
out_cen2FTP-bw.tr
out_cen2CBR1-bw.tr
out_cen2CBR2-bw.tr

Cenário 3

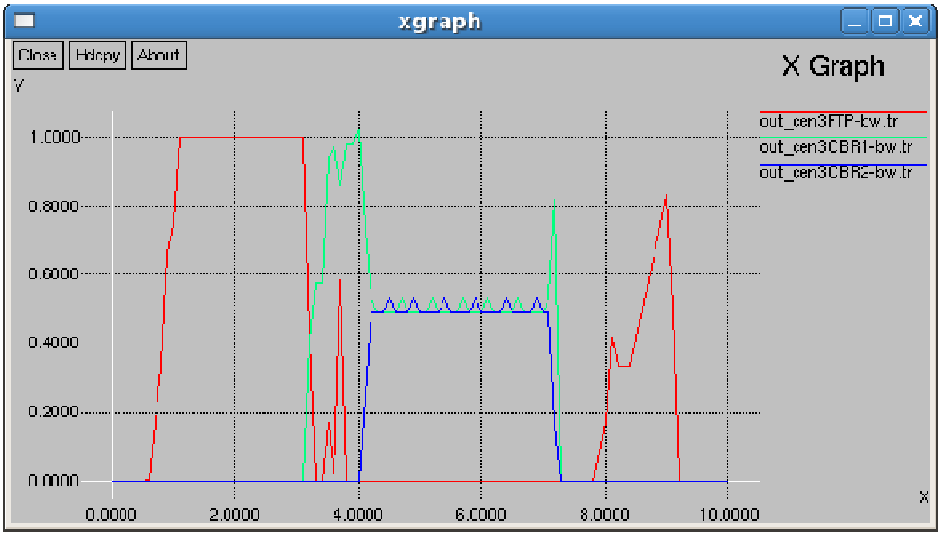
Este cenário estende o cenário 2 com algumas modificações. Basicamente é uma comparação entre os gráficos de largura de banda utilizadas pelas fontes de tráfego com as políticas de fila. São comparadas três políticas.

A primeira imagem abaixo apresenta o gráfico de largura de banda quando utilizado entre o n1 e n2 a política DROPTAIL. Esse tipo de filas é um simples FIFO que descarta o último pacote da fila quando ocorre um overflow. Não se preocupa com atraso, descartes ou vazão, apenas bota e tira da fila. Se a fila estiver cheia descarta o último pacote. Com esta política é possível observar que o tráfego TCP foi aos poucos sendo substituído pelo primeiro CBR. O segundo CBR se quer teve espaço para transmissão. Ao finalizar, no 7º segundo, inicia-se novamente o tráfego do FTP.



A próxima imagem abaixo apresenta o gráfico com o n1 e n2 com a política SFQ. O Stochastic Fair Queuing é uma modificação do esquema FQ (Fair Queuing), tentando sobrepor suas limitações. Uma função de hash é aplicada sobre o fluxo, determinando para qual fila ele é direcionado. Todos fluxos que apontam para uma mesma fila, serão tratados da mesma maneira, como se fossem o mesmo fluxo. Este esquema reduz o número de filas necessárias. A desvantagem desse esquema é que os fluxos que colidem são tratados injustamente. Contudo, as garantias são probabilísticas, dai o nome Estocástico. As filas são percorridas em estilo round robin, sem levar em conta o tamanho dos pacotes. Quando um novo pacote chega e não existem mais buffers disponíveis, o último pacote da fila mais longa é descartado.

Neste é possível observar que o tráfego é melhor dividido nas conexões UDP, como visto no gráfico. A largura máxima do canal é dividido contemplando as duas conexões CBR.



Na última imagem, é apresentado o gráfico da política RED - Random-Early Drop: O roteador pode detectar congestionamento através da análise do tamanho médio da fila. Assim ele pode notificar as conexões sobre o congestionamento, seja descartando o pacote ou marcando um bit nos cabeçalhos dos do pacotes. Ele irá notar o congestionamento quando o tamanho médio da fila ultrapassar um valor pré-definido de treshold, passando a marcar ou descartar os pacotes com uma certa probabilidade, a qual é função do tamanho médio da fila. Esse esquema é usado geralmente em conjunto com fluxos TCP, avisando o emissor quando a fila atinge o treshold.

Este é possível observar claramente o recurso do RED, que descarta o pacote ou marca um bit no cabeçalho do pacote, buscando diminuir o congestionamento. Mas também contemplando todas as fontes de tráfego.