

Algoritmos para Projeto de Redes de Telecomunicações: Software de Simulação NS-2

Programa de Pós-Graduação em Informática – PPGIA - PUCPR

Prof. Marcelo E. Pellenz
<http://www.ppgia.pucpr.br/~marcelo>
e-mail: marcelo@ppgia.pucpr.br

Tópicos

- Introdução
 - Simulador de Redes ns-2
 - Características
 - Funcionalidades
- Arquitetura do ns-2
- O simulador de redes *ns-2*
- Exemplos de simulação com o *ns-2*
- Estudos de caso

Introdução

- Simulador de Redes ns-2
 - Simulador de redes baseado em eventos discretos (Event Driven) para pesquisa em redes
 - Desenvolvido e distribuído pela Universidade da Califórnia (Berkeley):
 - The VINT project : Virtual InterNet Testbed
 - ISI (Information Sciences Institute - www.isi.edu)
 - University of Southern California (USC)
 - Financiamento:
 - Defense Advanced Research Projects Agency (DARPA)
 - National Science Foundation (NSF)

3

Introdução

- Características
 - Plataformas onde roda:
 - FreeBSD (desenvolvimento) , Linux, Solaris e Windows (com restrições)
 - As funcionalidades são oferecidas:
 - Diretamente na distribuição básica do NS
 - Contribuições (*patches*) que devem ser instaladas à parte
 - A distribuição do NS-2 é gratuita, inclusive o código fonte, que pode ser alterado para pesquisas específicas

4

Introdução

- Funcionalidades do NS-2
 - Simulador a nível de pacotes
 - Implementa camada de enlace e camadas superiores
 - Simulação de redes cabeadas e redes sem fio (wireless)
 - Implementa fontes de tráfego: [FTP](#), [HTTP](#), [Telnet](#), [CBR](#) e [VBR](#)
 - Implementa os protocolos de transporte: [TCP](#), [UDP](#)
 - Implementa protocolo de rede: [IP](#)

5

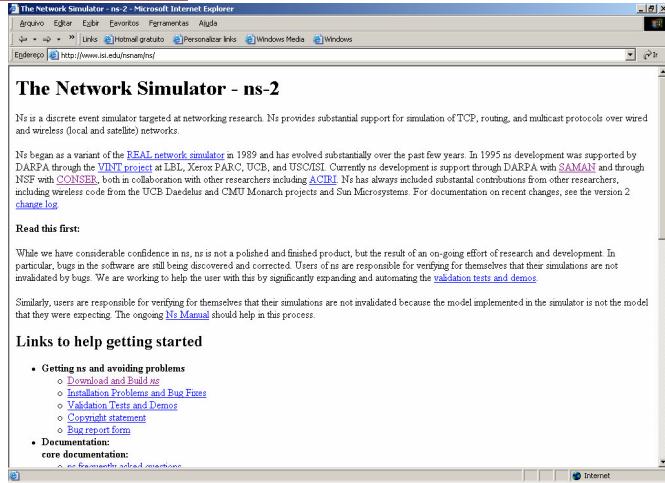
Introdução

- Funcionalidades do NS-2
 - Mecanismos para gerência de filas em roteadores:
 - [DropTail](#), [RED](#), [CBQ](#), [SFQ](#), [WFQ](#), [DRR](#)
 - Protocolos de roteamento
 - [Session](#), [DV](#), [LS](#), [multicast](#)
 - Implementa abordagens para QoS
 - [IntServ](#), [DiffServ](#), [MPLS](#), [QoS Routing](#)
 - Protocolos MAC para simulação de LANs
 - Comunicação sem fio
 - [WLAN](#), [Comunicação via Satélite](#), [GPRS](#), [Bluetooth](#), etc.

6

Introdução

- <http://www.isi.edu/nsnam/ns>

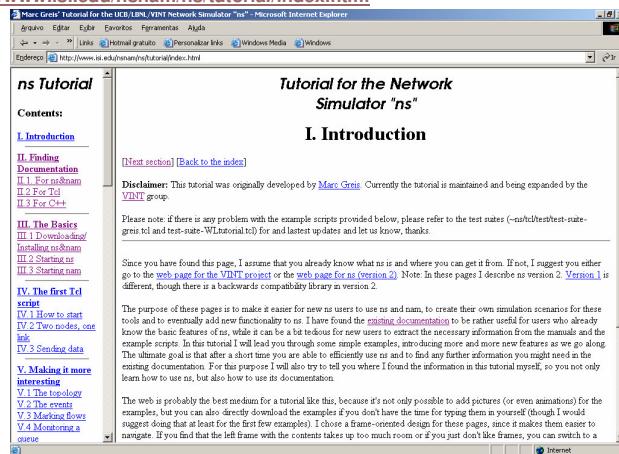


7

Introdução

- **Tutorial de Marc Greis**

<http://www.isi.edu/nsnam/ns/tutorial/index.html>

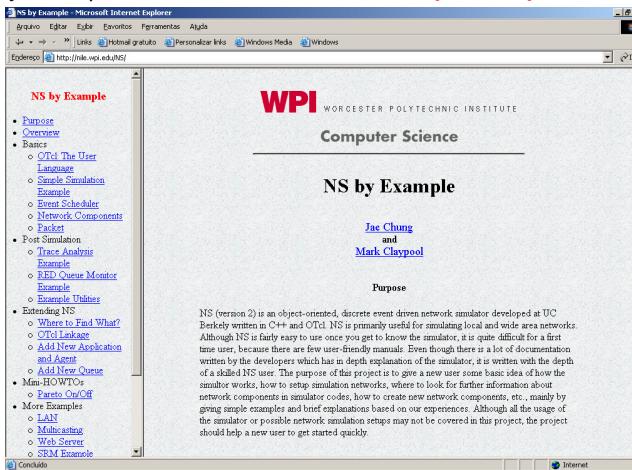


8

Introdução

- NS by Example

<http://nile.wpi.edu/NS/>



9

Network Simulator NS-2

- ARQUITETURA DO NS-2

- Combinação de C++ e Otcl
- Estrutura orientada a objetos
 - Evolução do NSv1 (objetos tcl/C++)
 - OTcl: extensão orientada a objetos do Tcl
- Separação de Controle e Dados
 - Operações de controle em Otcl (fácil configuração)
 - Os dados passam através dos objetos C++ (velocidade)
- Abordagem Modular

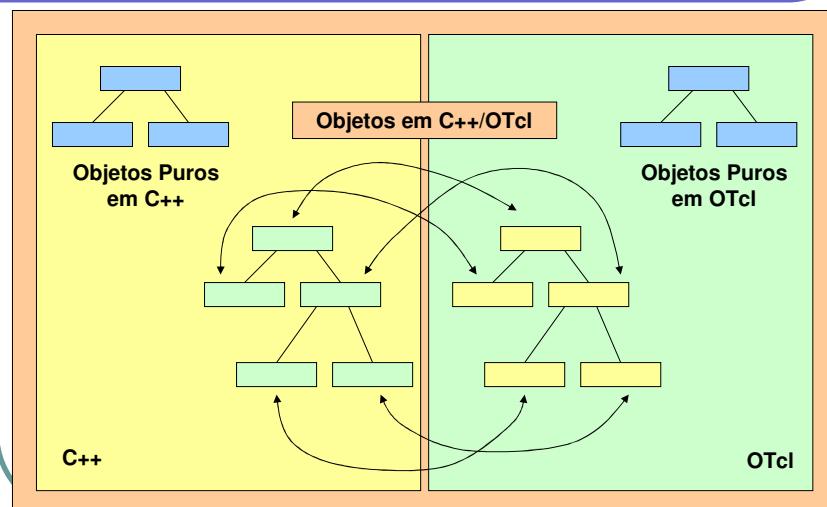
10

Modelo de programação do ns

- C++
 - Núcleo do simulador (eficiência)
 - Trata de eventos e pacotes
- Otcl
 - Interface para criar cenários através de *scripts*
 - É interpretada (os scripts de simulação não são compilados)
 - Provê flexibilidade
- Alguns objetos existem nas duas linguagens
- As duas linguagens tem interfaces entre si:
 - Pode-se chamar funções do C++ no Otcl (mais comum)
 - Pode-se chamar funções do Otcl no C++

11

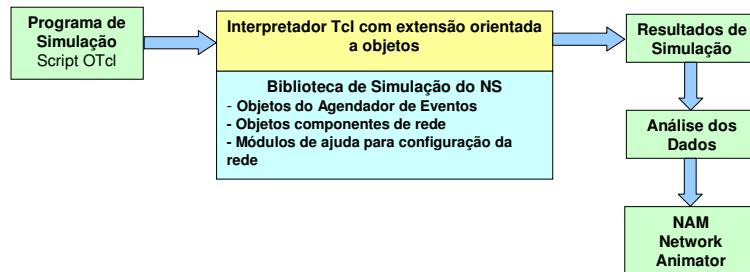
Otcl e C++



12

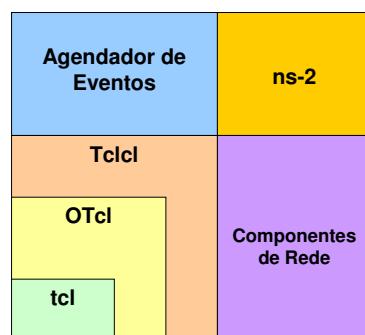
Network Simulator NS-2

- Estrutura do Simulador:



13

Network Simulator NS-2



14

Network Simulator NS-2

- Dualidade: OTcl e C++
- Interpretador tcl com extensões
 - Otcl: suporte para orientação a objetos
 - Tclcl: junção de C++ e otcl
 - Agendador de eventos discretos (event scheduler)
 - Evento:
 - Identificador único de pacote (ID)
 - Tempo de execução
 - Ponteiro para o objeto que manipula o evento
 - Componentes da rede

15

Resultados de Simulação

- O simulador pode ser configurado para gerar arquivos contendo os dados desejados
 - Opcionalmente o arquivo de *trace* pode ser processado para obter esses resultados
- Vazão de uma conexão
- Atraso e variação de atraso de pacotes
- Perda de pacotes
 - Nas filas: monitor de filas
 - Nos sistemas finais (fim-a-fim): agente LossMonitor
- Outros

16

Vantagens do NS-2

- Simulador padrão para a comunidade científica e acadêmica
- Simulador gratuito de código aberto
- Boa infra-estrutura para desenvolver programas
- Grande quantidades de protocolos e tecnologias existentes
- Oportunidade para estudar interações de protocolos em ambiente controlado
- Lista de discussão para dúvidas e discussões

17

Limitações do NS-2

- Limitação para tecnologias e protocolos sub-IP: ATM ?
- Curva de aprendizado: lenta
- Falta de documentação adequada
- Requer conhecimento de Tcl para criar cenários de simulação simples
- Requer conhecimento de C++ para estender o simulador
- Mais indicado para a plataforma Unix

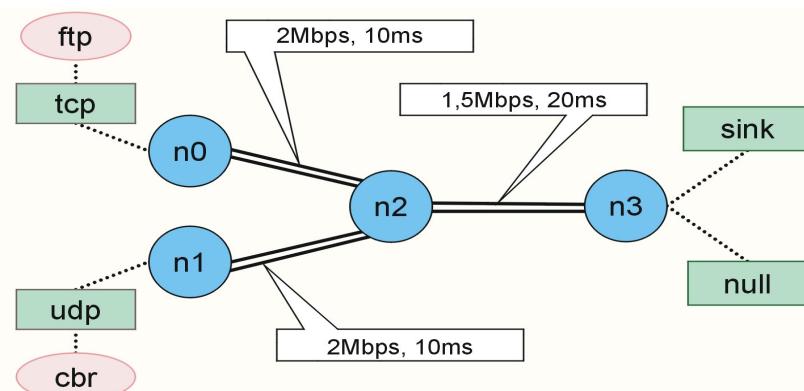
18

Componentes Básicos

- Nó (*node*)
 - Sistema final ou roteador
 - Máquina com implementação IP
 - Abstrai características da rede subjacente
- Enlace (*link*)
 - Interconectam dois nós
 - O ns só representa a camada IP
 - Enlaces são abstrações da interface física
- Agente (*agent*)
 - Entidade de transporte
 - Agentes: TCP/UDP
- Aplicação (*application*)
 - Não precisa simular a aplicação
 - Necessário apenas gerar tráfego

19

Componentes Básicos



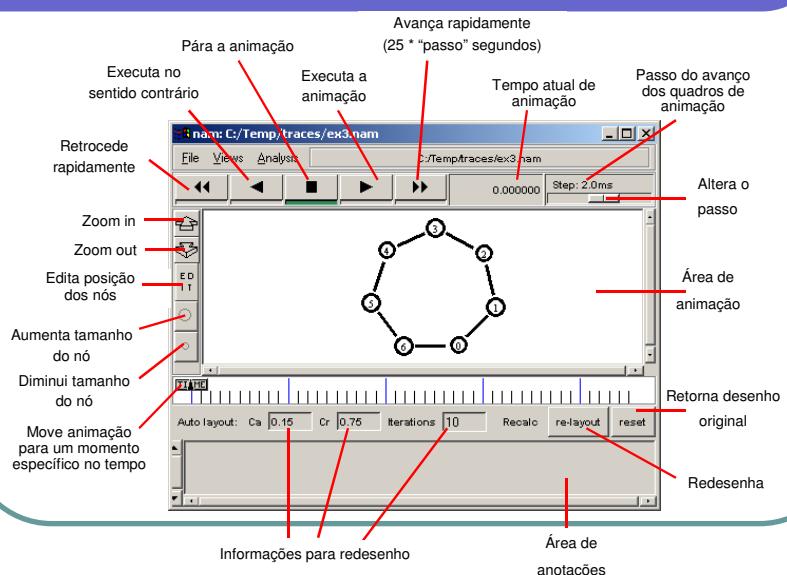
20

O Animador de Simulação *nam*

- Utilizado para compreender o que ocorreu em uma simulação passada
- Quando ativado, apresenta um console, que pode abrir várias animações
- Durante a simulação, o *ns* pode gerar um arquivo de trace para o *nam*
- Observações:
 - Arquivo de *trace* pode ficar muito grande (+100MB)
 - Simulação demora mais gerando o *trace*

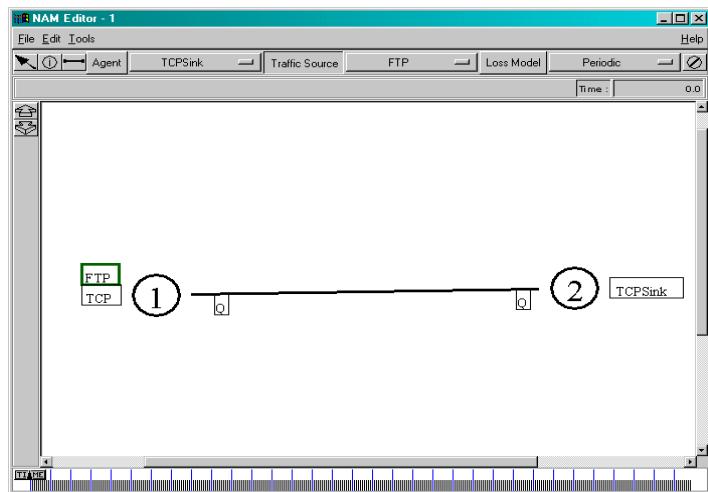
21

O Animador de Simulação *nam*



22

Editor do *nam* (limitado)



23

Uso do *ns* e *nam* no Ensino

- <http://www.isi.edu/nsnam/ns/edu>

Using ns and nam in Education

We are working to make ns-2 easier to use for networking education. We are targeting several audiences:

- ADVANCED UNDERGRADUATES should be able to use ns to compare network protocols. Our *nam* editor (released Sep. 2001) makes it possible to compare simple things without knowing Tcl.
- GRADUATE STUDENTS will do more advanced comparisons with C++ and C++ programming. (Simple "message parsing" C++ and Tcl modules are added to ns-2 to make this easier.)
- PROFESSORS can use ns and nam to animate networking principles in class. Our library of scripts includes existing demos, we welcome more!
- RESEARCHERS can use ns to investigate new networking concepts (as always).

the Network Simulator (ns) *the Network Animator (nam)* *the nam Graphical Editor*

ns is a public domain simulator boasting a rich set of tools together with its companion, nam, form a very powerful set of tools for teaching networking concepts. ns contains all the IP protocols typically covered in undergraduate and most graduate courses, and many experimental protocols contributed by us ever-expanding user base. With nam, these protocols can be visualized as animations (see some screen shots below).

This is the latest (Sept. 2001) addition to nam. With the nam editor, you no longer have to type TCL code to create animations. You can create your network topology and simulate various protocols and traffic sources by dragging the mouse.

Visit the [nam home page](#)

24

Network Simulator NS-2

- Estrutura dos Scripts NS
 - Criar o agendador de eventos (Instância do Simulador)
 - Configurações Iniciais (Habilita o *tracing*)
 - Ajustar opções gerais do ns e do nam
 - Configurar arquivo de trace do nam
 - Criar função de finalização
 - Criar Topologia da rede
 - Nós / Enlaces
 - Configurar roteamento
 - Inserir erros na transmissão

25

Network Simulator NS-2

- Estrutura dos Scripts NS
 - Cria protocolos agentes e aplicações (tráfego)
 - Criar agentes de transporte TCP e UDP
 - Criar aplicações geradoras de tráfego
 - Anexar agentes a nós e aplicações a agente
 - Conectar agentes nos sistemas finais
 - Inicializar o agendador de eventos
 - Iniciar transmissão de dados das aplicações
 - Finalizar transmissão de dados das aplicações
 - Executar simulação
 - Visualizar a animação com o nam
 - Analisar arquivos de trace

26

Network Simulator NS-2

- Instalação

- <http://www.isi.edu/nsnam/ns>
- <http://www-mash.cs.berkeley.edu/ns/ns-build.html>

- Modo Interativo

```
hercules:~> ns
% set ns [new Simulator]
_04
% $ns at 10 "puts \"Agenda Evento\""
1
% $ns at 15 "exit"
2
% $ns run
Agenda Evento
hercules:~>
```

27

Network Simulator NS-2

- Arquivo de Configuração

```
exemplo1.tcl
set ns [new Simulator]
$ns at 1 "puts \"Agenda Evento\""
$ns at 1.5 "exit"
$ns run
hercules:>> ns exemplo1.tcl
Agenda Evento
hercules:>>
```

28

Network Simulator NS-2

- Linguagem Script Tcl: A Linguagem do Usuário

```
c:\temp\ns> ns  
% set a 10  
10  
% set b $a  
10  
% set c [expr $a + $b]  
20  
% puts "$a $b $c"  
10 10 20  
% puts "[expr 1/60]"  
0  
% puts "[expr 1.0/60.0]"  
0.01666666666666666  
% puts "$a \t $b \t $c"  
10          10          20  
% set file1 [open dados.txt w]  
Filed285c8  
% puts $file1 "$a \t $b \t $c"  
% puts $file1 "$a \t $b \t $c"
```

```
% puts -nonewline $file1 "$a \t $b \t $c"  
% puts -nonewline $file1 "$a \t $b \t $c"  
% close $file1
```

29

Network Simulator NS-2

- Linguagem Script Tcl: A Linguagem do Usuário

```
exec xgraph data &  
  
if { expressao } {  
    < executa comandos >  
} else {  
    < executa comandos >  
}  
  
for { set i 0 } {$i < 5} {incr i} {  
    < executa comandos >  
}  
  
proc nomefuncao { par1 par2 ... } {  
    global var1 var2  
    < comandos >  
    return $resultado  
}
```

30

Network Simulator NS-2

- Básico Tcl: A Linguagem do Usuário

```
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}
```

Arquivo: [basic-tcl.tcl](#)

31

Network Simulator NS-2

- Básico OTcl:

Arquivo: [basic-otcl.tcl](#)

```
Class mom

mom instproc greet {} {
    $self instvar age_
    puts "Idade $age_"
}

Class kid -superclass mom

kid instproc greet {} {
    $self instvar age_
    puts "Idade $age_"
}

set a [new mom]
$a set age_ 45

set b [new kid]
$b set age_ 15

$a greet
$b greet
```

32

Network Simulator NS-2

- Exemplo 1

Arquivo: exemplo1.tcl

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Execute nam on the trace file
    exec nam out.nam &
    exit 0
}
```

33

Network Simulator NS-2

- Exemplo 1

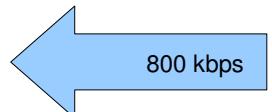
Arquivo: exemplo1.tcl

```
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```



34

Network Simulator NS-2

- Exemplo 1

Arquivo: exemplo1.tcl

```
#Create a Null agent (a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0

#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

35

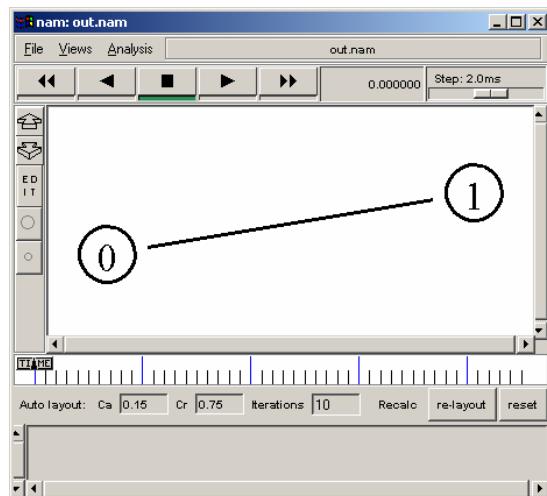
Network Simulator NS-2

The screenshot shows a Windows command prompt window titled 'C:\WINNT\system32\command.com'. The command entered is 'D:\users\NS>ns exemplo1.tcl'. The output shows:

```
D:\users\NS>ns exemplo1.tcl
D:\users\NS>
O volume na unidade D não tem nome.
O número de série do volume é 3C73-2888
Pasta de D:\users\NS
07/10/2004 20:06 <DIR> .
07/10/2004 20:06 <DIR> ..
20/07/2004 16:36 803 ptalho para NM.exe.lnk
20/07/2004 16:36 799 ptalho para NS.exe.lnk
28/07/2004 15:56 1.128 exemplo1.tcl
28/07/2004 15:56 1.919 exemplo2.tcl
28/07/2004 15:56 1.378 exemplo3.tcl
05/10/2004 14:30 1.298 filamn1.tcl
07/10/2004 20:06 1.133 filamn1k.tcl
28/07/2004 15:57 8.722.323 nglinstaller.exe
28/07/2004 15:57 1.571.979 nam.exe
29/09/2004 13:59 1.531.984 ns.exe
28/07/2004 15:56 2.292.916 out.nam
15/10/2004 10:14 5.291.336 out.tn
05/10/2004 15:20 914.515 qm.out
05/10/2004 15:20 2.568.821 tc1832.exe
11/09/2003 14:00 686.430 tracegraph202.zip
28/07/2004 15:57 16 arquivo(s) 25.669.933 bytes
2 pasta(s) 5.898.416.128 bytes disponíveis
D:\users\NS>nam out.nam
```

36

Network Simulator NS-2



37

Network Simulator NS-2

- Exemplo 1

Arquivo: [exemplo1.tcl](#)

- Identificar os instantes de início e fim da fonte de tráfego CBR
- Diferenciar tempo de transmissão e propagação
- Monitorar pacotes individuais
- Identificar disciplina de fila
- Alterar o tamanho dos pacotes CBR para 100 Bytes
- Alterar o tamanho dos pacotes CBR para 800 Bytes
- Agendar 2 novos eventos para a fonte CBR:
 - Término em 2 segundos
 - Início em 3 segundos

38

Network Simulator NS-2

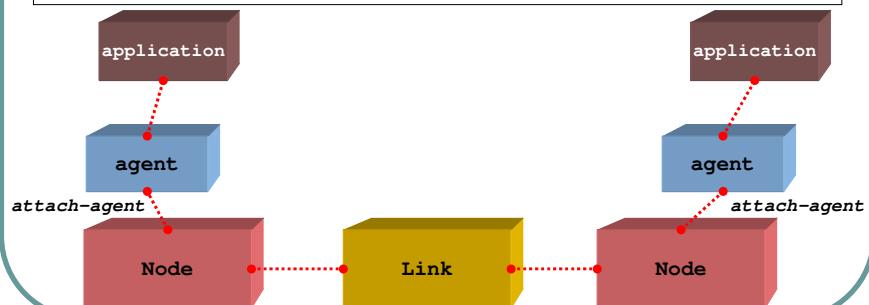
CRIANDO A REDE:

Nodes

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

Links and queuing

```
$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>  
<queue_type>: DropTail, RED, CBQ, FQ ...
```



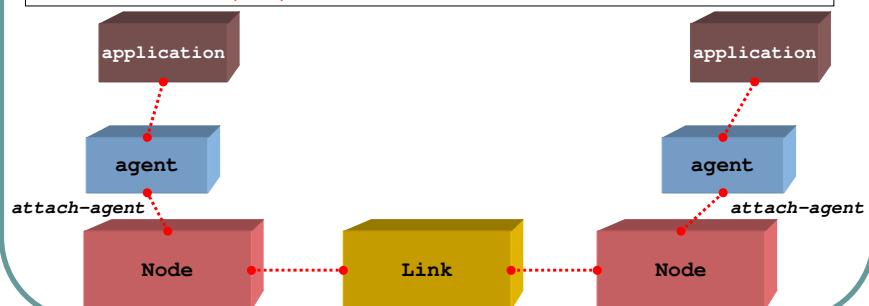
39

Network Simulator NS-2

CRIANDO CONEXÕES:

TCP

```
set tcp [new Agent/TCP]  
set tcpsink [new Agent/TCPSink]  
$ns attach-agent $n0 $tcp  
$ns attach-agent $n1 $tcpsink  
$ns connect $tcp $tcpsink
```

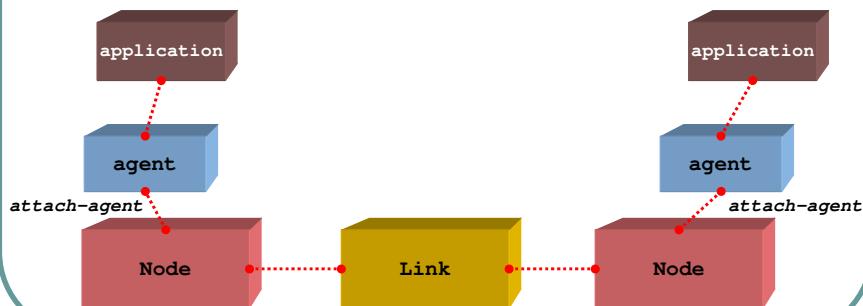


40

Network Simulator NS-2

CRIANDO TRÁFEGO SOBRE O TCP:
FTP

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



41

Network Simulator NS-2

INSERINDO ERROS:

Creating Error Module

```
set loss_module [new ErrorModel]
$loss_module set rate_ 0.01
$loss_module unit pkt
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
```

Inserting Error Module

```
$ns lossmodel $loss_module $n0 $n1
```

42

Network Simulator NS-2

TRACING:

Trace packets on all links

```
$ns trace-all [open test.out w]
```

```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src> <dst> <seqno> <aseqno>
+ 1 0 2 cbr 210 ----- 0 0.0 3.1 0 0
- 1 0 2 cbr 210 ----- 0 0.0 3.1 0 0
r 1.00234 0 2 cbr 210 ----- 0 0.0 3.1 0 0
```

Trace packets on all links in nam-1 format

```
$ns namtrace-all [open test.nam w]
```

43

Network Simulator NS-2

Exemplo 2

Arquivo: [exemplo2.tcl](#)

```
set ns [new Simulator]
# arquivos de trace
set f [open out.tr w]
set nf [open out.nam w]

$ns trace-all $f
$ns namtrace-all $nf

# criacao de alguns nós
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Criacao de alguns agentes
set udp0 [new Agent/UDP]
set null0 [new Agent/Null]
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
```

44

Network Simulator NS-2

- Exemplo 2

Arquivo: exemplo2.tcl

```
# Criacao de algumas aplicacoes
set cbr0 [new Application/Traffic/CBR]
set ftp [new Application/FTP]

# Criacao de enlaces
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

# anexando agentes
$cbr0 attach-agent $udp0
$ns attach-agent $n0 $udp0
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0

$ftp attach-agent $tcp
$ns attach-agent $n1 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
```

45

Network Simulator NS-2

- Exemplo 2

Arquivo: exemplo2.tcl

```
# Imprimindo algumas variáveis na saída
puts [$cbr0 set packet_size_]
puts [$cbr0 set rate_]

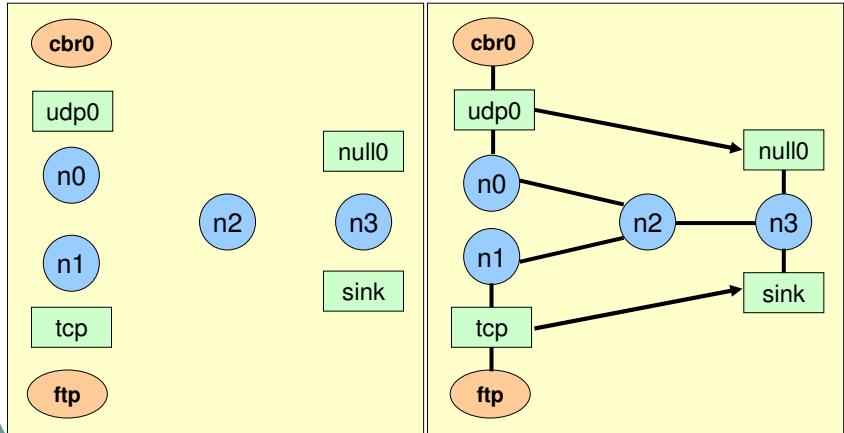
# escalonando algumas tarefas
$ns at 0.1 "$cbr0 start"
$ns at 0.5 "$ftp start"
$ns at 1.35 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n3 $sink"
$ns at 3.0 "finaliza"

proc finaliza {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    puts "Executando o nam..."
    exec nam out.nam &
    exit 0
}
# Finalmente, inicia a simulação
$ns run
```

46

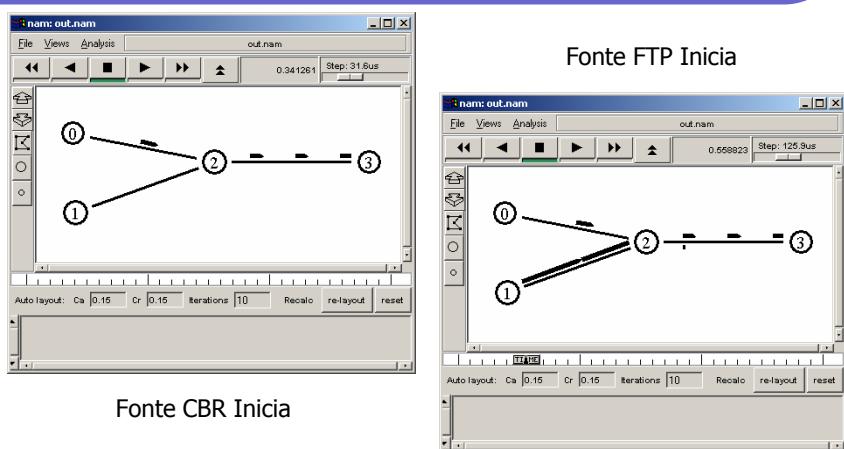
Network Simulator NS-2

- Objetos e Conexões



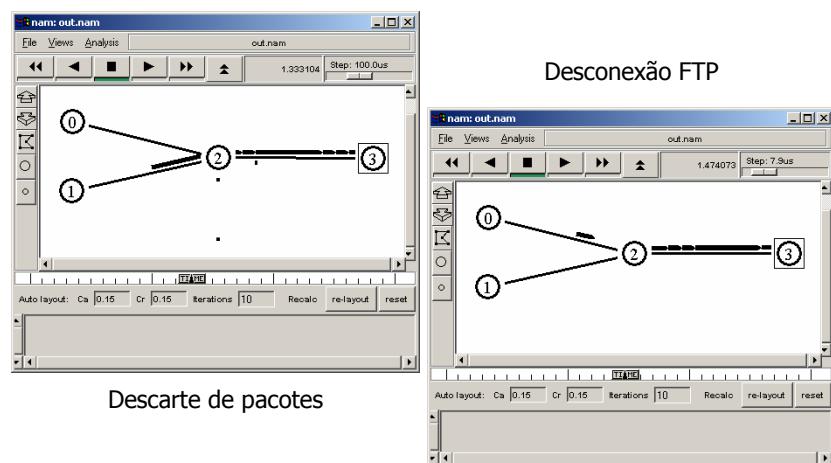
47

Exemplo 2 – Animação



48

Exemplo 2 – Animação



49

Exemplo 2 – Animação

- Exemplo 2

Arquivo: [exemplo2.tcl](#)

- Identificar os instantes de início e fim das fontes de tráfego CBR e FTP
- Identificar o *handshaking* da conexão TCP
- Monitorar pacotes individuais de ACK
- Identificar disciplina de fila nos nós da rede
- Verificar o processo de descarte no nó 2

50

Network Simulator NS-2

- Exemplo 3

Arquivo: exemplo3.tcl

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Execute nam on the trace file
    exec nam out.nam &
    exit 0
}
```

51

Network Simulator NS-2

- Exemplo 3

Arquivo: exemplo3.tcl

```
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ

$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

#Monitor the queue for the link between node 2 and node 3
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

52

Network Simulator NS-2

- Exemplo 3

Arquivo: exemplo3.tcl

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1
```



53

Network Simulator NS-2

- Exemplo 3

Arquivo: exemplo3.tcl

```
# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

#Connect the traffic sources with the traffic sink
$ns connect $udp0 $null0
$ns connect $udp1 $null0
```



54

Network Simulator NS-2

- Exemplo 3

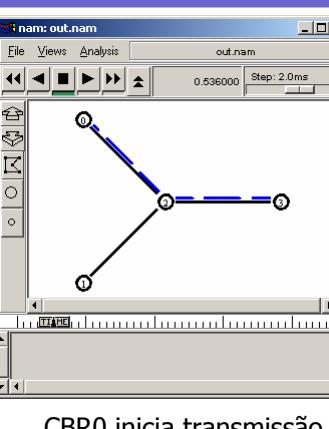
Arquivo: exemplo3.tcl

```
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

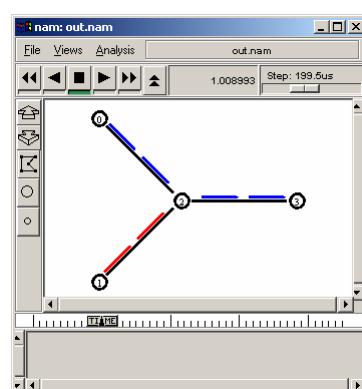
#Run the simulation
$ns run
```

55

Network Simulator NS-2

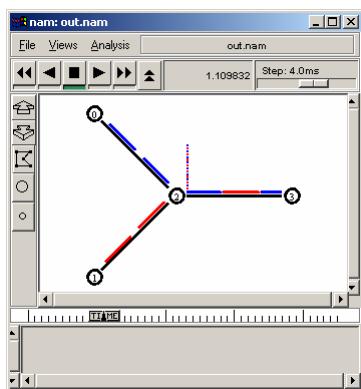


CBR1 inicia transmissão



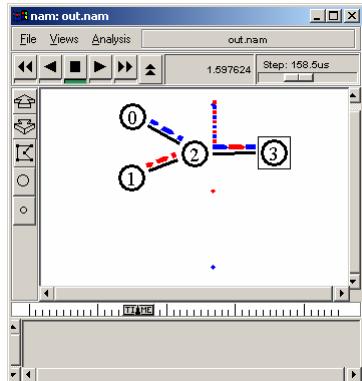
56

Network Simulator NS-2



Pacotes na fila

Descarte de Pacotes



57

Network Simulator NS-2

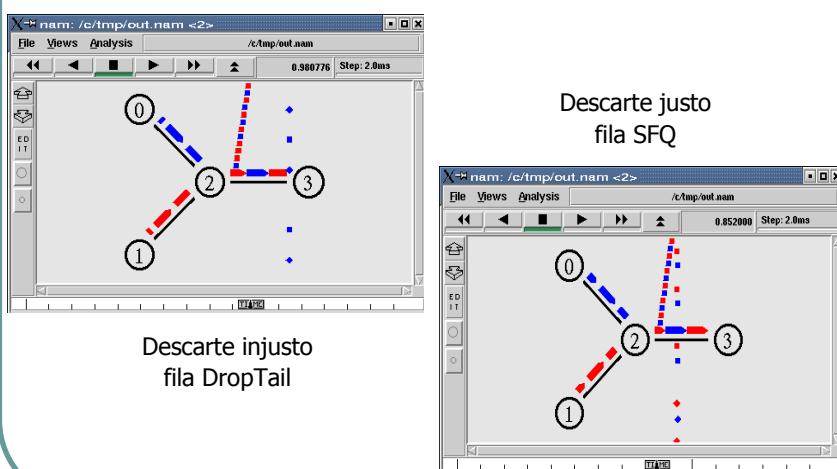
- Exemplo 3

Arquivo: [exemplo3.tcl](#)

- Identificar o tamanho *default* da fila no enlace 2-3
- Modificar e avaliar as estratégias de descarte no enlace 2-3:
 - DropTail
 - SFQ

58

Network Simulator NS-2



59

Exemplo 4

- Implementar o exemplo4.tcl a partir do exemplo3.tcl

```
#Open the NAM trace file
set nam_file [open out.nam w]
$ns namtrace-all $nam_file

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

```
#Simulation time
set SimTime 3.0
#Bottleneck link Bandwidth
set bw 10Mb
#Bottleneck link delay
set delay 20ms
#Bottleneck link queuetype
set queuetype DropTail
```

```
#Buffer Size
set BufferSize 50
#TCP packet size
set packetsize 1000
#TCP window size
set windowsize 80
```

60

Exemplo 4

```
#Set Queue size of the bottleneck link (n2-n3) to 20  
$ns queue-limit $n2 $n3 $BufferSize
```

```
#Create four nodes  
set n0 [$ns node]  
set n1 [$ns node]  
set n2 [$ns node]  
set n3 [$ns node]
```

```
#Connect the nodes - Create links between the nodes  
$ns duplex-link $n0 $n2 100Mb 2ms DropTail  
$ns duplex-link $n1 $n2 100Mb 2ms DropTail  
$ns duplex-link $n2 $n3 $bw $delay $queuetype
```

61

Exemplo 4

```
#Setup a TCP connection  
set agent_tcp [new Agent/TCP]  
  
#Attach TCP Agent to source node n0  
$ns attach-agent $n0 $agent_tcp  
set agent_sink [new Agent/TCPSink]  
  
#Attach a TCPSink Agent to destination node n3  
$ns attach-agent $n3 $agent_sink  
  
#Connect TCP Agent with TCPSink Agent  
$ns connect $agent_tcp $agent_sink  
  
#Flow Identity for TCP  
$agent_tcp set fid_ 1
```

62

Exemplo 4

```
#TCP parameters  
$agent_tcp set packet_size_ $packetsize  
$agent_tcp set window_ $windowsize  
  
#Setup a FTP traffic over TCP connection  
set traf_ftp [new Application/FTP]  
$traf_ftp attach-agent $agent_tcp
```

63

Exemplo 4

```
#Setup a UDP connection  
set agent_udp [new Agent/UDP]  
#Attach UDP Agent to source node n1  
$ns attach-agent $n1 $agent_udp  
set agent_null [new Agent/Null]  
#Attach a Null Agent to destination node n3  
$ns attach-agent $n3 $agent_null  
#Connect UDP Agent with NULL Agent  
$ns connect $agent_udp $agent_null  
#Flow Identity for UDP  
$agent_udp set fid_ 2  
#Setup a CBR traffic over UDP connection  
set traf_cbr [new Application/Traffic/CBR]  
$traf_cbr attach-agent $agent_udp
```

64

Exemplo 4

```
#CBR parameters  
$trat_cbr set packet_size_ 1000  
$trat_cbr set rate_ 4Mb  
$ns at 0.0 "$ns trace-queue $n2 $n3 $trace_file"
```

- Verifique o algoritmo *slow start* do TCP usando o NAM

65

Network Simulator NS-2

- Inicialização e Término da Simulação:

```
set ns [new Simulator]  
  
#open the trace file  
set file1 [open out.tr w]  
$ns trace-all $file1  
  
#open the NAM trace file  
set file2 [open out.nam w]  
$ns namtrace-all $file2  
  
#define a finish procedure  
proc finish {} {  
    global ns file1 file2  
    $ns flush-trace  
    close $file1  
    close $file2  
    exec nam out.nam &  
    exit 0  
}  
  
$ns at 125.0 "finish"  
$ns run
```

As diretivas *trace-all* e
namtrace-all
podem gerar arquivos
bastante grandes

66

Network Simulator NS-2

- Estrutura do Arquivo de Trace:

Event	Time	From Node	To Node	Pkt Type	Pkt Size	Flags	Fid	Src addr	Dst addr	Seq num	Pkt id
-------	------	-----------	---------	----------	----------	-------	-----	----------	----------	---------	--------

- Event: tipo de evento (r,+,-,d)
- Time: tempo em que o evento ocorreu
- From node: nó de entrada do enlace onde o evento ocorreu
- To node: nó de saída do enlace onde o evento ocorreu
- Pkt Type: tipo de pacote (TCP, UDP)
- Pkt Size: tamanho do pacote
- Flags
- Fid (flow id): identificação de fluxo para IPv6
- Src addr (source address): node.port
- Dst addr (destination address): node.port
- Seq num (sequence number): número de seqüência dos pacotes da camada de rede
- Pkt id: identificação única para cada pacote

67

Network Simulator NS-2

- Exemplo de dados do arquivo de trace:

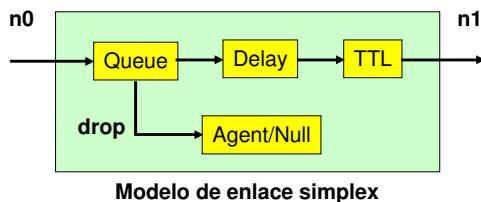
```
+ 0.1 1 2 cbr 1000 ----- 2 1.0 5.0 0 0
- 0.1 1 2 cbr 1000 ----- 2 1.0 5.0 0 0
r 0.114 1 2 cbr 1000 ----- 2 1.0 5.0 0 0
+ 0.114 2 3 cbr 1000 ----- 2 1.0 5.0 0 0
- 0.114 2 3 cbr 1000 ----- 2 1.0 5.0 0 0
r 0.240667 2 3 cbr 1000 ----- 2 1.0 5.0 0 0
+ 0.240667 3 5 cbr 1000 ----- 2 1.0 5.0 0 0
- 0.240667 3 5 cbr 1000 ----- 2 1.0 5.0 0 0
r 0.286667 3 5 cbr 1000 ----- 2 1.0 5.0 0 0
+ 0.9 1 2 cbr 1000 ----- 2 1.0 5.0 1 1
- 0.9 1 2 cbr 1000 ----- 2 1.0 5.0 1 1
r 0.914 1 2 cbr 1000 ----- 2 1.0 5.0 1 1
+ 0.914 2 3 cbr 1000 ----- 2 1.0 5.0 1 1
- 0.914 2 3 cbr 1000 ----- 2 1.0 5.0 1 1
+ 1 0 2 tcp 40 ----- 1 0.0 4.0 0 2
- 1 0 2 tcp 40 ----- 1 0.0 4.0 0 2
r 1.01016 0 2 tcp 40 ----- 1 0.0 4.0 0 2
+ 1.01016 2 3 tcp 40 ----- 1 0.0 4.0 0 2
- 1.01016 2 3 tcp 40 ----- 1 0.0 4.0 0 2
```

68

Network Simulator NS-2

- Definição do conjunto de **enlaces** e **nós** da Rede:
 - No NS a fila de saída de um nó é implementada como parte do enlace de saída

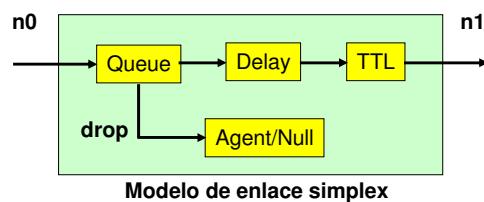
```
set n0 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n2 10Mb 10ms DropTail
```



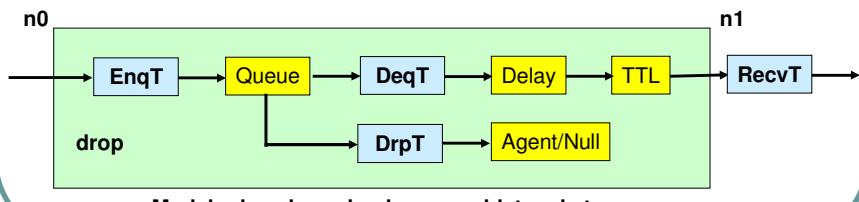
Modelo de enlace simplex

69

Network Simulator NS-2



Modelo de enlace simplex

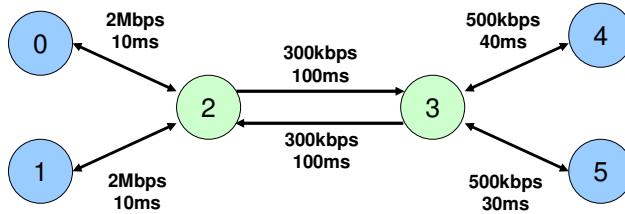


Modelo de enlace simplex com objetos de trace

70

Exercício

- Tráfego: FTP sobre TCP + CBR sobre UDP
- Executar o arquivo de simulação: `ex1.tcl`



71

Exercício

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
    global ns file1 file2
    $ns flush-trace
    close $file1
    close $file2
    exec nam out.nam &
    exit 0
}
```

72

Exercício

```
#Create six nodes
set n0 [\$ns node]
set n1 [\$ns node]
set n2 [\$ns node]
set n3 [\$ns node]
set n4 [\$ns node]
set n5 [\$ns node]

#Create links between the nodes
\$ns duplex-link \$n0 \$n2 2MB 10ms DropTail
\$ns duplex-link \$n1 \$n2 2MB 10ms DropTail
\$ns simplex-link \$n2 \$n3 0.3Mb 100ms DropTail
\$ns simplex-link \$n3 \$n2 0.3Mb 100ms DropTail
\$ns duplex-link \$n3 \$n4 0.5Mb 40ms DropTail
\$ns duplex-link \$n3 \$n5 0.5Mb 30ms DropTail

#Give node position (for NAM)
\$ns duplex-link-op \$n0 \$n2 orient right-down
\$ns duplex-link-op \$n1 \$n2 orient right-up
\$ns simplex-link-op \$n2 \$n3 orient right
\$ns simplex-link-op \$n3 \$n2 orient left
\$ns duplex-link-op \$n3 \$n4 orient right-up
\$ns duplex-link-op \$n3 \$n5 orient right-down

#Set Queue Size of link (n2-n3) to 10
\$ns queue-limit \$n2 \$n3 10
```

73

Exercício

```
#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
\$ns attach-agent \$n0 Stcp
set sink [new Agent/TCPSSink/DelAck]
\$ns attach-agent \$n4 \$sink
\$ns connect Stcp \$sink
\$tcp set fid_ 1
\$tcp set window_ 8000
\$tcp set packetSize_ 552

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
\$ftp attach-agent \$tcp
\$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
\$ns attach-agent \$n1 Sudp
set null [new Agent/Null]
\$ns attach-agent \$n5 \$null
\$ns connect Sudp \$null
\$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
\$cbr attach-agent \$udp
\$cbr set type_ CBR
\$cbr set packet_size_ 1000
\$cbr set rate_ 0.01mb
\$cbr set random_ false
```

74

Exercício

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"

# next procedure gets two arguments: the name of the
# tcp source node, will be called here "tcp",
# and the name of output file.

proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "Now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file"
$ns at 0.1 "plotWindow $tcp $winfile"

$ns at 125.0 "finish"
$ns run
```

75