

Universidade do Minho




Sistemas Telemáticos

O Network Simulator 2 (NS2)

António Costa <costa@di.uminho.pt>

Grupo de Comunicações por Computador
Departamento de Informática
Universidade do Minho




Horário de Atendimento:

- Quarta-Feira, 15h – 18h
- Terça-Feira, 10h – 13h

GCOM-DI-UM
1
ST 2004, A.Costa

Universidade do Minho




Sumário

- Introdução ao Network Simulator 2
- Componentes do NS2
- Fundamentos da simulação por eventos
- Arquitectura do NS2
- Utilização do NS2
- Tcl e Otcl
- Um exemplo completo
- Como funciona internamente o NS2
- Geradores de Topologias

GCOM-DI-UM
2
ST 2004, A.Costa

Universidade do Minho




Referências

Parte desta apresentação é baseada no seguinte material disponível on-line:

- NS2 Tutorial
 - Haobo Yu and Nader Salehi, USC/ISI, {haoboy,salehi}@isi.edu
- IPAM Tutorial: Network Modeling and Traffic Analysis with NS-2
 - John Heidemann (USC/ISI) and Polly Huang (ETH-Zurich)
- NS2 Tutorial
 - Kevin Fall (LBL) and Kannan Varadhan (USC/ISI)
- Manual do Network Simulator 2, VINT Project,
 - Kevin Fall (LBL) and Kannan Varadhan (USC/ISI)

GCOM-DI-UM
3
ST 2004, A.Costa

Universidade do Minho




Network Simulator 2

- Um simulador discreto por eventos...
 - Projectado para a simulação de redes e protocolos:
 - Nível 2 e superiores do modelo de referência...
 - Inclui suporte para:
 - Redes com fios e redes sem fios (incluindo ligações via satélite)
 - Protocolos da família TCP/IP:
 - IP e IP *multicast*, TCP e UDP
 - protocolos de encaminhamento *unicast* e *multicast*
 - Aplicações: FTP, Telnet, Web, etc...
 - Permite simular falhas de nós e ligações
 - Gera estatísticas, *traces*, etc. e saídas para visualizador
 - Permite diferentes níveis de abstracção...

GCOM-DI-UM
4
ST 2004, A.Costa

Universidade do Minho



Network Simulator 2


- Principais objectivos do NS2:
 - Plataforma para o **ensino** e a **investigação** na área de redes de computadores...
 - Projecto e estudo de protocolos novos ou já existentes
 - Comparação de diferentes protocolos
 - Principais áreas de investigação em que tem sido usado:
 - QoS: Intserv/Diffserv
 - Multicast (encaminhamento, multicast fiável)
 - Transporte (TCP, Controlo de congestão)
 - Aplicações (Web Caching e Multimedia)

São estes os “objectivos” para Sistemas Telemáticos!

- Também pode ser útil no planeamento...

GCOM-DI-UM
5
ST 2004, A.Costa

Universidade do Minho



Network Simulator 2

- Ambiente aberto que facilita a *colaboração*:
 - Código de distribuição gratuita
 - Todos os Unix e (alguns) Windows
 - Alguns itens novos só no FreeBSD (Exemplo: emulação)

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

- Alternativas
 - Experimentação em laboratório
 - Detalhes operacionais, mas a uma escala mais limitada
 - Nem sempre é possível...
 - Estudos analíticos
 - Ausência dos detalhes e do comportamento dinâmico...
 - Outros simuladores (comerciais ou não)
 - Questão de popularidade...

GCOM-DI-UM
6
ST 2004, A.Costa

Universidade do Minho

História e Estado Actual

- 1988: Columbia University: NEST (bancada de simulação)
- 1989: Cornell University: REAL (simulador de redes...)
- 1989: L. Berkeley Laboratory, NS-1 (baseado no REAL)
- 1995: NS-2, Projecto VINT (Virtual InterNetwork Testbed),
 - Consórcio: LBL, Xerox PARC, UCB, USC/ISI
- A versão actual ns-2
 - 2.27 de Janeiro 2004, mantida pela USC/ISI
 - Um nova versão a cada seis meses...
 - Números:
 - Tudo são 250 MB (inclui Tcl/tk, Otcl, tclCL, ns2, nam, xgraph, etc)
 - 100K linhas de C++, 70K linhas de Otcl, 30K linhas de testes, e 20K linhas de documentação

Os linux do LabCom estão com a versão 2.1b9a na /usr/local/bin/ns

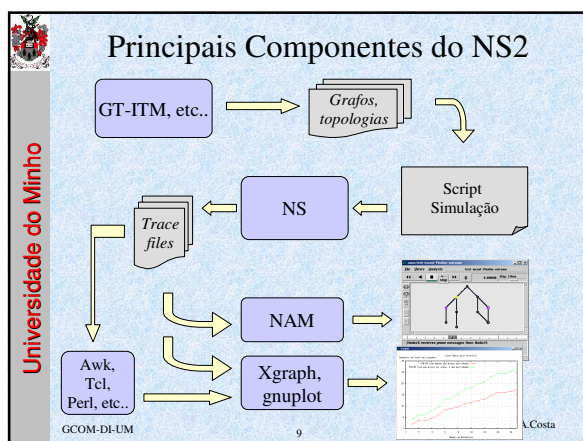
GCOM-DI-UM
7
ST 2004, A.Costa

Universidade do Minho

Principais Componentes do NS2

- NS** (o *Network Simulator* propriamente dito)
- NAM** (*Network AniMator*)
 - visualizador/animador dos resultados do NS (ou outros)
 - Algumas versões mais recentes permitem **editar** topologias e cenários de simulação...
- Pré-processadores:
 - Geradores de tráfego...
 - Geradores de topologias (INET, GT-ITM, TIERS, BRITE)
- Pós-processadores:
 - Analísadores dos ficheiros de *trace*
 - (scripts Awk, Perl ou Tcl)
 - Analísadores gráficos:
 - Xgraph, gnuplot, nam, etc...

GCOM-DI-UM
8
ST 2004, A.Costa



Universidade do Minho

Simulação discreta por eventos

- Modelar o mundo real com “eventos”**
 - Simulador mantém uma lista de “eventos”
 - Processa a lista de “eventos”:
 - Obtém próximo evento e executa-o, enquanto houver eventos...
 - Cada “evento” acontece num determinado instante de tempo *virtual* (tempo simulado)...
 - A execução de cada “evento” consome tempo *real*... mas que se não reflecte (normalmente) no tempo *virtual*...

O NS usa um modelo muito simples:

- uma única *thread* de controlo...
- não há preocupações de *locks* ou *race conditions*...

GCOM-DI-UM
10
ST 2004, A.Costa

Universidade do Minho

Simulação discreta por eventos

- Um exemplo: dois modelos**

Modelação simples usando uma fila de espera:

Evento 1, $t = 1$, Nó A faz *enqueue* do pacote na LAN

Evento 2, $t = 1.01$, LAN faz *dequeue* do pacote e provoca recepção no nó B.

Modelação mais detalhada do CSMA/CD:

Evento 1, $t = 1$, Nó A envia pacote para a sua placa de rede

- A placa de rede do nó A inicia detecção portadora...

Evento 2, $t = 1.005$, Placa de rede detecta meio livre e inicia a transmissão do pacote...

Evento 3, $t = 1.006$, Placa de rede do nó B inicia recepção do pacote

Evento 4, $t = 1.01$, Placa do nó B acaba de receber o pacote

- Placa do nó B entrega pacote à aplicação...

GCOM-DI-UM
11
ST 2004, A.Costa

Universidade do Minho

Simulação discreta por eventos

- O modelo certo é aquele que melhor se ajusta aos objectivos da simulação...
 - A simulação exige **sempre** um certo grau de abstracção da realidade...
- Mais detalhe, implica sempre mais recursos...
 - Mais gastos de memória, maior tempo de execução...
 - O que pode significar topologias menos complexas e tempos de simulação mais pequenos...

Objectivo do NS: suportar múltiplos níveis de abstracção

GCOM-DI-UM
12
ST 2004, A.Costa

Universidade do Minho

Arquitectura do NS2

- Metodologia orientada aos objectos
 - Abordagem modular
- Usa *duas* linguagens de programação:
 - C++ e OTcl (ambas *object-oriented*)
- Objectivos:
 - Conseguir reunir o melhor das duas linguagens para obter “Escalaabilidade” e “Extensibilidade”
 - Separação entre “Controlo” e “Dados”
 - Processamento ao nível do pacote em C++:
 - Execuções mais rápidas... para as tarefas mais frequentes, e que manipulem grandes quantidades de informação...
 - Operações de Controlo em OTcl:
 - Mais fácil e rápido de escrever e de fazer alterações... para configurações, mudanças ocasionais e definição da simulação...

GCOM-DI-UM

13

ST 2004, A.Costa

Universidade do Minho

Arquitectura do NS2

- Vantagens desta arquitectura:
 - bom compromisso entre velocidade de execução e tempo de escrita de código...
 - Linguagem de configuração poderosa e bem documentada
 - Consegue-se reutilização de código e maior facilidade na manutenção...
- Desvantagens:
 - Duas linguagens para aprender...
 - *Debug* mais complexo...
 - Exige planeamento cuidadoso da modularidade...

NS é complexo ! A curva de aprendizagem do NS é longa!

GCOM-DI-UM

14

ST 2004, A.Costa

Universidade do Minho

Arquitectura do NS2

A Dualidade Otcl/C++

GCOM-DI-UM

15

ST 2004, A.Costa

Universidade do Minho

Arquitectura do NS2

GCOM-DI-UM

16

ST 2004, A.Costa

- Por vezes é difícil encontrar o compromisso certo...
- A tentação de apostar numa só linguagem é grande...
- Pode-se sempre ajustar a granularidade à posteriori, migrando métodos do Otcl para o C++ (ou vice-versa)

Universidade do Minho

Arquitectura do NS2

Componentes de Redes e protocolos

TclCL

OTcl

Tcl

C/C++

Escalaçador de Eventos

ns-2

O “utilizador” do NS precisa conhecer o Tcl (e suas extensões) para escrever as simulações (scripts de simulação)

GCOM-DI-UM

17

ST 2004, A.Costa

- Tcl e C++ na base do NS
- OTcl (extensão para objectos)
- TclCL:
 - Permite a ligação C++/Otcl
 - Torna possível partilha de métodos e de variáveis entre as duas linguagens
- Escalonador de eventos
- Componentes de redes e protocolos:
 - Nível 2 e superiores
 - Emulação

Universidade do Minho

Utilização do NS2: *modo interactivo*

```

costa@polo[~]$ ns
% set ns [new Simulator]
_o4
% $ns at 1.0 "puts \"Ola Mundo!\""
1
% $ns at 2.0 "exit"
2
% $ns run
Ola Mundo!
costa@polo[~]$
            
```

GCOM-DI-UM

18

ST 2004, A.Costa

Qual o tempo de simulação? E de execução?

Universidade do Minho

Utilização do NS2: *modo batch*

```

simple.tcl
set ns [new Simulator]
$ns at 1.0 "puts \"Ola Mundo!\""
$ns at 2.0 "exit"
$ns run
costa@polo[~]$ ns simple.tcl
Ola Mundo!
costa@polo[~]$

```

GCOM-DI-UM
19
ST 2004, A.Costa

Universidade do Minho

Tcl

Variáveis e expressões:

```

set a 4
set b [expr $a*$a]
puts "Quadrado de $a e $b"
unset a b

```

Controlo de fluxo:

```

set mesada(jan) 100
...
set mesada(dez) 200
set total 0
set n 0
foreach mes {jan fev mar dez} {
    set total [expr $total+$mesada($mes)]
    incr n
}

```

```

set f [open "file.txt" r]
while { [gets $f line] >= 0 } {
    puts $line
}

```

```

set a { a b c d e f }
for { set i 0 } { $i < [lindex $a] } { incr i } {
    puts "Elemento $i tem valor [lindex $a $i]"
}

```

GCOM-DI-UM
20
ST 2004, A.Costa

Universidade do Minho

Tcl

Procedimentos:

```

proc fac x {
    if { $x <= 1 } {
        return 1
    }
    expr $x * [fac [expr $x-1]]
}

```

Variáveis locais e globais:

```

set f1 [open "file" r]
set f2 [open "file" w]
proc finish { } {
    global f1 f2
    set x "Terminando..."
    close $f1
    close $f2
    puts $x
}

```

Passagem por referência:

```

proc copia { src dest } {
    upvar $dest d
    set d $src
}

```

GCOM-DI-UM
21
ST 2004, A.Costa

Universidade do Minho

OTcl

- Classes são objectos com suporte para herança
- Analogias com o C++:

- C++ admite uma única declaração
- Construtor e destrutor C++
- this**
- Métodos *virtual*
- Métodos rescritos podem ser invocados explicitamente...
- Variáveis estáticas C++
- Suporta herança múltipla...

- OTcl atribue métodos a objectos
- Métodos *init* e *destroy*
- \$self**
- Todos os métodos são "virtuais"
- Invocação implícita dos métodos rescritos: **\$self next**
- Variáveis de classe...
- Suporta herança múltipla...

GCOM-DI-UM
22
ST 2004, A.Costa

Universidade do Minho

OTcl

```

Class Pessoa
# construtor
Pessoa instproc init {nome id}{
    $self instvar nome_ idade_
    set nome_ $nome
    set idade_ $id
}
# metodo "fala"
Pessoa instproc fala {}{
    $self instvar idade_
    $self instvar nome_
    puts "Chamo-me $nome_ e tenho $idade_ anos."
}
# subclass que rescreve metodo "fala"
Class Filho -superclass Pessoa
Filho instproc fala {} {
    $self next
    puts "Ainda sou crianca."
}

```

```

# criar a mae e o filho...
set mae [new Pessoa "Ana" 40]
set filho [new Filho "Jose" 8]

# filho faz anos...
$filho set idade_ 9

# falamos os dois...
$mae fala
$filho fala

```

GCOM-DI-UM
23
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

- Lista de tarefas

1. Criar o escalonador de eventos...
2. Activar as opções de *trace* para produzir os *outputs* desejados
3. Criar a topologia da rede
4. Activar o encaminhamento (unicast e multicast)
5. Introduzir erros e falhas nas ligações
6. Criar conexões de transporte (TCP e/ou UDP)
7. Transmitir dados entre as aplicações...

GCOM-DI-UM
24
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

1. Criar o escalonador de eventos:

```
set ns [new Simulator -multicast]
```

- Escalonar eventos:

```
$ns at <time> <event>
```
- Iniciar a execução no escalonador

```
$ns run
```
- Outras:

```
$ns cancel <event>
$ns use-scheduler <type>
$ns after <delay> <event>
$ns now
```

Qualquer comando
ns/tcl

GCOM-DI-UM
25
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

2. Activar as opções de *trace* desejadas:

- Todos os pacotes em todos os links:

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

```
<event> <time> <from> <to> <pkt> <size> -- <fid> <src> <dst> <seq> <attr>
+ 1 0 2 chr 210 ..... 0 0.0 3.1 0 0
- 1 0 2 chr 210 ..... 0 0.0 3.1 0 0
r 1.00234 0 2 chr 210 ..... 0 0.0 3.1 0 0
```
- Todos os *traces* mas no formato NAM (para visualizador):

```
$set nf [open out.nam w]
$ns namtrace-all $nf
```
- Apenas numa ligação

```
$ns trace-queue $n0 $n1
$ns namtrace-queue $n0 $n1
```

GCOM-DI-UM
26
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

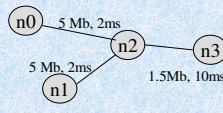
3. Criar a topologia da rede

- Nós (routers)

```
set n0 [$ns Node]
set n1 [$ns Node]
set n2 [$ns Node]
set n3 [$ns Node]
```
- Ligações

```
$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
```

```
$ns <link-type> <node1> <node2> <bandwidth> <delay> <queue_type>
<link-type>: simplex-link, duplex-link, duplex-intserv-link
<queue_type>: DropTail, RED, CBQ, FQ, SFQ, DRR
```



GCOM-DI-UM
27
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

3. Criar a topologia da rede (cont.)

- Rede local LAN, Ethernet (CSMA/CD), com dois nós

```
$ns make-lan "$n1 $n2" $bw $delay LL Queue/DropTail Mac/Csma/cd
```

```
$ns make-lan <node-list> <bandwidth> <delay> <ll_type> \
    <if_queue_type> <mac_type> <channel_type>
<node-list>: lista dos nós que fazem parte da LAN
<ll_type>: LL (Link Layer) - implementa o nível lógico
<if_queue_type>: DropTail - disciplina da fila de espera
<mac_type>: MAC (Mac Layer) - implementa técnica de acesso ao meio
<channel_type>: Channel (Physical Layer) - implementa meio partilhado
```

GCOM-DI-UM
28
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

4. Activar o Encaminhamento (unicast e multicast)

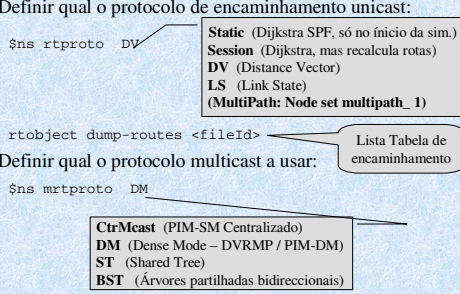
- Definir qual o protocolo de encaminhamento unicast:

```
$ns rtproto DV
```

Static (Dijkstra SPF, só no início da sim.)
Session (Dijkstra, mas recalcula rotas)
DV (Distance Vector)
LS (Link State)
(MultiPath: Node set multipath 1)
- Definir qual o protocolo multicast a usar:

```
$ns mrtproto DM
```

CtrlMeast (PIM-SM Centralizado)
DM (Dense Mode - DVRMP / PIM-DM)
ST (Shared Tree)
BST (Árvores partilhadas bidireccionais)



GCOM-DI-UM
29
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

5. Introduzir erros e falhas nas ligações

- Criar modelo para erros

```
set loss_module [new ErrorModel]
$loss_module set rate 0.01
```


% de erros

```
$loss_module unit pkt
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
```
- Aplicar o modelo para erros

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

Aplica o modelo de erros na *queue*
do link *n0* *n1*

GCOM-DI-UM
30
ST 2004, A.Costa



Uma simulação no NS2

5. Introduzir erros e falhas nas ligações (cont.)

Universidade do Minho

- Simula a falha das ligações
 - afecta o encaminhamento!
 - Os módulos de encaminhamento reagem às falhas e recalculam as suas rotas
- Quatro modelos para falhas:


```
$ns rtmodel Trace <config_file> $n0 $n1
$ns rtmodel Exponential {<params>} $n0 $n1
$ns rtmodel Deterministic {<params>} $n0 $n1
$ns rtmodel-at <time> up/down $n0 $n1
```
- Os parâmetros são:


```
[<start>] <up_interval> <down_interval> [<finish>]
```

GCOM-DI-UM
31
ST 2004, A.Costa



Uma simulação no NS2

6. Criar conexões de transporte (UDP)

Universidade do Minho

- Cria instância UDP no nó origem



```
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
```
- Cria instância no nó destino (para descartar pacotes)


```
set null [new Agent/Null]
$ns attach-agent $n1 $null
```
- Conecta as duas instâncias


```
$ns connect $udp $null
```
- Instala aplicação geradora de tráfego sobre UDP


```
set src [new Application/Traffic/CBR]
$src attach-agent $udp
```

GCOM-DI-UM
32
ST 2004, A.Costa



Uma simulação no NS2

6. Criar conexões de transporte (UDP) (cont.)

Universidade do Minho

- 4 Geradores de Tráfego:
 - CBR (Constant Bit Rate)



```
set src [new Application/Traffic/CBR]
```
 - Exponential on-off

```
set src [new Application/Traffic/Exponential]
```
 - Pareto on-off

```
set src [new Application/Traffic/Pareto]
```
 - Trace (gera tráfego a partir dos dados de um ficheiro)


```
set tfile [new Tracefile]
$tfile filename exemplo-trace.bin
set src [new Application/Traffic/Trace]
$src attach-tracefile $tfile
```

GCOM-DI-UM
33
ST 2004, A.Costa



Uma simulação no NS2

6. Criar conexões de transporte (TCP)

Universidade do Minho


- Instalação de agentes TCP originadores e receptores de tráfego (num só sentido):


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


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


```
set telnet [new Application/Telnet]
$telnet attach-agent $tcp
```

GCOM-DI-UM
34
ST 2004, A.Costa



Uma simulação no NS2


6. Criar conexões de transporte (TCP) (cont.)

Universidade do Minho

- Geração de tráfego sobre TCP a partir de ficheiros
 - Formato binário (**nativo!**)
 - 2 campos de 32 bits por registo:
 - tempo em ms até gerar próximo pacote...
 - tamanho em bytes do próximo pacote...
- Instalação de um agente ligado ao ficheiro de dados:


```
set tfile [new Tracefile]
$tfile filename <file>
set src [new Application/Traffic/Trace]
$src attach-tracefile $tfile
```

GCOM-DI-UM
35
ST 2004, A.Costa



Uma simulação no NS2

7. Transmitir dados entre as aplicações...

Universidade do Minho

- Características:
 - Funcionar sobre os protocolos de transporte
 - Transmitir dados dos utilizadores (HTTP por exemplo)
- Duas soluções distintas:
 - TCP: Application/TcpApp
 - UDP: Agent/Message

GCOM-DI-UM
36
ST 2004, A.Costa

Universidade do Minho

Uma simulação no NS2

7. Transmitir dados entre as aplicações...

- Criar conexão FullTcp (bidireccional!)

```

set tcp1 [new Agent/TCP/FullTcp]
set tcp2 [new Agent/TCP/FullTcp]
$ns attach-agent $n1 $tcp1
$ns attach-agent $n2 $tcp2
$ns connect $tcp1 $tcp2
$tcp2 listen

```
- Transmitir dados (de forma fiável e ordenada)

```

set appl [new Application/TcpApp $tcp1]
set app2 [new Application/TcpApp $tcp2]
$appl connect $app2
# <comando> vai ser executado no destino...
$ns at 1.0 "$appl send <size> \"<comando>\""

```

GCOM-DI-UM

37

ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Dense Mode

GCOM-DI-UM

38

ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Cria o escalonador e define o *trace* desejado

```

# Create scheduler
set ns [new Simulator]

# Turn on multicast
$ns multicast

# Turn on Tracing
set fd [open "mcast.nam" w]
$ns namtrace-all $fd

```

GCOM-DI-UM

39

ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Criar a topologia

```

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

# Create links
$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n0 $n3 1.5Mb 10ms DropTail

```

GCOM-DI-UM

40

ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Activa o encaminhamento multicast e cria 2 grupos

```

# Routing protocol: let's run distance vector
$ns mrtproto DM

# Allocate group addresses
set group1 [Node allocaddr]
set group2 [Node allocaddr]

```

GCOM-DI-UM

41

ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Fonte no nó 1 envia tráfego CBR para o grupo 1...

```

# Transport agent for the traffic source
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
$udp0 set dst_addr_ $group1
$udp0 set dst_port_ 0

# Constant Bit Rate source #0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
# Start at time 1.0 second
$ns at 1.0 "$cbr0 start"

```

GCOM-DI-UM

42

ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Fonte no nó 3 envia tráfego CBR para grupo 2...

```
# Transport agent for the traffic source
set udpl [new Agent/UDP]
$ns attach-agent $n3 $udpl
$udpl set dst_addr_ $group2
$udpl set dst_port_ 0

# Constant Bit Rate source #0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udpl
# Start at time 1.1 second
$ns at 1.1 "$cbr1 start"
```

Universidade do Minho

GCOM-DI-UM
43
ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Receptor no nó 2, junta-se e abandona os grupos dinamicamente:

```
# Can also be Agent/Null
set rcvr [new Agent/LossMonitor]

# Assign it to node $n2
$ns at 1.2 "$n2 join-group $rcvr $group2"
$ns at 1.25 "$n2 leave-group $rcvr $group2"
$ns at 1.3 "$n2 join-group $rcvr $group2"
$ns at 1.35 "$n2 join-group $rcvr $group1"
```

Universidade do Minho

GCOM-DI-UM
44
ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast

- Procedimento habitual para finalizar simulação:

```
$ns at 2.0 "finish"
proc finish {} {
    global ns fd
    close $fd
    $ns flush-trace
    puts "running nam..."
    exec nam out.nam &
    exit 0
}
$ns run
```

Universidade do Minho

GCOM-DI-UM
45
ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast (Melhorias NAM)

- Definição da cor dos pacotes

```
# Colors for packets from two mcast groups
$ns color 10 blue
$ns color 11 red

# Prune packets (predefined)
$ns color 30 purple
# Graft packets
$ns color 31 green
```

Universidade do Minho

GCOM-DI-UM
46
ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast (Melhorias NAM)

- Definir a disposição da topologia na janela:

```
# Manual layout: order of the link is significant!
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n0 $n3 orient right-down

# Show queue on simplex link n0->n1
$ns duplex-link-op $n0 $n1 queuePos 0.5
```

Universidade do Minho

GCOM-DI-UM
47
ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast (Melhorias NAM)

- Definir cores para as fontes...

```
# Group 0
$udp0 set fid_ 10
$n1 color blue
$n1 label "Source for group 0"

# Group 1
$udp1 set fid_ 11
$n3 color red
$n3 label "Source for group 1"
```

Universidade do Minho

GCOM-DI-UM
48
ST 2004, A.Costa

Universidade do Minho

Exemplo Multicast (Melhorias NAM)

- Definir cores para os receptores...

```

$n2 label "Receiver"
$ns at 1.2 "$n2 join-group $rcvr $group2; \
  $n2 add-mark m0 red"
$ns at 1.25 "$n2 leave-group $rcvr $group2; \
  $n2 delete-mark m0"
$ns at 1.3 "$n2 join-group $rcvr $group2; \
  $n2 add-mark m1 red"
$ns at 1.35 "$n2 join-group $rcvr $group1; \
  $n2 add-mark m2 blue"

```

Universidade do Minho

GCOM-DI-UM
49
ST 2004, A.Costa

Universidade do Minho

Estrutura interna do NS

- Escalonador
- Topologia
- Encaminhamento
- Transporte
- Percurso dos Pacotes
- Formato dos Pacotes
- Application

Universidade do Minho

GCOM-DI-UM
50
ST 2004, A.Costa

Universidade do Minho

Escalonador de Eventos

- Três tipos de escalonadores:
 - List: uma lista ligada simples, ordenada, $O(N)$
 - Heap: $O(\log N)$
 - Calendar: baseado em funções de *hash* $O(1)$

Universidade do Minho

GCOM-DI-UM
51
ST 2004, A.Costa

Universidade do Minho

Estrutura dos Nós

Universidade do Minho

GCOM-DI-UM
52
ST 2004, A.Costa

Universidade do Minho

Estrutura das Ligações

Universidade do Minho

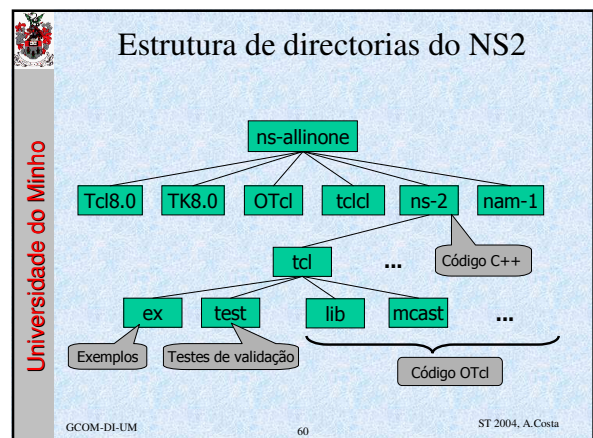
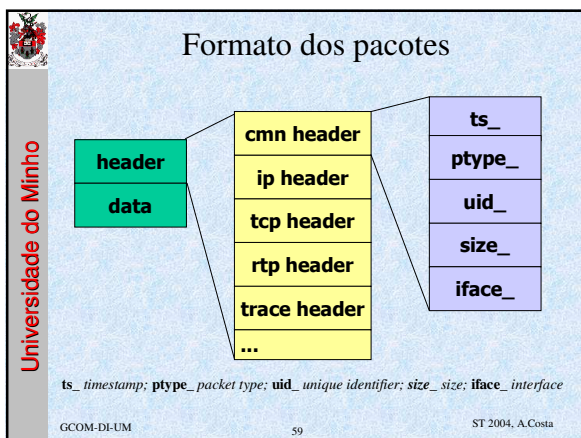
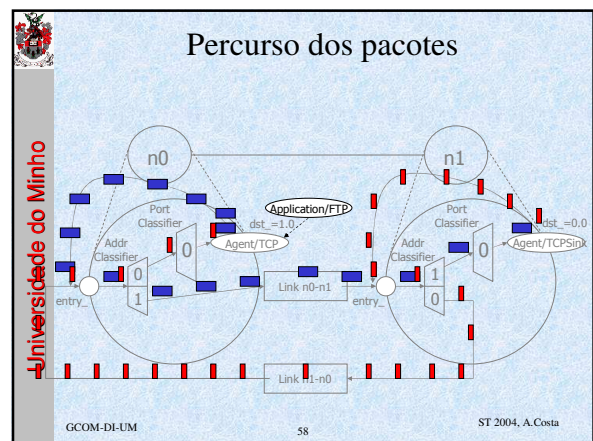
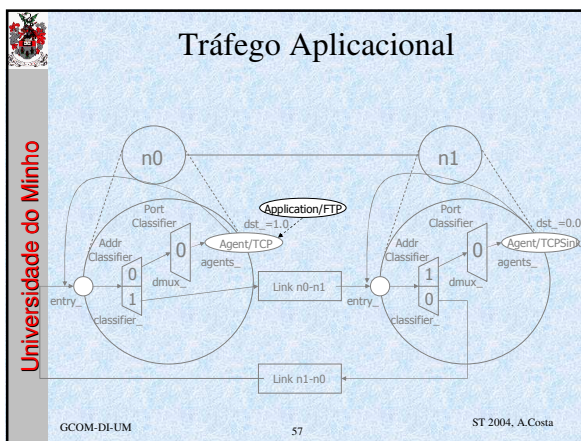
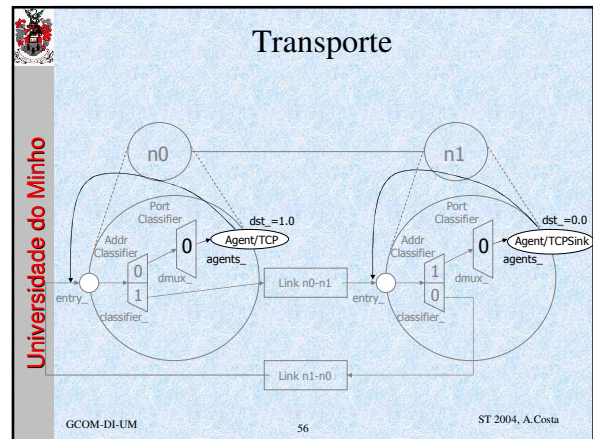
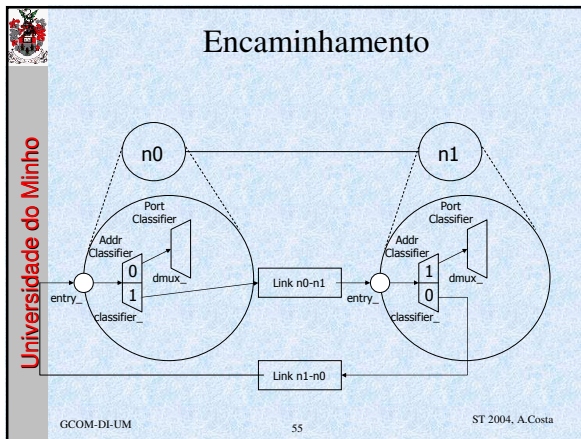
GCOM-DI-UM
53
ST 2004, A.Costa

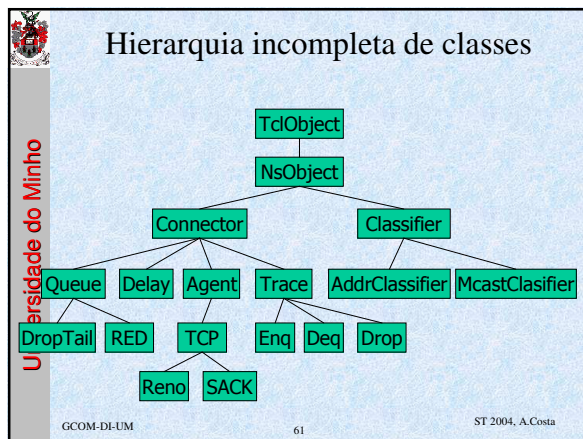
Universidade do Minho

Encaminhamento

Universidade do Minho

GCOM-DI-UM
54
ST 2004, A.Costa





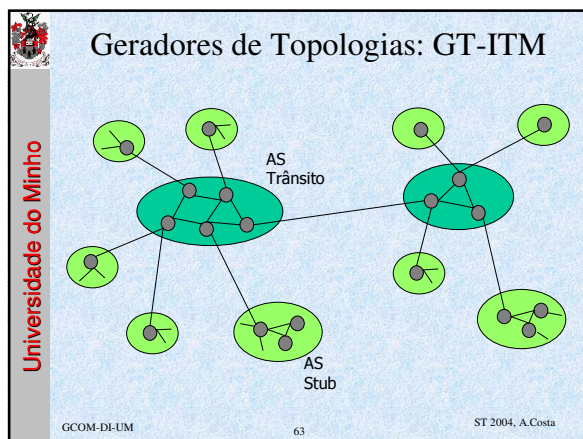
Geradores de Topologias

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

Packages	Graphs	Edge Method
NTG	n-level	probabilistic
RTG	Flat random	Waxman
GT-ITM	Flat random, n-level, Transit-stub	various
TIERS	3-level	spanning tree

Vem com a distribuição do NS2

GCOM-DI-UM 62 ST 2004, A.Costa



Geradores de Topologias: GT-ITM

- Um exemplo simples: gerar 10 grafos com 100 nós cada...

Passo 1: produzir um ficheiro de configuração (r100)

```
# <method keyword> <number of graphs> [<initial seed>]
# <n> <scale> <edgemethod> <alpha> [<beta>] [<gamma>]
geo 10 # 10 topologias "flat random"
100 100 3 .033 # 100 nós, num espaço virtual 100x100
# a probabilidade de link é 0.033
```

Passo 2: gerar as topologias no formato SGB

```
Executar "itm r100" (produz dez ficheiros r100-[0-9].gb)
```

Passo 3: Converter as topologias para scripts NS

```
Executar "sgb2ns r100-0.gb r100-0.tcl"
Para produzir a script NS...
```

GCOM-DI-UM 64 ST 2004, A.Costa

Geradores de Topologias: GT-ITM

- Uma pequena script ns para visualizar a topologia:

```
source r100-0.tcl
set ns [new Simulator]
$ns namtrace-config [open r100-0.nam w]
create-topology ns node 1.5Mb
$ns at 1.0 "exit 0"
$ns run
```

Gera file para o NAM

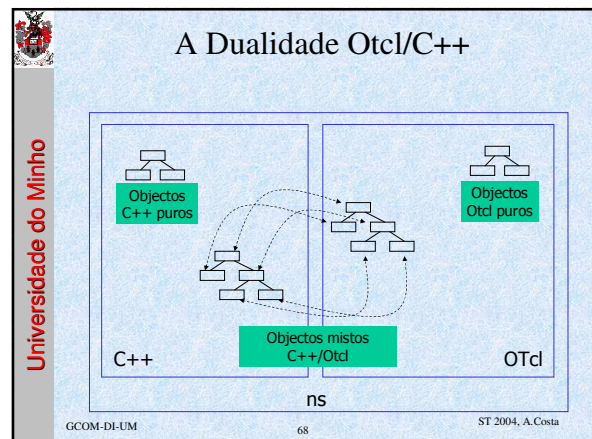
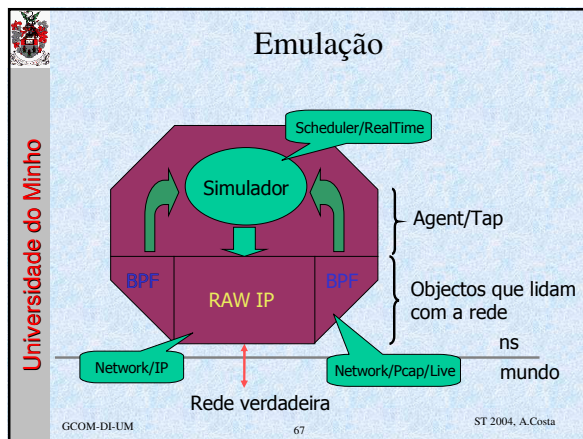
Cria topologia

GCOM-DI-UM 65 ST 2004, A.Costa

Emulação

- Ligar o Simulador ao mundo real:
 - Injectar os pacotes vindos da rede no simulador e enviar os pacotes do simulador para a rede...
 - Actualmente só funciona no FreeBSD...
- Para quê?
 - Sujeitar o simulador a tráfego real e as aplicações verdadeiras às condições impostas pelo simulador!
- O que é necessário:
 - Um escalonador de tempo real (Scheduler/RealTime)
 - Sincronizar o tempo de simulação com o tempo real das aplicações
 - Se tempo de simulação não acompanhar a realidade: Erro!

GCOM-DI-UM 66 ST 2004, A.Costa

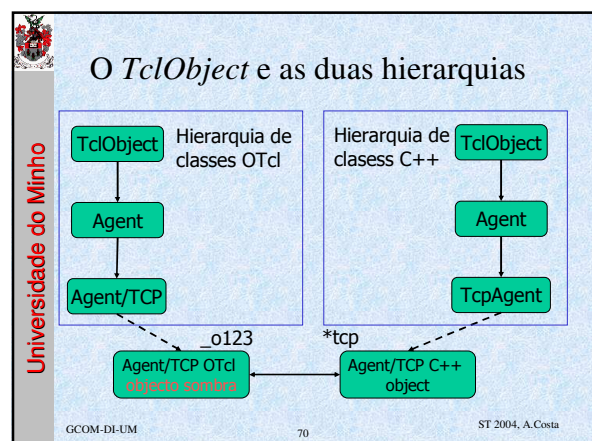


Como ligar o C++ e o Otcl?

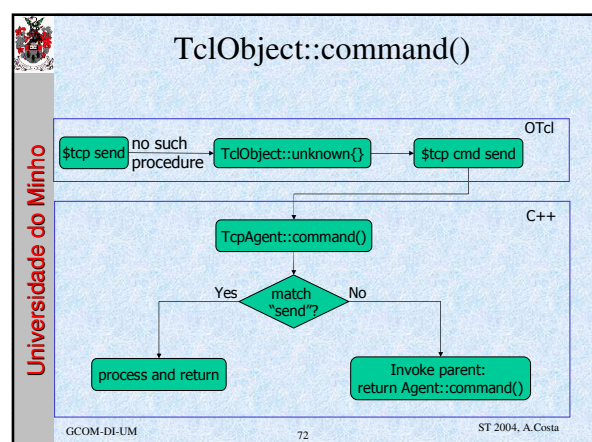
TclObject	Raiz da hierarquia de objectos NS-2 bind(): liga variáveis C++ a variáveis OTcl command(): liga métodos OTcl a implementações C++
TclClass	Cria e inicializa objectos <i>TclObject</i>
Tcl	Métodos C++ de acesso ao interpretador Tcl
TclCommand	Comandos globais
EmbeddedTcl	Inicialização das scripts NS

Universidade do Minho

GCOM-DI-UM 69 ST 2004, A.Costa



- TclObject::command()**
- Permite implementar métodos OTcl em C++
 - Ponto armadilhado: O método OTcl *cmd{ }*
 - Todos os argumentos a seguir a *cmd{ }* são passados como parâmetro ao método *TclObject::command()*
- Universidade do Minho
- GCOM-DI-UM 71 ST 2004, A.Costa



Universidade do Minho

TclObject

- Como construir os objectos?
 - Procedimentos globais: new{ }, delete{ }
- Exemplo:


```
set tcp [new Agent/TCP]
...
delete $tcp
```

GCOM-DI-UM

73

ST 2004, A.Costa

