Modelagem e Avaliação de Desempenho

Pós Graduação em Engenharia Elétrica - PPGEE

Prof. Carlos Marcelo Pedroso

Simulação de Sistemas

- Simulação é a técnica de solução de um problema pela análise de um modelo que descreve o comportamento do sistema utilizando um computador digital.
- Metodologia:
  - Construção de um modelo da situação e reproduzir computacionalmente.
  - Inclusão de alterações para o estudo de otimizações desejadas.

Simulação de Sistemas

- O método de Monte Carlo:
  - Deveu-se a revisão de uma técnica matemática utilizada por cientistas do projeto Manhattan, em Los Alamos, década de 1940, publicada em 1949.
  - Na aplicação desta técnica, os dados são gerados empregando-se um gerador de número aleatórios e uma distribuição de probabilidade que descreve a variável aleatória de interesse.

Simulação de Sistemas

- O método de Monte Carlo
  1. Definir o domínio de entradas possíveis.
  2. Gerar as entradas de acordo com uma distribuição de probabilidade que descreve a entrada.
  3. Realizar o processamento determinístico das entrada.
  4. Agregar os resultados e retornar ao passo 2.

Geração de Variáveis Aleatórias

- Método da inversa
  - Toma-se a distribuição acumulada da variável aleatória, da por $P(X \leq x)=F(x)$.
  - Atribui-se um valor randômico (Ri) entre 0 e 1 para $F(x)$.
  - Calcula-se o valor de x.
  - Desta forma, para cada valor randômico entre 0 e 1 será obtido um valor de x_i.
- Exemplos: distribuição exponencial, distribuição empírica (desenvolvidos em sala).

Geração de Variáveis Aleatórias

- Exercícios:
  - Calcule a expressão para obter uma variável aleatória que segue a distribuição uniforme.
  - Calcule a expressão para obter uma variável aleatória que segue a distribuição triangular.

Geração de Números Randômicos

- Um dos problemas a serem resolvidos é como gerar números randômicos, uniformemente distribuídos entre 0 e 1.
- Gerador Congruente Linear (“LCG”)
  - Definido pela equação linear $x_{n+1}=(ax_n+b) \bmod m$
  - Produz uma sequência entre {0, 1, …, m-1}
  - Pode-se chamar $LCG(m, a, b, x_0)$
  - x_0 é a semente (valor inicial)
  - Ansi $C \rightarrow LCG(2^{31}, 1103515245, 12345, 12345)$
  - *Minimal Standard* $\rightarrow LCG(2^{31}, 16807, 0, 1)$

Geração de Números Randômicos

- Método Tausworthe

1. **Escolha do Polinômio Recorrente**:
   - Um polinômio primitivo de grau r é escolhido. Por exemplo, $P(x) = x^r + a_1 x^{r-1} + \ldots + a_{r-1} x + 1$, onde os coeficientes a_i são binários (0 ou 1).
   - Esse polinômio define a forma como os bits são gerados. A escolha do polinômio primitivo é importante para garantir um período máximo da sequência gerada (de até $2^r - 1$).
2. **Recorrência Linear**:
   - Os bits são gerados de forma recursiva usando a fórmula:

$$x_n = (x_{n-r} \oplus a_1 x_{n-r+1} \oplus a_2 x_{n-r+2} \oplus \ldots \oplus a_{r-1} x_{n-1}),$$

     onde $\oplus$ representa a soma módulo 2 (ou operação *XOR* bit a bit).
   - A sequência de bits $\{x_n\}$ começa com um conjunto de valores iniciais, chamados de *seed* (semente), e o valor de cada novo bit depende de r bits anteriores.

Método Tausworthe

3. **Agrupamento de Bits**:
    - Os bits gerados pela sequência podem ser agrupados para formar números pseudoaleatórios de um tamanho maior, como números inteiros de 32 bits ou números de ponto flutuante entre 0 e 1.
    - Por exemplo, 32 bits consecutivos podem ser convertidos em um único número de ponto flutuante dividindo o número inteiro resultante por 2^{32}.
4. **Período da Sequência**:
    - O período máximo de uma sequência gerada pelo método de Tausworthe é $2^r - 1$, onde r é o grau do polinômio. Isso significa que, após $2^r - 1$ números gerados, a sequência começa a se repetir.
    - Para atingir o período máximo, é essencial que o polinômio usado seja primitivo.

Método Tausworthe

Exemplo Prático

Vamos considerar um exemplo com $r = 3$ e um polinômio $P(x) = x^3 + x + 1$:

1. **Definir os coeficientes**: Para $r = 3$, temos $a_1 = 0$ e $a_2 = 1$.
2. **Recorrência**: A fórmula para gerar os bits seria:

$$x_n = x_{n-3} \oplus x_{n-1}.$$

3. **Escolher uma semente inicial**: Digamos que começamos com os bits $x_0 = 1$, $x_1 = 0$, e $x_2 = 1$.
4. **Gerar a sequência**:
   - $x_3 = x_0 \oplus x_2 = 1 \oplus 1 = 0$
   - $x_4 = x_1 \oplus x_3 = 0 \oplus 0 = 0$
   - $x_5 = x_2 \oplus x_4 = 1 \oplus 0 = 1$
   - E assim por diante.

A sequência gerada seria: 1, 0, 1, 0, 0, 1, ...

Geração de Variáveis aleatórias

- Algumas distribuções podem não possuir expressão analítica para distribuição acumulada (é o caso da distribuição normal).
- Neste caso, é necessário aplicar outros métodos.
- Um dos métodos é o método “acceptance-rejection”
- Para gerar uma VA X com distribuição F(x):
  - Toma-se uma distribuição G(y), com método analítico conhecido.
  - G deve ser próxima de F, com quociente F/G=c

Geração de Variáveis aleatórias

- Escolha da distribuição proposta: Escolha uma distribuição g(x) que seja simples de amostrar. Esta distribuição deve cobrir a forma de f(x). Além disso, deve existir uma constante c tal que:
  - $f(x) \leq c \cdot g(x) f(x)$, para todos os x no suporte de f(x).
- Amostragem: Gere uma amostra X da distribuição proposta g(x).
- Geração de um valor uniforme: Gere um número aleatório U de uma distribuição uniforme no intervalo [0,1].
- Verificação da condição: Calcule a razão f(X) / c·g(X). Se U for menor ou igual a essa razão, aceite X como uma amostra de f(x). Caso contrário, rejeite X e repita os passos 2 a 4.
- Repetição: Continue o processo até que você tenha o número desejado de amostras de f(x).

Acceptance-Rejection Method

- Vantagens:
  - Flexibilidade para usar diferentes distribuições propostas.
  - Pode ser usado para distribuições complexas.
- Desvantagens:
  - A eficiência do método depende da escolha de g(x) e da constante ccc. Se c for muito grande ou se g(x) não cobrir bem f(x), muitos valores podem ser rejeitados, tornando o método ineficiente.

Acceptance-Rejection Method

- Considere um exemplo prático do método de aceitação-rejeição para gerar amostras de uma distribuição de densidade de probabilidade f(x), que é uma distribuição triangular no intervalo [0,1], com o seguinte formato:
  - $f(x)=2x$ para $0 \leq x \leq 1$.
  - Ou seja, f(x) cresce linearmente de 0 a 1.

Passo a passo do exemplo:

1. **Escolha da distribuição proposta $g(x)$:**
   - Uma escolha comum é usar uma distribuição uniforme $g(x) = 1$ no intervalo $[0, 1]$, já que é fácil de gerar amostras a partir dela.
   - Precisamos de uma constante c tal que $f(x) \leq c \cdot g(x)$ para todo $x \in [0, 1]$. Como o valor máximo de $f(x)$ é 2 (quando $x = 1$), podemos escolher $c = 2$.

Acceptance-Rejection Method

2. **Amostragem da distribuição proposta**:
   - Gere uma amostra X da distribuição uniforme $g(x)$ no intervalo $[0, 1]$. Por exemplo, suponha que $X = 0.7$.
3. **Geração de um valor uniforme U**:
   - Gere um número aleatório U de uma distribuição uniforme no intervalo $[0, 1]$. Suponha que $U = 0.5$.
4. **Verificação da condição de aceitação**:
   - Calcule a razão:

$$\frac{f(X)}{c \cdot g(X)} = \frac{2 \cdot 0.7}{2 \cdot 1} = 0.7.$$

   - Compare U com a razão calculada: se $U \leq 0.7$, então aceitamos $X = 0.7$ como uma amostra de $f(x)$; caso contrário, rejeitamos e voltamos ao passo 2.
   - Neste exemplo, como $U = 0.5 \leq 0.7$, aceitamos $X = 0.7$.

Acceptance-Rejection Method

5. **Repetição**:
    - Repetimos os passos até obtermos o número desejado de amostras.

Resumo do exemplo: Nesse caso, a distribuição uniforme foi usada como a proposta, e a constante $c = 2$ foi escolhida para garantir que $f(x) \leq c \cdot g(x)$. O método de aceitação-rejeição permite que a amostra de X seja aceita com probabilidade proporcional à forma da distribuição desejada $f(x)$, garantindo que, ao final, tenhamos amostras que seguem aproximadamente a distribuição triangular $f(x)$.

Acceptance-Rejection Method

- *Distribuição Normal:*

1. *Gere duas variáveis randômicas com distr. Uniforme U(0,1), R_1 e R_2*
2. *Seja $x=-lnR_1$*
3. *Se $R_2 > e^{-(1/2)(x-1)^2}$, volte ao passo 1*
4. *Gere R_3*
5. *Se $R_3>0.5$, retorne $\mu+\sigma x$, caso contrário retorne $\mu-\sigma x$*

Distribuição Normal

- Aproximação:

  $x_i = F^{-1}(R_i) = [R_i^{0,135} - (1-R_i)^{0,135}]/0,1975$

  - Média 0, desvio padrão 1 [N(0, 1)]

  É possível transformar para qualquer outra média μ e desvio padrão σ, fazendo:

  - $y_i = \mu + \sigma x_i$

Exercício

1- Utilize o Método de Monte Carlo para realizar a simulação de uma fila com um servidor, onde o intervalo entre chegadas segue a distribuição exponencial e o tempo de atendimento também segue a distribuição exponencial. Compare o tempo médio na fila com os resultados obtidos com a teoria de filas, modelo M/M/1.

2- Utilize o Método de Monte Carlo para realizar uma simulação de forma a determinar o valor do número π através de uma simulação.

Análise de resultados

- A análise de resultados de uma simulação deve ser feita de maneira muito cuidadosa
  - Especialmente, não cometa o erro de generalizar resultados específicos
  - Para fazer qualquer tipo de inferência sobre os resultados, é necessário realizar uma análise estatística

Confiança estatística

- Um intervalo de confiança compreende um intervalo numérico que possui uma probabilidade igual a (1-α) de incluir o verdadeiro valor da medida de desempenho sob análise, com um nível de confiança.
  - (1-α) representa o intervalo de confiança.
  - α representa o erro admitido ao se concluir sobre a presença do verdadeiro valor da variável no intervalo calculado.

Confiança estatística

- Suponha que foi simulado o tempo médio na fila em um sistema.
  - Assumindo que a variável aleatória X representa o tempo médio na fila.
  - A simulação foi realizada 5 vezes, tomando-se o cuidado de iniciar a simulação com valores de sementes diferentes

Confiança estatística

- Os resultados obtidos foram:

Rodada	X
1	63,2
2	69,7
3	67,3
4	64,8
5	72

- O semi-intervalo h é calculado por:

$$h = t_{n-1,1-\alpha/2} \frac{\sigma}{\sqrt{n}}$$

  - n é o número de rodadas
  - σ é o desvio padrão
  - t indica os valores críticos para distr. t student

Valores críticos – t student

Valores de t para υ graus de liberdade

υ	0,995	0,99	0,975	0,95	0,90
1	63,66	31,82	12,71	6,31	3,08
2	9,92	6,96	4,30	2,92	1,89
3	5,84	4,54	3,18	2,35	1,64
4	4,60	3,75	2,78	2,13	1,53
5	4,03	3,36	2,57	2,02	1,48
6	3,71	3,14	2,45	1,94	1,44
7	3,50	3,00	2,36	1,90	1,42
8	3,36	2,90	2,31	1,86	1,40
9	3,25	2,82	2,26	1,83	1,38
10	3,17	2,76	2,23	1,81	1,37
11	3,11	2,72	2,20	1,80	1,36
12	3,06	2,68	2,18	1,78	1,36
13	3,01	2,65	2,16	1,77	1,35
14	2,98	2,62	2,14	1,76	1,34
15	2,95	2,60	2,13	1,75	1,34
16	2,92	2,58	2,12	1,75	1,34
17	2,90	2,57	2,11	1,74	1,33
18	2,88	2,55	2,10	1,73	1,33
19	2,86	2,54	2,09	1,73	1,33
20	2,84	2,53	2,09	1,72	1,32
21	2,83	2,52	2,08	1,72	1,32
22	2,82	2,51	2,07	1,72	1,32
23	2,81	2,50	2,07	1,71	1,32
24	2,80	2,49	2,06	1,71	1,32
25	2,79	2,48	2,06	1,71	1,32
26	2,78	2,48	2,06	1,71	1,32
27	2,77	2,47	2,05	1,70	1,31
28	2,76	2,47	2,05	1,70	1,31
29	2,76	2,46	2,04	1,70	1,31
30	2,75	2,46	2,04	1,70	1,31
40	2,70	2,42	2,02	1,68	1,30
60	2,66	2,39	2,00	1,67	1,30
120	2,62	2,36	1,98	1,66	1,29
> 120	2,58	2,33	1,96	1,65	1,28

Confiança estatística

No caso anterior, a média calculada é 67.22 e o desvio padrão σ é igual a 3.84;

- Para 99% de confiança, $\alpha=0,05$ e $t_{4,\ 0.975}=2.78$
- O valor de h calculado é de 4,77
- Com 97.5% de confiança a verdadeira média estará entre 62.44 e 71.99.

Exercícios

- Utilize a simulação de fila realizada anteriormente, para chegadas exponenciais e atendimentos exponenciais.
  - Calcule o semi intervalo h para um nível de confiança de 99%
  - O que fazer para melhorar a resposta? (melhorar a resposta implica em reduzir ao mínimo o valor de h).

Exercícios

- Suponha novamente o sistema com uma fila. No entanto, desta vez, suponha que a chegada é modelada por uma distribuição normal N(5,10) e o atendimento é modelado também por uma distribuição normal N(4, 20).
  - Determine o tempo médio de fila e tempo médio no sistema.
  - Realize a simulação de forma a obter uma boa resposta para para o nível de confiança de 99%.
  - Interprete os resultados.

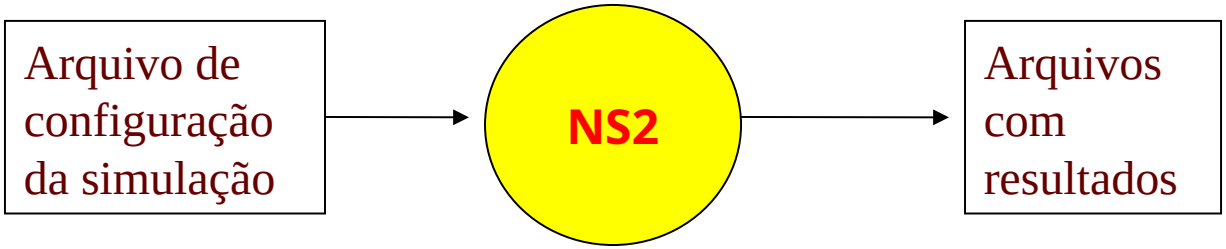
Exercícios

- Suponha novamente o sistema com uma fila. No entanto, desta vez, suponha que a chegada é modelada por uma distribuição exponencial com média 4 e o atendimento é modelado também por uma distribuição de Pareto com parâmetros α=2,5 e β=2. A distribuição de Pareto é uma distribuição de cauda pesada.
  - Determine o tempo médio de fila e tempo médio no sistema.
  - Realize a simulação de forma a obter uma boa resposta para para o nível de confiança de 99%.
  - Interprete os resultados.

Distribuição de Pareto	
Parâmetros	α, β $\alpha > 0$, parâmetro de forma $\beta > 0$, parâmetro de escala
Limites	$b \leq x < +\infty$
Densidade de Probabilidade	$f(x) = \frac{\alpha\beta^{\alpha}}{x^{\alpha+1}}$
Distribuição Acumulada	$F(x) = 1 - (\frac{\beta}{x})^{\alpha}$
Esperança ($E[X]$)	$\frac{\alpha\beta}{\alpha-1}, \alpha > 1$
Variança ($Var[X]$)	$\frac{\alpha\beta}{(\alpha-1)^2(\alpha-2)}, \alpha > 2$

Network Simulator Versão 2

- O NS2 é um simulador escrito em C++ com interpretador OTcl como frontend.



NS2 – Exemplo 1

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

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

#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
```

NS2 – Exemplo1

```
#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

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

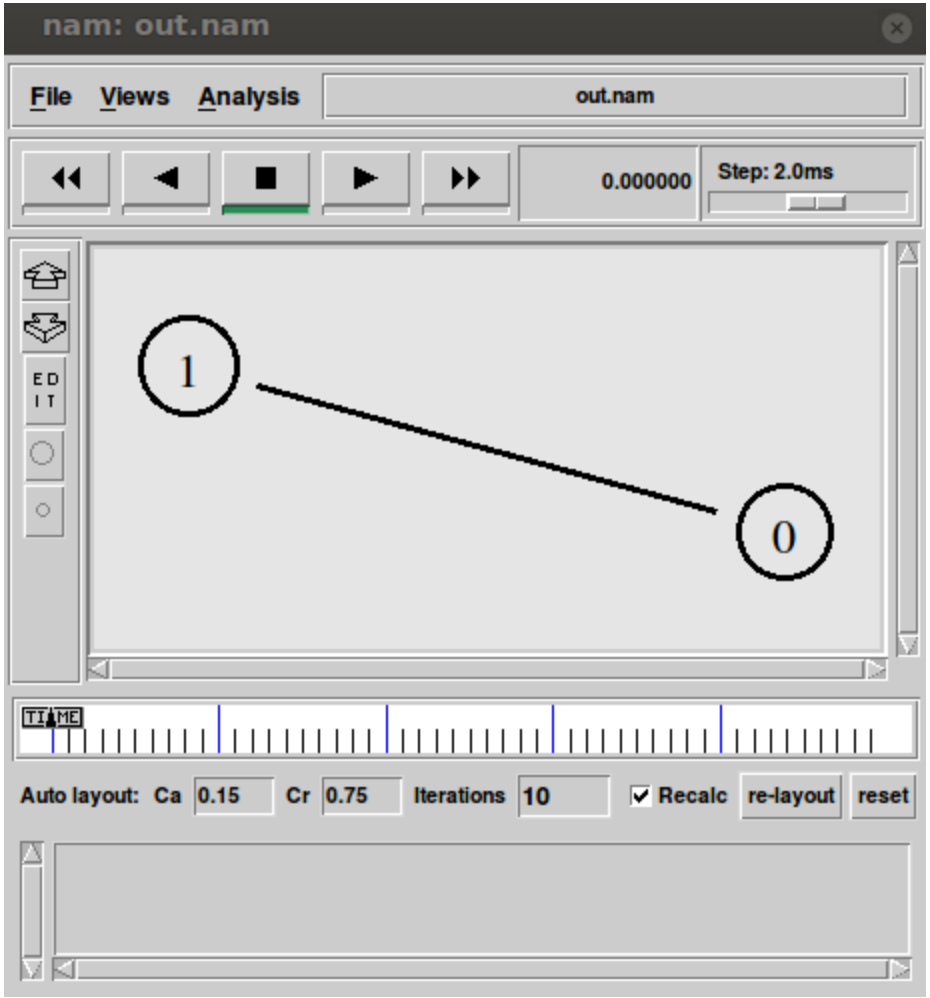
NS2 – Exemplo1

```
#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
```

NS2 – Exemplo1

```
#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
}
```

NAM - Network Animator

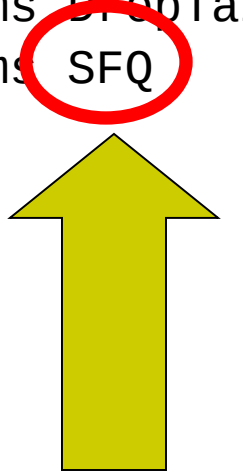


NS2 – Exemplo2

```
#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
```

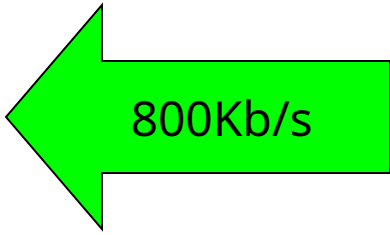
NS2 – Exemplo2

```
#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
```



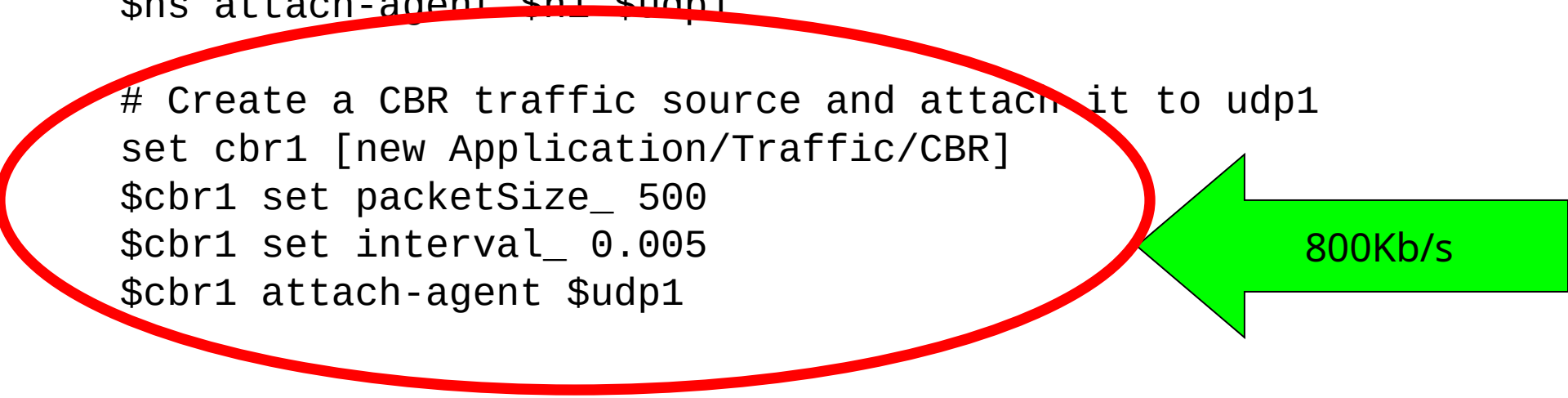
NS2 – Exemplo2

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

#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
```



NS2 – Exemplo2

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

# 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
```



NS2 – Exemplo2

```
#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
```

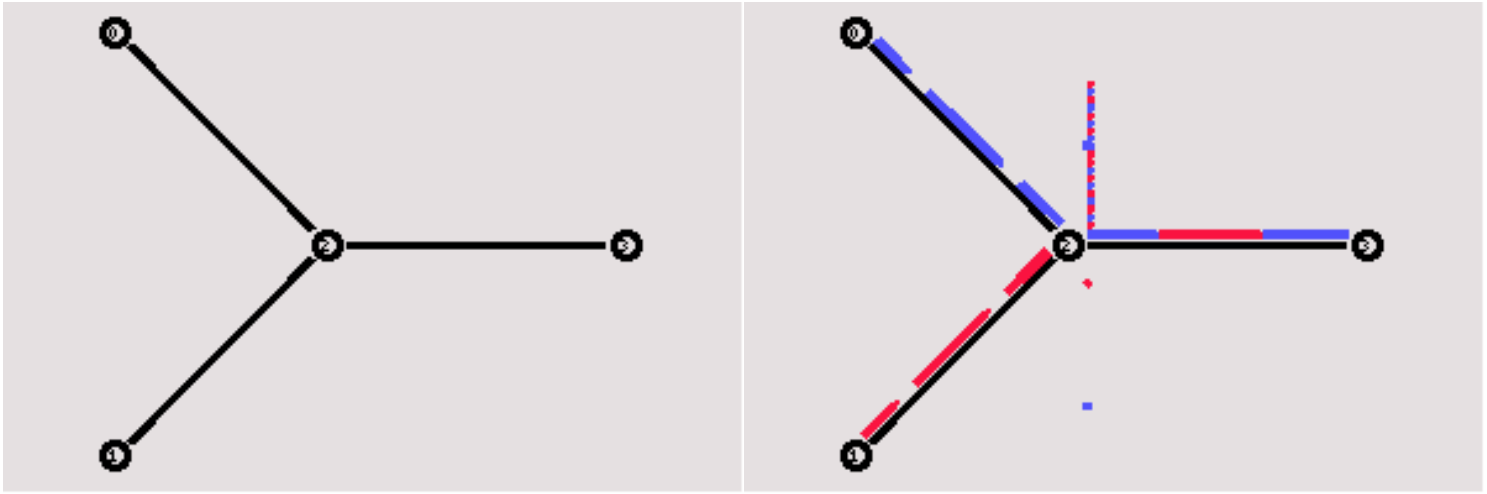
NS2 – Exemplo2

```
#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
```

NS2 – Exemplo2

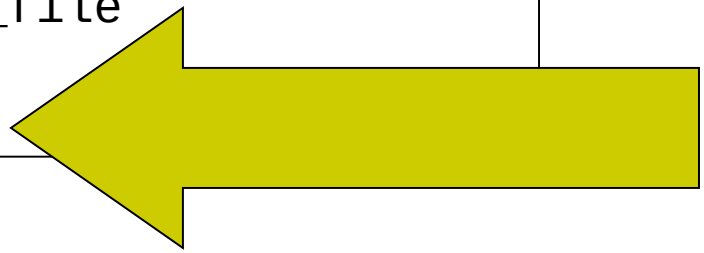
```
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

NS2 – Exemplo2



NS2 – Exemplo4

```
#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
#Initialize a variable
set old_data 0
```

NS2 – Exemplo4

```
#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
```

NS2 – Exemplo4

```
#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
```

NS2 – Exemplo4

```
#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
```

NS2 – Exemplo4

```
#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
```

Acceptance-Rejection Method

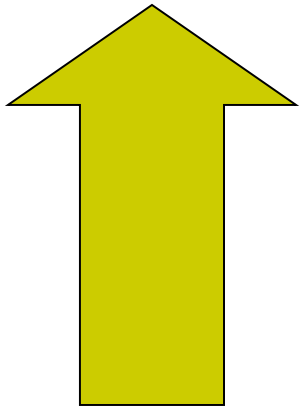
2. **Amostragem da distribuição proposta**:
   - Gere uma amostra X da distribuição uniforme $g(x)$ no intervalo $[0, 1]$. Por exemplo, suponha que $X = 0.7$.
3. **Geração de um valor uniforme U**:
   - Gere um número aleatório U de uma distribuição uniforme no intervalo $[0, 1]$. Suponha que $U = 0.5$.
4. **Verificação da condição de aceitação**:
   - Calcule a razão:

$$\frac{f(X)}{c \cdot g(X)} = \frac{2 \cdot 0.7}{2 \cdot 1} = 0.7.$$

   - Compare U com a razão calculada: se $U \leq 0.7$, então aceitamos $X = 0.7$ como uma amostra de $f(x)$; caso contrário, rejeitamos e voltamos ao passo 2.
   - Neste exemplo, como $U = 0.5 \leq 0.7$, aceitamos $X = 0.7$.

NS2 – Exemplo4

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



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

NS2 -Transmitindo sobre o UDP

- UDP
  - set udp [new Agent/UDP]
  - set null [new Agent/Null]
  - $ns attach-agent $n0 $udp
  - $ns attach-agent $n1 $null
  - $ns connect $udp $null

Geradores de tráfego sobre o UDP

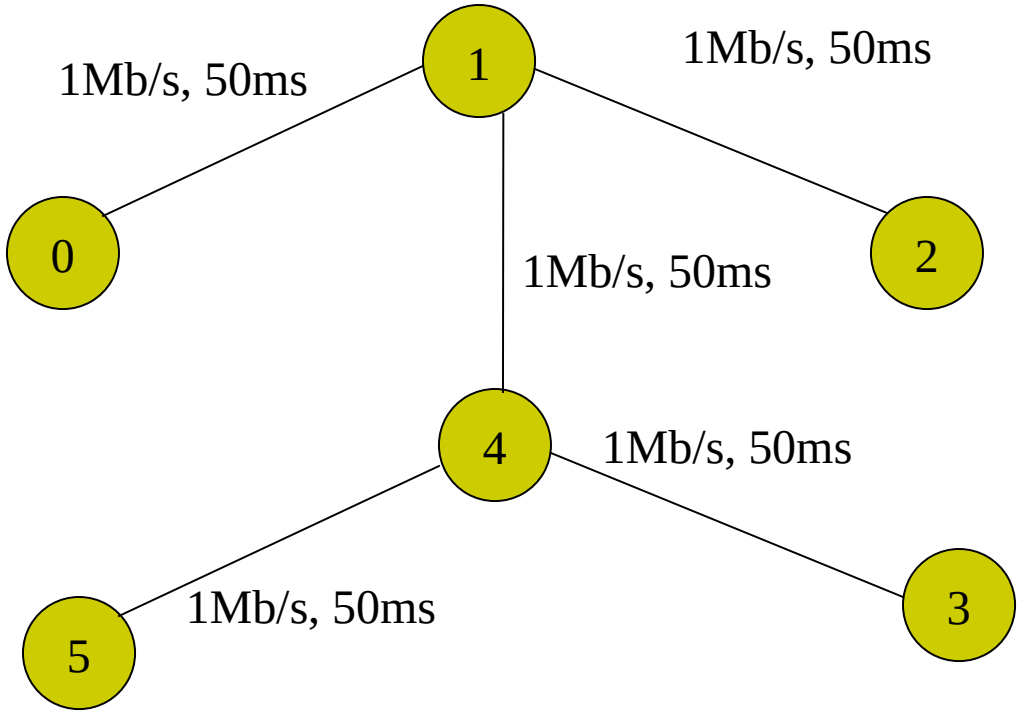
- CBR
  - set src [new Application/Traffic/CBR]
  - $src attach-agent $udp
  - $ns at 3.0 “$src start”
- Exponential
  - set src [new Application/Traffic/Exponential]
- Pareto on/off
  - set src [new Application/Traffic/Pareto]

Criando uma conexão TCP

- 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

Exercícios

- Escreva uma simulação para a topologia abaixo:



Exercício

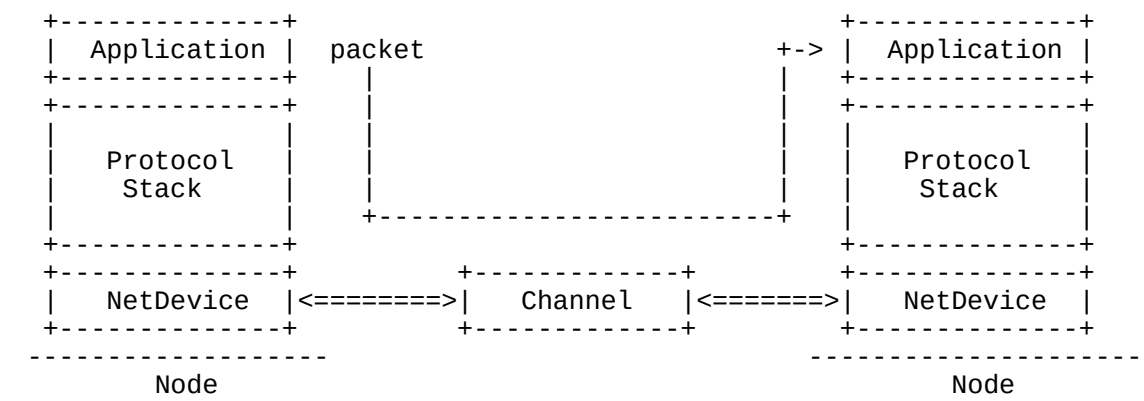
- Adicione aplicativos CBR transmitindo de 0 para 2, de 3 para 2 e de 5 para 2 sobre o protocolo UDP;
- Aumente progressivamente a taxa de geração de tráfego e determine o ponto de saturação da rede. Compare com o máximo teórico;
- Repita a operação utilizando como gerador de tráfego uma aplicação do tipo FTP e verifique como o algoritmo de gerência de janela ativa reduziu a taxa de transmissão. A divisão de banda é justa?
- Troque o algoritmo de descarte para SFQ e verifique se a justiça melhorou
- Adicione um gerador de tráfego UDP anote o efeito sobre os aplicativos TCP

Network Simulator Versão 3

- O NS3 é um simulador totalmente escrito em C++.
- A topologia de simulação também é configurada em um arquivo fonte C++, e compilada para produzir um executável.
- O NS3 possui capacidades de simulação da camada 1,5 até a camada 4 (com algumas poucas aplicações).

NS3

Arquitetura



Agendamento de Eventos: Exemplo

```
Void TcpWebClient::Send (void)
{
  NS_LOG_FUNCTION_NOARGS ();
  NS_ASSERT (m_sendEvent.IsExpired ());

  Ptr<Packet> p;

  p = Create<Packet> (m_data, m_dataSize);
  m_bytesSent += m_dataSize;
  m_txTrace (p);
  m_socket->Send (p);
  ++m_sent;

  NS_LOG_INFO ("Client: sent " << m_size << " bytes to " << (AddressPrinter)m_peerAddress);

  ScheduleTransmit (m_interval);

}
```

```
void
TcpWebClient::ScheduleTransmit (Time dt)
{
  NS_LOG_FUNCTION_NOARGS ();
  m_sendEvent = Simulator::Schedule (dt, &TcpWebClient::Send, this);
}
```

NS3 - Modelos Suportados

Camada de Enlace:

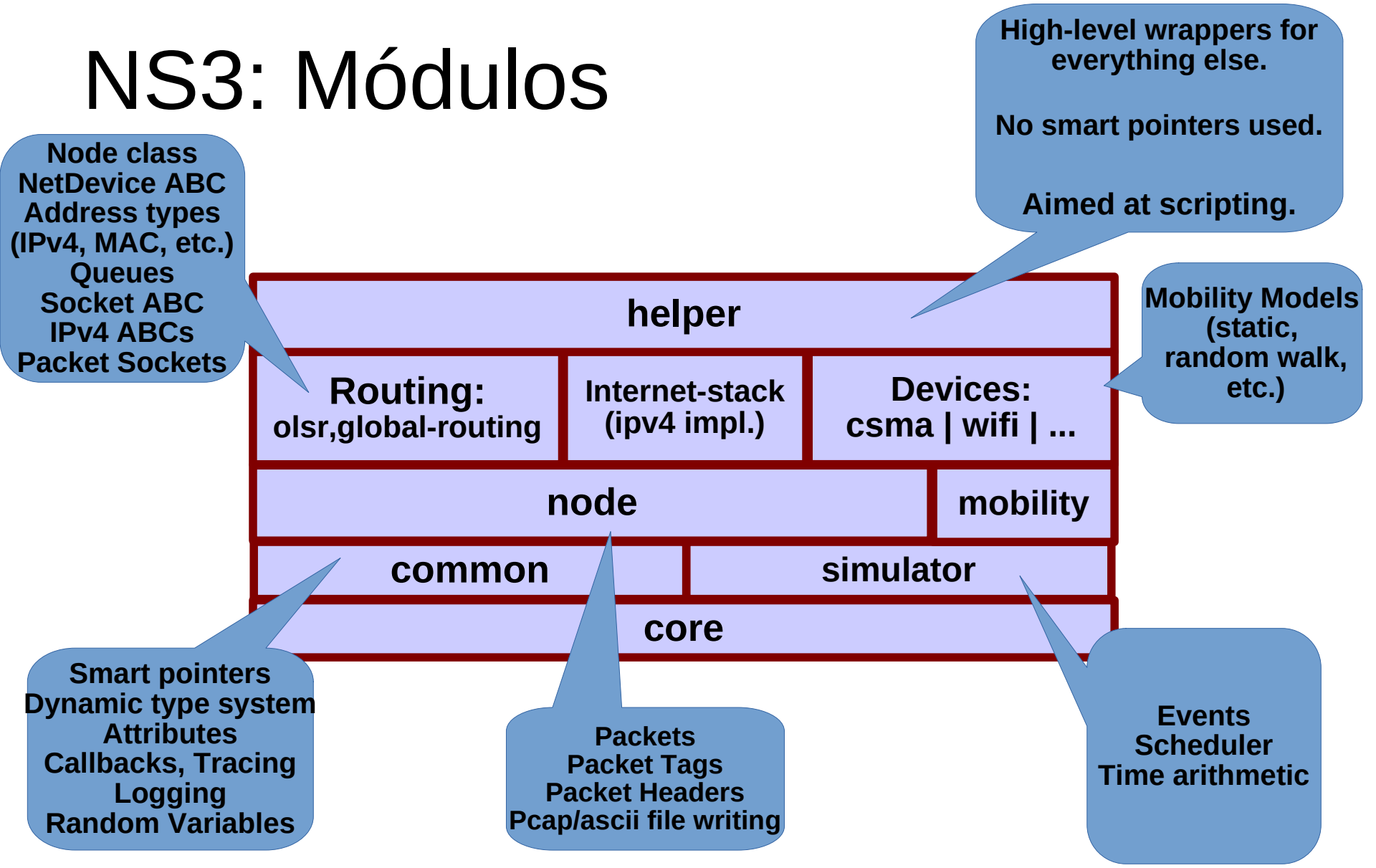
- Point-to-point (PPP links)
- Csma (Ethernet links)
- Bridge: 802.1D Learning Bridge
- Wifi (802.11 links)
    - EDCA QoS support (but not HCCA)
    - Both infrastructure (with beacons), and adhoc modes
- Mesh
    - 802.11s
    - ”Flame”: Forwarding LAyer for MEshing protocol
        - ”Easy Wireless: broadband ad-hoc networking for emergency services”
- LTE, Wimax

NS3 - Modelos Suportados

Aplicações:

- **Onoff**
  - Generates streams, alternating on-and-off periods
  - Highly parameterized
    - Can be configured to generate many types of traffic
      - E.g. OnTime=1 and OffTime=0 means CBR
    - Works with either UDP or TCP
- **Packet sink**: receives packets or TCP connnections
- **Ping6, v4ping**: send ICMP ECHO request
- **Udp-client/server**: sends UDP packet w/ sequence number
- **Udp-echo**: sends UDP packet, no sequence number
- **Radvd**: router advertisement (for IPv6)
- **Socket**: pode ser implementada uma nova aplicação.

NS3: Módulos


High-level wrappers for everything else.
No smart pointers used.
Aimed at scripting.
Node class
NetDevice ABC
Address types (IPv4, MAC, etc.)
Queues
Socket ABC
IPv4 ABCs
Packet Sockets
helper
Routing:
olsr,global-routing
Internet-stack
(ipv4 impl.)
Devices:
csma | wifi | ...
Mobility Models (static, random walk, etc.)
node
mobility
common
simulator
core
Smart pointers
Dynamic type system
Attributes
Callbacks, Tracing
Logging
Random Variables
Packets
Packet Tags
Packet Headers
Pcap/ascii file writing
Events
Scheduler
Time arithmetic

NS3

- O código fonte do simulador pode ser encontrado em http://www.nsnam.org/
- A documentação está disponível em: http://www.nsnam.org/documentation/
- Download - Configure - Compile - Run
- Alguns conceitos de orientação a objeto
- Estudo dos exemplos:
  - First.cc
  - Second.cc
  - Third.cc

NS3: Exemplo Simple Web Server/Client

- Códigos fonte para o modelo:
  - tcp-web-server.cc / tcp-web-server.h
  - tcp-web-client.cc / tcp-web-server.h
- Códigos fonte para o helper:
  - tcp-web-helper.cc / tcp-web-helper.h
  - tcp-web-helper.cc / tcp-web-helper.h
- Código fonte para o exemplo de aplicação:
  - Tcp-webserver-example.cc