Ficha 5 - Network Simulator 2 e TCP/IP

Ano Letivo de 2021/2022

---

1 - Suponha que envia 10 KBytes de informação para 3 máquinas diferentes.

Máquina A: Distância de 600 Km por cabo, usando um canal com uma largura de banda de 200 MHz e com 8 níveis por elemento de sinalização;

Máquina B: Distância de 40 Km por cabo, usando um canal com uma largura de banda de 10 MHz, uma relação S/N de 100 dB;

Máquina C: Ligação por satélite geo-estacionário, situado a uma altitude de 20.000 Km, com um débito máximo de 1 Gbps;

Supondo que não existem erros e admitindo as velocidades máximas de transmissão teóricas, calcule o tempo que o ficheiro demora a chegar às várias máquinas destino.

Notas:

- Despreze os tempos de processamento da informação;
- Indique sempre as unidades que utiliza;
- Apresente os cálculos realizados;

Desenvolva um script TCL preparado para correr no Network Simulator 2 (NS-2) que permita simular o envio da informação descrita.

2 - Pretende-se analisar e comparar a transmissão de dados usando os protocolos UDP e TCP. Para esta análise deverá construir a rede da Fig. 1 para ser simulada no NS2.

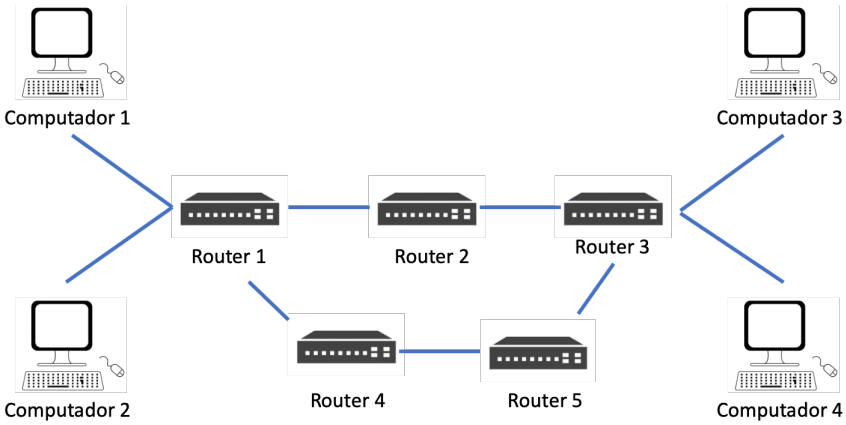

Fig. 1 - Rede

Usando a rede especificada, o "Computador 1" vai enviar ao "Computador 4" por TCP um bloco de dados de 200KB, que começa a ser transmitido no instante 0,5 segundos. Ao mesmo tempo, entre o "Computador 2" e o "Computador 3", existe tráfego que corresponde a uma *stream* de dados que está a ser enviada por UDP (que também começa no instante 0.5 segundos). Dependendo do cenário considerado a *stream* de dados está activa ou desligada.

Características da rede

- Detalhes das ligações:
  - Computador 1 – Router 1: ligação a 80Mb/s
  - Computador 2 – Router 1: ligação a 0.4Gb/s
  - Router 1 - Router 2: Ligação a 200Mb/s
  - Router 2 – Router 3: Ligação a 1Gb/s
  - Router 1 – Router 4: ligação a 300Mb/s
  - Router 4 – Router 5: ligação a 400Mb/s
  - Router 5 – Router 3: ligação a 200Mb/s
  - Router 3 – Computador 3: ligação a 80Mb/s
  - Router 3 – Computador 4: Ligação a 40Mb/s

  - Os tempos de propagação são todos de 10 ms, com excepção da ligação entre o "Router 3" e o "Computador 3" que será de 3ms.
  - Todas as filas são do tipo *DropTail* com o tamanho por *default*. (Ver Nota 1).
  - Todas as ligações são *full-duplex*.
  - Será usado um protocolo de *routing* dinâmico (rtproto DV).

Cenários

- **Cenário 1**:
  - Sem tráfego originado no "Computador 2".
- **Cenário 2**:
  - Computador 2 – Computador 3: 3 Mb/s.

Notas gerais

- Para efeitos de simulação, todo o tráfego do "Computador 1" para o "Computador 4" será criado usando o gerador de tráfego CBR existente no NS2 o qual gerará um pacote de dados com 200KB.
- Use o parâmetro `rate_` do CBR para criar a *stream* de dados iniciadas no "Computador 2". Este parâmetro fará com que o NS2 crie pacotes com o tamanho e cadência pretendidos.
- O tráfego iniciado no "Computador 2" será sempre UDP.
- Use sempre os valores por *default* para o tamanho das filas (excepto a do "Computador 1"), dos pacotes e da janela TCP, excepto quando lhe for pedido explicitamente que os altere.
- Despreze todos os tempos de processamento existentes durante a transmissão dos dados.
- Adeque os tempos de simulação a cada uma das simulações executadas.

Trabalho

1 – Crie a rede de teste descrita.

1.1 - Crie os nós e as ligações entre eles.
1.2 - Identifique cada fluxo de dados com uma cor diferente.
1.3 - Mostre as filas presentes em cada nó.

2 – Para o "**Cenário 1**" quebre a ligação entre o "Router 1" e o "Router 2" no instante 0.6 segundos. Mantenha a ligação quebrada durante 0.1 segundos.

3 - Supondo o "**Cenário 2**", use o TCP com uma janela de transmissão igual a 20. Quebre a ligação entre o "Router 1" e "Router 2" no instante 0.6 segundos. Mantenha-a quebrada durante 0.1 segundos.

Nota 1

Se gerar o pacote de 200KB usando o gerador de tráfego CBR com um *packetSize_* contendo a totalidade do ficheiro e um *maxpkts_* de 1, isso criará uma quantidade de pacotes superior à capacidade da fila. Isso vai provocar a perda de todos os pacotes que não couberam inicialmente na fila da ligação. Para solucionar esse problema pode-se aumentar a fila da ligação para um número superior ao número total de pacotes em que os 200KB vão ser divididos, de modo a que nenhum se perca à partida. Use uma fila com o tamanho adequado.

Ex:

```
$ns queue-limit $n0 $n4 800 ;# 800 is the max number of packets that the queue will hold
```

bwxd.tcl

By *F.Cela & A.Goller* on 02/16/01

Description: Issues regarding Stop-and-Wait protocols.

Stop-and-Wait protocols have some desirable properties:

- Since the sender cannot transmit a new packet until the acknowledgement for the last one sent is received, the protocol is ACK-clocked. These protocols automatically adjust the transmission speed to both the speed of the network and the rate the receiver sends new acknowledgements.
- They respect an underlying conservation principle of computer networks: "Given a producer sending packets to a consumer, if the system is in equilibrium, that equilibrium will be preserved if a new packet is sent only when a previously sent packet is consumed."

However, these protocols become rather inefficient if the path connecting the sender and the receiver has a large Bandwidth x Delay product (that is, the maximum number of bits that can be seen in transit on the link). A possible solution to this problem would be to give the source a credit of packets that can be sent without having to wait for acknowledgements. This would allow the source to fill the pipe, then, the arrival of acknowledgements would maintain and adjust a continuous flow of data. Therefore, a Stop-and-Wait protocol would be a particular case of such protocols, where a credit of one packet is given to the source.These issues are illustrated in the following 4 animations described below:

Animation 1:
Script used: bwxd.tcl
Parameters for this animation: ns bwxd.tcl 10Mb 4ms 1 0.5
Description: These two hosts are connected by means of a 10Mbit/s, 4ms delay link. Host 0 is using a Stop-and-Wait protocol. The line is very ineficiently used.

Animation 2:
script used: bwxd.tcl
Parameters for this animation: ns bwxd.tcl 10Mb 4ms 10 0.5
Description: The two hosts are connected by means of a 10Mbit/s, 4ms delay link. Host 0 is using a window with capacity for 10 segments of 500bytes. The pipe is not being filled with packets and therefore the link is not being efficiently used. The effective bandwidth is, in this case, approximately (10pkt*500bytes*8)/0.008 = 5Mbit/s, half of the available bandwidth.

Animation 3:
script used: bwxd.tcl
Parameters for this animation: ns bwxd.tcl 10Mb 4ms 20 0.5
Description: Bandwidth x Delay = 40000bits = 10 packets of 500 bytes; we have adjusted the window to 20 packets but the pipe is still not full. Why? In our previous calculus we assumed the first ACK was sent exactly when the reception at n1 of the first packet begins. That is not true; the first ACK is sent when the reception of the first packet finishes and the packet is processed (you may need to slow down a bit the animation to see this fact).

Animation 4:
script used: bwxd.tcl
Parameters for this animation: ns bwxd.tcl 10Mb 4ms 21 0.5
Description: We have added one packet more to the size of the window in the previous example; now we are keeping the pipe full.

Ns version: ns-2.1b7

------bwxd.tcl script ------

```
# TCP Sliding Window
#

if {$argc == 4} {
    set bandwidth [lindex $argv 0]
    set delay [lindex $argv 1]
    set window [lindex $argv 2]
    set time [lindex $argv 3]
} else {
    puts "            bandwidth"
    puts "     n0 --------------- n1"
    puts " TCP_window    delay"
    puts "Usage: $argv0 bandwidth delay window simulation_time"
    exit 1
}

# Create the 'Simulator' object
set ns [new Simulator]

# Open a file for writing the nam trace data
set nf [open out.nam w]

$ns namtrace-all $nf

# Add a 'finish' procedure that closes the trace and starts nam
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}

$ns color 1 Red
$ns color 2 Blue

# Define the topology
set n0 [$ns node]
set n1 [$ns node]

#   object      from  to  bandwith   delay    queue
$ns duplex-link $n0   $n1 $bandwidth $delay   DropTail
$ns duplex-link-op $n0   $n1 orient left-right

# Create a traffic source in node n0
set tcp [$ns create-connection TCP/RFC793edu $n0 TCPSink $n1 1]
$tcp set window_ $window
$tcp set packetSize_ 500

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

# When to start and to stop sending

$ns at 0.0 "$ftp start"
$ns at $time "finish"

####### END OF USER CODE ###########################################################
##################################################################################

# Start the simulation
$ns run
```