

Servicios de Red
Práctica 2
Simulación del protocolo TCP en NS

Himar Alonso Díaz

6 de marzo de 2006

Contenido

1. Ejemplo de parada y espera	2
1.1. Descripción del programa. Código fuente	2
1.2. Funcionamiento	4
2. Ejemplo de <i>Slow Start Protocol</i>	6
2.1. Descripción del programa. Código fuente	6
2.2. Funcionamiento	8
3. Ejemplo de cómo TCP ajusta el tamaño de la ventana	10
3.1. Descripción del programa. Código fuente	10
3.2. Funcionamiento	11
4. Ejemplo de ventana deslizante	14
4.1. Descripción del programa. Código fuente	14
4.2. Variación de parámetros	15
4.2.1. Animación 1	16
4.2.2. Animación 2	16
4.2.3. Animación 3	16
5. Cálculo de retardo y RTT	17

Introducción

El objetivo de esta práctica es comprobar el funcionamiento del protocolo de la capa de transporte TCP en el *Network Simulator*. Para ello haremos uso de cuatro ejemplos que describen los siguientes protocolos:

- Protocolo de parada y espera –o en inglés *stop and wait*–.
- Protocolo *Slow start*.
- Protocolo de *multiplicative decrease*
- Protocolo de ventana deslizante.

En cada una de las simulaciones se explicará el funcionamiento del *script TCL* utilizado, para lo cual se mostrará un ejemplo de transmisión de datos a través de una aplicación FTP. También se incluirán algunas muestras gráficas de lo que sucede en la simulación, y algunos diagramas de tiempo explicativos.

En el último apartado veremos cómo calcular el tiempo que se tarda en transmitir los paquetes, y en recibir las confirmaciones –que es a lo que llamaremos RTT–.

1. Ejemplo de parada y espera

El protocolo de parada y espera –o también *stop and wait*– consiste en lo siguiente: después de transmitir un paquete, el emisor espera hasta recibir la respuesta ACK, que es la confirmación que enviará el receptor. Entonces el emisor sabrá que el paquete fue transmitido correctamente.

1.1. Descripción del programa. Código fuente

```
# Creación del simulador:
set ns [new Simulator]

# Creación de nodos:
set n0 [$ns node]
set n1 [$ns node]

# Etiquetas de los nodos:
$ns at 0.0 "$n0 label Transmisor"
$ns at 0.0 "$n1 label Receptor"

# Archivos de salida:
set nf [open stop-n-wait.nam w]
$ns namtrace-all $nf
set f [open stop-n-wait.tr w]
$ns trace-all $f

# Creación de los enlaces:
```

```
$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
```

```
# Límite de la cola de entrada:
```

```
$ns queue-limit $n0 $n1 10
Agent/TCP set nam_tracevar_true
```

```
# Creación del Agente TCP y de los sumideros:
```

```
set tcp [new Agent/TCP]
$tcp set window_1
$tcp set maxcwnd_1
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
```

```
# Configuración de la aplicación FTP
```

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

```
# Creación de monitores:
```

```
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_
```

```
# Visualización de eventos. Nos muestra cuándo se envía una trama, y cuándo se reciben los ACK:
```

```
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
$ns at 3.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Stop and Wait with normal operation\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0\""
$ns at 0.35 "$ns trace-annotate \"Receive Ack_0\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_1\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_1\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_2\""
$ns at 1.23 "$ns trace-annotate \"Receive Ack_2 \\""
$ns at 1.43 "$ns trace-annotate \"Send Packet_3\""
$ns at 1.67 "$ns trace-annotate \"Receive Ack_3\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_4\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_4\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_5\""
$ns at 2.55 "$ns trace-annotate \"Receive Ack_5 \\""
$ns at 2.75 "$ns trace-annotate \"Send Packet_6\""
$ns at 2.99 "$ns trace-annotate \"Receive Ack_6\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""
```

Rutina de salida:

```
proc finish {} {  
    global ns nf  
    $ns flush-trace  
    close $nf  
    puts "Filtrando..."  
    puts "Ejecutando Nam..."  
    exec nam stop-n-wait.nam &  
    exit 0  
}
```

Ejecución de la simulación:

```
$ns run
```

1.2. Funcionamiento

En la Figura 1 se muestra cómo el emisor transmite un paquete, después de haber establecido la conexión con el receptor –mediante un paquete enviado anteriormente–:

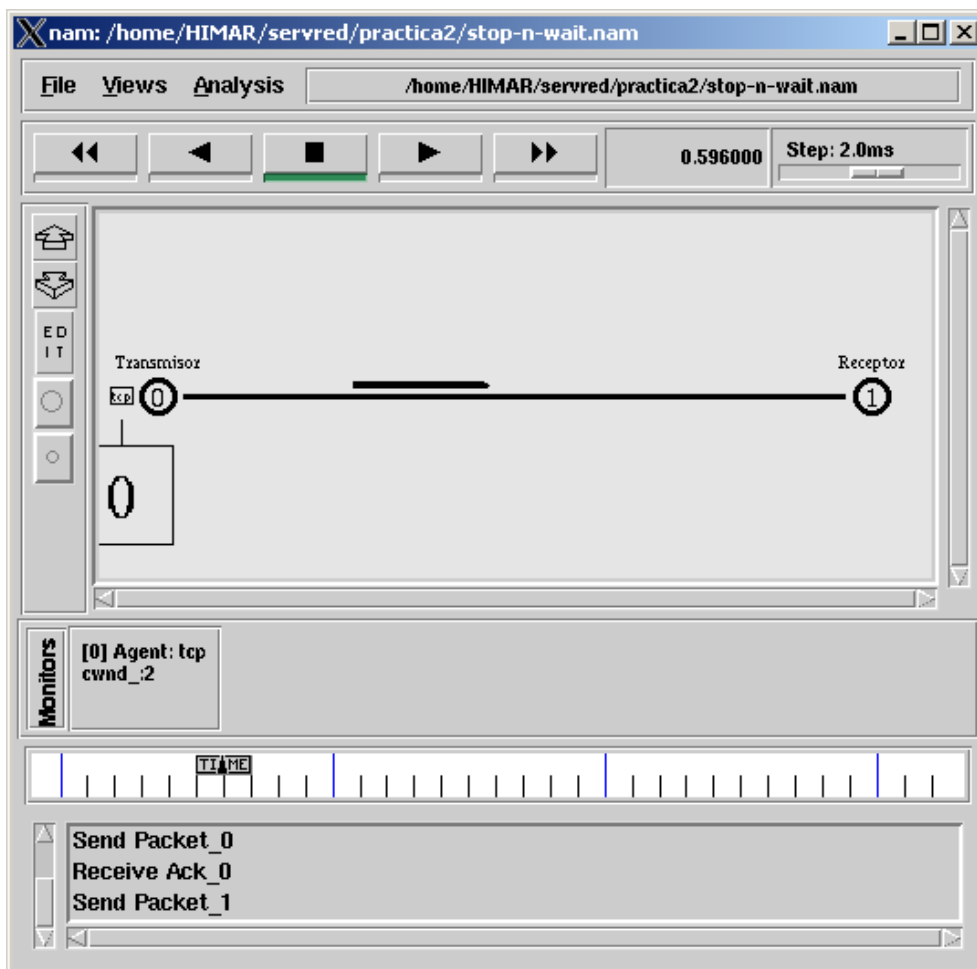


Figura 1: El emisor envía un paquete

En la Figura 2 el receptor confirma la recepción del primer paquete. Hasta que esto no sucede, el emisor no envía el segundo:

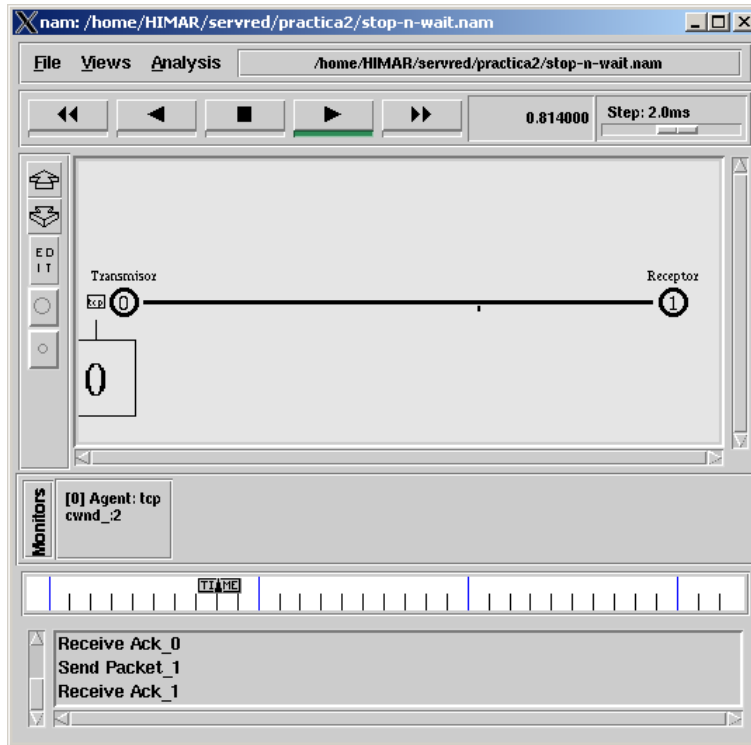


Figura 2: El receptor confirma que ha recibido el paquete

En general, el protocolo de parada y espera sigue un diagrama temporal como el que se muestra en la Figura 3:

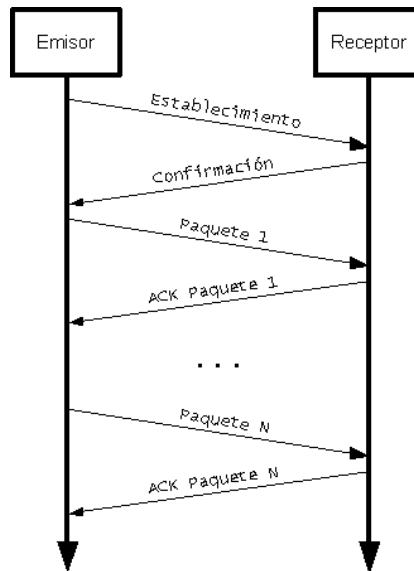


Figura 3: Diagrama temporal del protocolo de parada y espera

2. Ejemplo de *Slow Start Protocol*

El protocolo *Slow Start* es un algoritmo de TCP que permite el control de *congestión*. Funciona de la siguiente manera: En la primera transmisión se envía un paquete. Si no se produce congestión, en la siguiente transmisión se aumenta el tamaño de la ventana (este aumento se realiza de forma exponencial. Cada vez que aumenta el tamaño de la ventana lo hace siguiendo la sucesión $a_n = 2^n$). El aumento sólo se produce mientras no haya congestión. Cuando se detecta, se mantiene fijo el tamaño de la ventana.

2.1. Descripción del programa. Código fuente

```
# Creación del simulador:
set ns [new Simulator]

# Creación de nodos:
set n0 [$ns node]
set n1 [$ns node]

# Etiquetas de los nodos
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"

# Archivos de salida:
set nf [open slow-start.nam w]
$ns namtrace-all $nf
set f [open slow-start.tr w]
$ns trace-all $f

# Creación de enlaces
$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right

# Límite de la cola de entrada
$ns queue-limit $n0 $n1 10
Agent/TCP set nam_tracevar_true

# Creación del Agente TCP y de los sumideros:
set tcp [new Agent/TCP]
$tcp set maxcwnd_8
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink

# Configuración de la aplicación FTP:
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

Creación de monitores:

```
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_
```

Visualización de eventos. Nos muestra cuándo se envía una trama, y cuándo se reciben los ACK, así como el tamaño de la ventana en cada momento:

```
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
$ns at 3.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Slow Start with maximum window size 8 (normal operation)\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0 : Initial window size = 1\""
$ns at 0.34 "$ns trace-annotate \"Receive Ack_0\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_1,2 : Increase window size to 2\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_1,2\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_3,4,5,6 : Increase window size to 4\""
$ns at 1.23 "$ns trace-annotate \"Receive Ack_3,4,5,6 \""
$ns at 1.43 "$ns trace-annotate \"Send Packet_7,8,9,10,11,12,13,14 : Increase window size to 8\""
$ns at 1.67 "$ns trace-annotate \"Receive Ack_7,8,9,10,11,12,13,14\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_15,16,17,18,19,20,21,22 : Keep maximum window size,8\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_15,16,17,18,19,20,21,22\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_23,24,25,26,27,28,29,30 : Keep maximum window size, 8\""
$ns at 2.56 "$ns trace-annotate \"Receive Ack_23,24,25,26,27,28,29,30\""
$ns at 2.78 "$ns trace-annotate \"Send Packet_31,32,33,34,35,36,37,38 : Keep maximum window size, 8\""
$ns at 3.00 "$ns trace-annotate \"Receive Ack_31\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""
```

Rutina de salida:

```
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    puts "filtering..."
    puts "running nam..."
exec nam slow-start.nam &
```

```
    exit 0  
}
```

Ejecución de la simulación:

```
$ns run
```

2.2. Funcionamiento

En las Figuras 4, 5 y 6 puede apreciarse el crecimiento exponencial de la cantidad de paquetes a enviar. En el ejemplo de la simulación, la ventana alcanza un tamaño máximo de ocho paquetes.

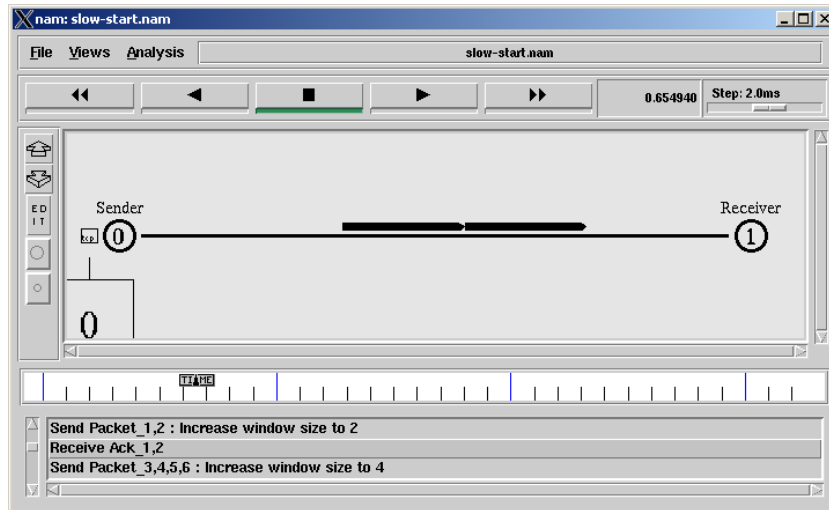


Figura 4: El emisor envía dos paquetes

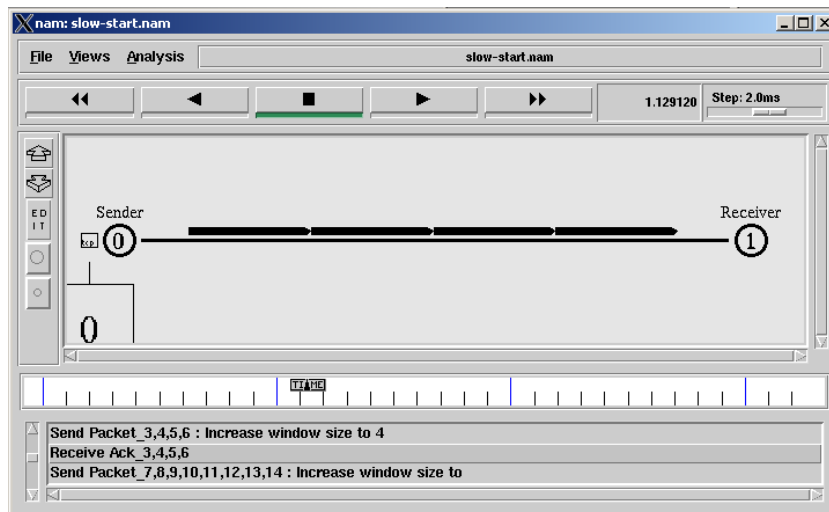


Figura 5: El emisor envía cuatro paquetes

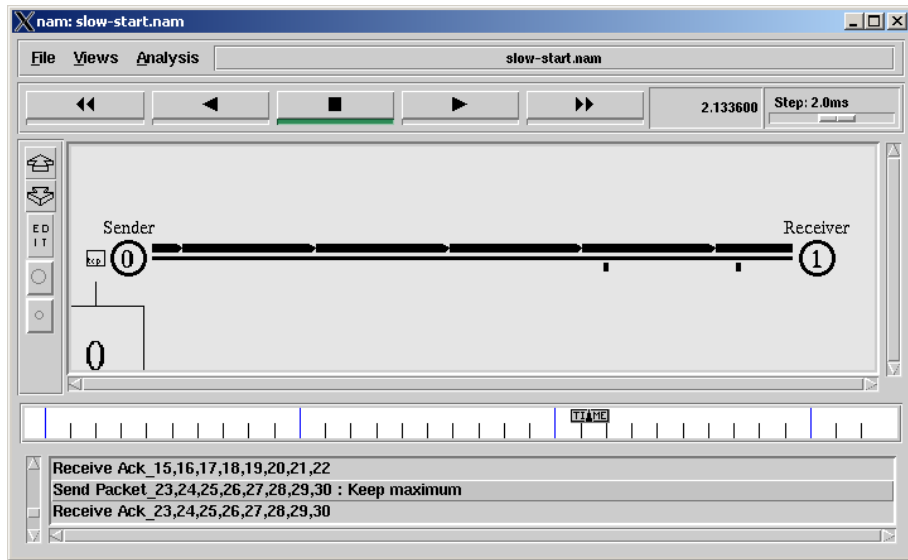


Figura 6: El emisor envía ocho paquetes. El tamaño de la ventana no aumenta más

En el diagrama de la Figura 7 se resume el funcionamiento del protocolo *slow start*. Una vez se detecta la congestión la ventana no sigue aumentando de tamaño:

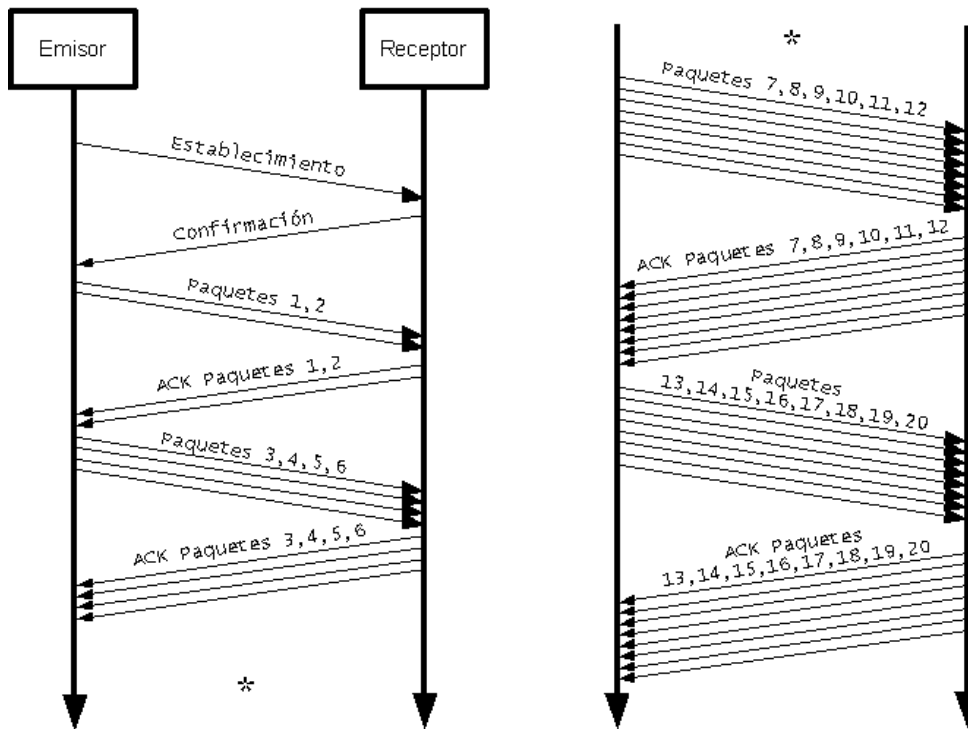


Figura 7: Diagrama temporal del protocolo *slow start*

3. Ejemplo de cómo TCP ajusta el tamaño de la ventana

El protocolo TCP contempla los posibles casos de congestión y pérdida de paquetes durante una transmisión. En estos casos, o también cuando aumentan los retardos o se detectan varios *time-out*, TCP ajusta el tamaño de la ventana, mediante el algoritmo conocido como *Multiplicative Decrease*. En el siguiente ejemplo analizaremos esta situación con *Network Simulator*.

3.1. Descripción del programa. Código fuente

```
# Creación del simulador:
set ns [new Simulator]

# Creación de nodos. Utilizaremos una topología de 4 nodos:
foreach i " 0 1 2 3 "{
    set n$i [$ns node]
}

# Etiquetas de los nodos:
$ns at 0.0 "$n0 label TCP"
$ns at 0.0 "$n3 label TCP"

# Archivos de salida:
set nf [open m-decrease.tr w]
$ns trace-all $nf
set f [open m-decrease.nam w]
$ns namtrace-all $f

# Creación de enlaces:
$ns duplex-link $n0 $n1 5Mb 20ms DropTail
$ns duplex-link $n1 $n2 0.5Mb 100ms DropTail
$ns duplex-link $n2 $n3 5Mb 20ms DropTail
$ns queue-limit $n1 $n2 5
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 queuePos 0.5

# Creación del Agente TCP y de los sumideros:
Agent/TCP set nam_tracevar_true
Agent/TCP set window_20
Agent/TCP set ssthresh_20
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
```

```
$ns connect $tcp $sink
```

```
# Configuración de la aplicación FTP:
```

```
set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
```

```
# Creación de monitores:
```

```
$ns add-agent-trace $tcp tcp
```

```
$ns monitor-agent-trace $tcp
```

```
$tcp tracevar cwnd_
```

```
$tcp tracevar ssthresh_
```

```
# Rutina de salida:
```

```
proc finish {} {
```

```
    global ns nf
```

```
    $ns flush-trace
```

```
    close $nf
```

```
    puts "filtering..."
```

```
    puts "running nam..."
```

```
    exec nam m-decrease.nam &
```

```
    exit 0
```

```
}
```

```
$ns at 0.1 "$ftp start"
```

```
$ns at 5.0 "$ftp stop"
```

```
$ns at 5.1 "finish"
```

```
$ns at 0.0 "$ns trace-annotate \"TCP with multiplicative decrease\""
```

```
# Ejecución de la simulación:
```

```
$ns run
```

3.2. Funcionamiento

En este caso hemos encontrado un problema con el simulador, ya que **NO** utiliza el protocolo de *Multiplicative Decrease* estudiado, sino que comienza a enviar con *slow start*, y cuando detecta pérdida de paquetes ajusta el tamaño de la ventana automáticamente.

Según se observa a lo largo de la ejecución de la simulación, una vez se detecta que se ha perdido algún paquete, el nodo origen vuelve a comenzar de nuevo con *slow start*, de manera que aumenta el tamaño de la ventana de forma exponencial, tal y como se explicó en el apartado anterior.

En la Figura 8 se observa una transmisión sin pérdidas, con un tamaño de ventana de 8. Sin embargo, al aumentar a 16 el tamaño de ventana, hay paquetes que se pierden (Figura 9), de modo que se vuelve a comenzar desde tamaño 1. En la Figura 10 se observa un instante de simulación posterior, donde ya el tamaño de ventana está estabilizado.

En la última figura se muestra un diagrama temporal (Figura 11) donde se explica cómo TCP ajusta el tamaño de la ventana para evitar la pérdida de paquetes.

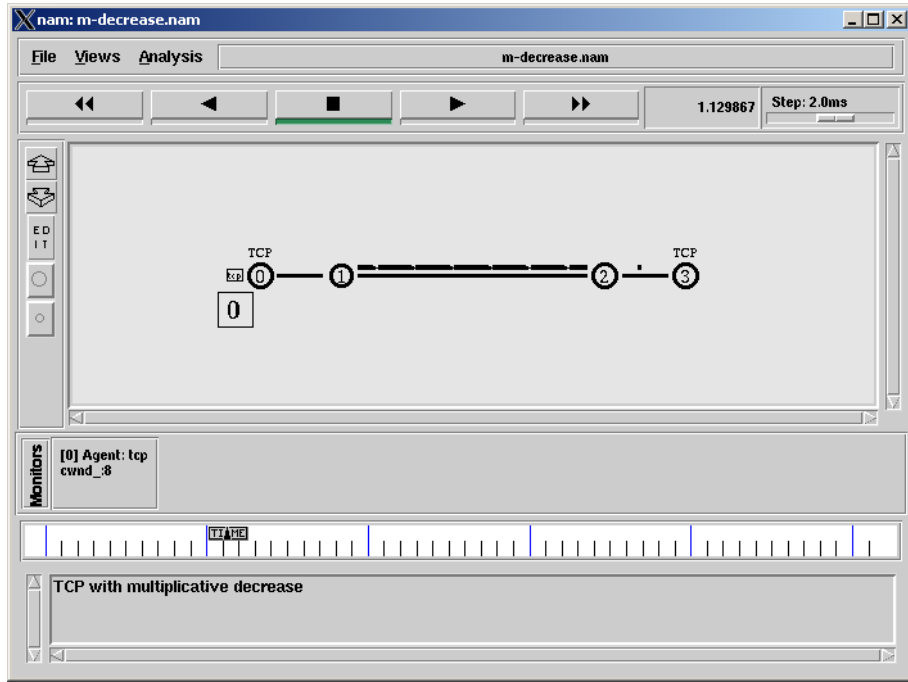


Figura 8: La ventana aumenta de tamaño hasta ocho paquetes sin problema

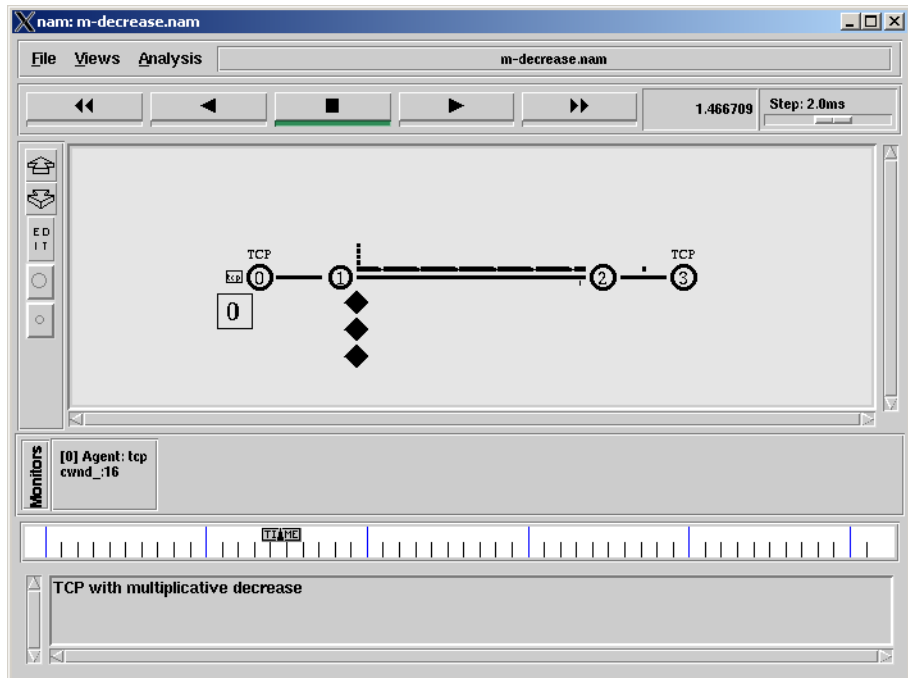


Figura 9: Con un tamaño de ventana de 16 se pierden paquetes

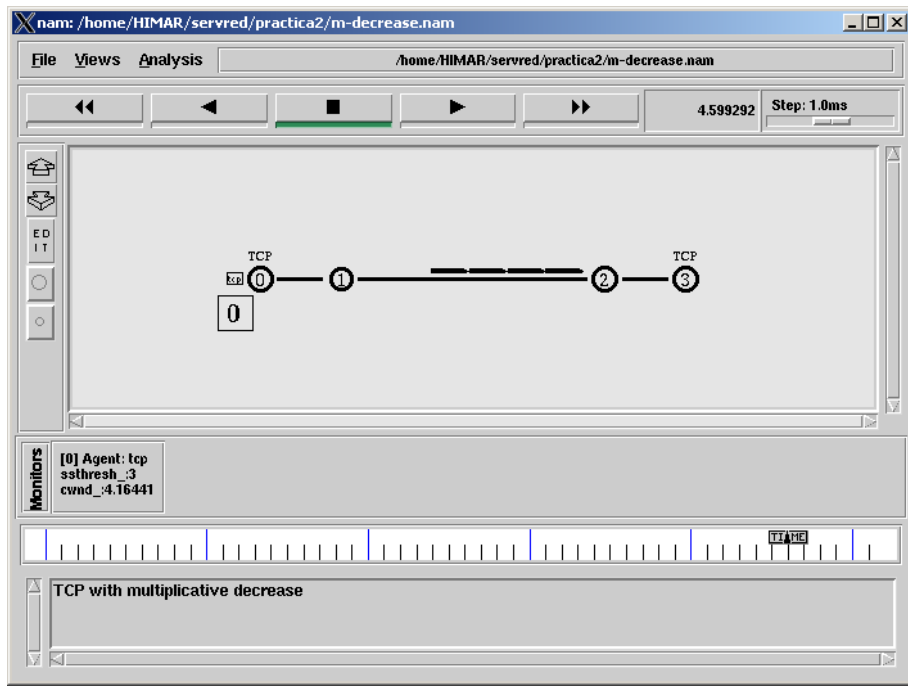


Figura 10: TCP ajusta el tamaño de la ventana para evitar la pérdida de paquetes

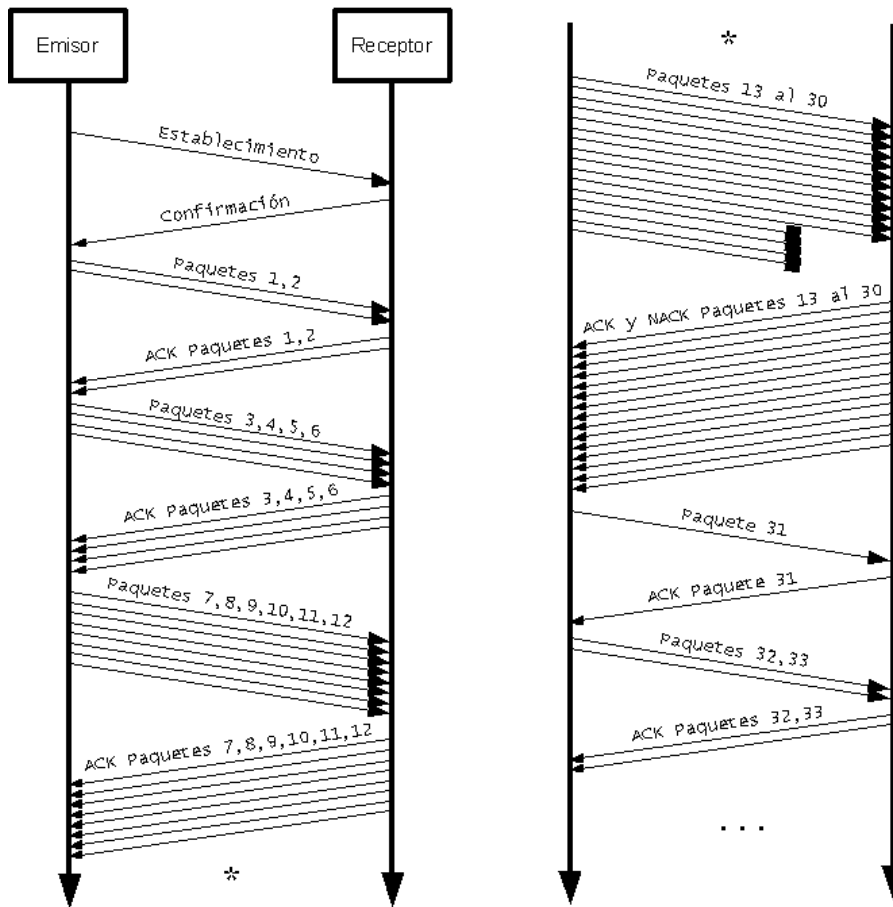


Figura 11: TCP ajusta el tamaño de la ventana para evitar la pérdida de paquetes

4. Ejemplo de ventana deslizante

En el siguiente ejemplo utilizaremos el mecanismo de *ventana deslizante* que emplea TCP para realizar el control de flujo. En lugar de analizar el mecanismo en sí, nos fijaremos en lo que sucede al modificar parámetros tales como:

1. El ancho de banda (*bandwidth*)
2. El retardo (*delay*)
3. El tamaño de la cola en los nodos intermedios (*qsize*)
4. El tamaño de ventana (*tcp-window*)
5. El tiempo de ejecución (*time*)

4.1. Descripción del programa. Código fuente

```
# Definición de variables
set bandwidth 0.2Mb
set delay 40ms
set qsize 10
set tcp_window 20
set time 3.1

# Creación del simulador:
set ns [new Simulator]

# Archivos de salida:
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
$ns trace-all $tf

# Rutina de salida:
proc finish {} {
    global ns trace_nam nf tf
    $ns flush-trace
    close $nf
    close $tf
    exec nam out.nam &
    exit 0
}

# Definición del código de colores:
$ns color 1 Red
$ns color 2 Green
$ns color 3 Blue
```

```

# Creación de nodos:
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Definición de los enlaces entre los nodos
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 $bandwidth $delay DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 queuePos 0.5
$ns queue-limit $n0 $n1 $qsize
set qmon [$ns monitor-queue $n1 $n2 1 2]

# Creación del Agente TCP y de los sumideros
Agent/TCP set nam_tracevar_true
set ttcp0 [new Agent/TCP]
$ns attach-agent $n0 $ttcp0
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $ttcp0 $sink
$ttcp0 set fid_1
$ttcp0 set window_$tcp_window

# Configuración de la aplicación FTP:
set ftp0 [new Application/FTP]
$ftp0 attach-agent $ttcp0

# Creación de monitores:
$ns add-agent-trace $ttcp0 tcp
$ns monitor-agent-trace $ttcp0
$ttcp0 tracevar cwnd_

# Eventos de comienzo y final de la simulación:
$ns at 0.1 "$ftp0 start"
$ns at $time "finish"

# Ejecución de la simulación:
$ns run

```

4.2. Variación de parámetros

A continuación observaremos qué sucede al variar los parámetros que comentábamos anteriormente.

4.2.1. Animación 1

Utilizaremos los siguientes parámetros:

```
# Variables para la Animación 1  
set bandwidth 0.2Mb  
set delay 40ms  
set qsize 10  
set tcp_window 20  
set time 3.1
```

**¿El tamaño de la ventana es 20 desde el establecimiento de la conexión?
¿Qué observa?**

No. Comienza enviando un paquete para realizar el establecimiento de la conexión, y posteriormente va aumentando de uno en uno hasta llegar a 20. En contraste con el *slow start*, que aumentaba de forma exponencial, esta forma de proceder es mucho más lenta.

4.2.2. Animación 2

Utilizaremos los siguientes parámetros:

```
# Variables para la Animación 2  
set bandwidth 155Mb  
set delay 2ms  
set qsize 100  
set tcp_window 20  
set time 1
```

¿Qué ocurre al aumentar el ancho de banda?

Se puede aumentar la tasa de transferencia. Esto supone un aumento considerable en la velocidad, sin dar tiempo a que se llene la cola.

4.2.3. Animación 3

Utilizaremos los siguientes parámetros:

```
# Variables para la Animación 3  
set bandwidth 64Kb  
set delay 100ms  
set qsize 100  
set tcp_window 20  
set time 2
```

¿Qué ocurre ahora cuando el ancho de banda es más pequeño?

Sucede justamente lo contrario. Hay que disminuir la tasa de transferencia, con lo cual quedan almacenados muchos paquetes en la cola, que llega incluso a desbordarse.

5. Cálculo de retardo y RTT

Para realizar el cálculo de los retardos y del RTT (*Round-Trip delay Time* - tiempo desde que se envía la trama hasta que se recibe la confirmación) utilizaremos la información del fichero de salida `out.tr`, donde está almacenada toda la información de la simulación. Para filtrar el contenido del mismo utilizaremos el programa `awk`, que se ejecuta a través de la línea de comandos.

De las columnas que se muestran a la salida, sólo son relevantes para nosotros las cuatro primeras, que nos indican:

1. Evento: envío o recepción

2. Instante de tiempo

3. Nodo origen

4. Nodo destino

- Para visualizar *en qué instantes envía paquetes el nodo 0*, filtramos el contenido del fichero `out.tr` por “tramas enviadas” y “nodo origen = 0”:

```
$ awk '$1=="-" && $3==0 {print $2}' out.tr
```

- Para visualizar *en qué instantes recibe paquetes el nodo 3*, filtramos el contenido del fichero `out.tr` por “tramas recibidas” y “nodo destino = 3”:

```
$ awk '$1=="r" && $4==3 {print $2}' out.tr
```

- Para visualizar *en qué instantes recibe confirmación el nodo 0*, filtramos el contenido del fichero `out.tr` por “tramas recibidos” y “nodo origen = 0”:

```
$ awk '$1=="r" && $4==0 {print $2}' out.tr
```

A continuación se muestra una tabla con los tiempos pertenecientes a los primeros 20 paquetes enviados en la simulación. En la gráfica de la Figura 12 se puede ver la evolución del RTT y el RTTa a medida que se van enviando los paquetes.

Paquete	Enviado	Retardo	Confirmación	RTT	RTTa
0	0,10000	0,16224	0,22448	0,12448	0,89056
1	0,22448	0,34272	0,40496	0,18048	0,80180
2	0,23280	0,38432	0,44656	0,21376	0,72830
3	0,40496	0,5232	0,58544	0,18048	0,65982
4	0,41328	0,5648	0,62704	0,21376	0,60406
5	0,44656	0,6064	0,66864	0,22208	0,55631
6	0,45488	0,648	0,71024	0,25536	0,51869
7	0,58544	0,70368	0,76592	0,18048	0,47642
8	0,59376	0,74528	0,80752	0,21376	0,44359
9	0,62704	0,78688	0,84912	0,22208	0,41590
10	0,63536	0,82848	0,89072	0,25536	0,39583
11	0,66864	0,87008	0,93232	0,26368	0,37931
12	0,67696	0,91168	0,97392	0,29696	0,36902
13	0,71024	0,95328	1,01552	0,30528	0,36105
14	0,71856	0,99488	1,05712	0,33856	0,35824
15	0,76592	1,03648	1,09872	0,33280	0,35506
16	0,77424	1,07808	1,14032	0,36608	0,35644
17	0,80752	1,11968	1,18192	0,37440	0,35868
18	0,81584	1,16128	1,22352	0,40768	0,36481
19	0,84912	1,20288	1,26512	0,41600	0,37121

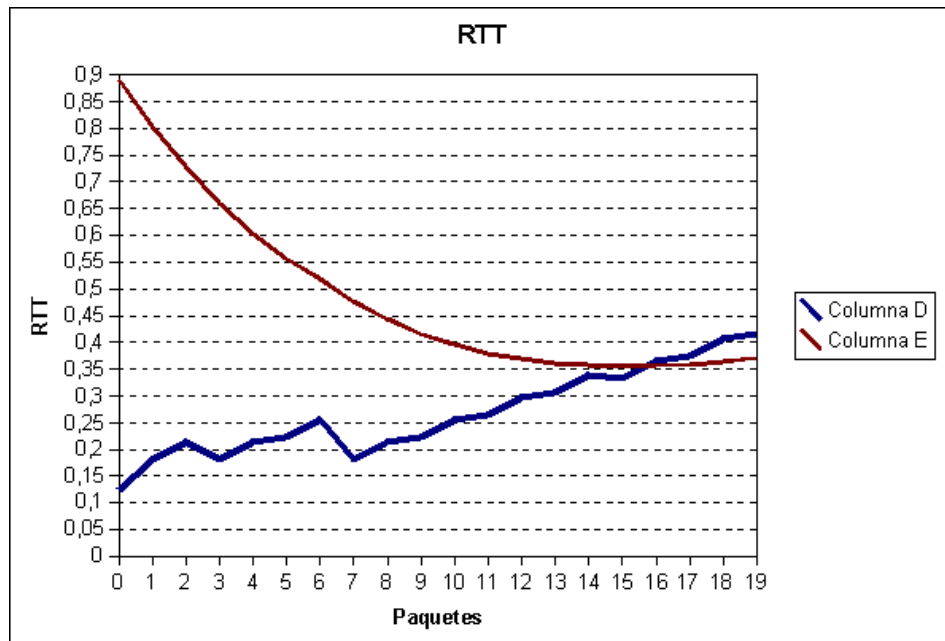


Figura 12: Gráfica de RTT y RTTa