# **Práctica 2** - Omnet++
# Análisis de prestaciones en redes conmutadas (I)

26 de febrero de 2018

## Índice

# 1. Objetivo

En la presente práctica realizaremos una serie de simulaciones con el objetivo de analizar el comportamiento que presenta un switch con respecto a la pérdida de paquetes, dependiendo del tipo de configuración que se establezca para la gestión de sus colas de entrada y salida.

# 2. Preparación del entorno de trabajo

Para preparar el entorno de trabajo realizad los siguientes pasos:

1. Iniciad Omnet++ y comprobad que el proyecto "inet" existe y está incluido en vuestro *Workspace* (si no, se deben seguir los pasos descritos en la práctica 1 para importar el proyecto "inet").

2. Descargad el archivo *Practica-2.tar.gz* desde Poliformat.

3. Extraed dicho archivo en cualquier ubicación de vuestro home. Los ficheros incluídos son: RedMediana.ned y RedMediana.ini.

4. Copiad la carpeta extraída a la raíz del proyecto "inet" ("Copiar" en navegador de archivos y "Paste" en carpeta "inet" del explorador de proyectos).

5. Añadid la nueva carpeta a la lista "NED Source Folders" (clic derecho sobre proyecto *Inet - Properties - Omnet++ - NED source folders*, como se vió en la práctica 1 apartado "Creación de la red").

Nota: Se recomienda anotar los valores numéricos obtenidos en las simulaciones que se preguntan en los ejercicios.

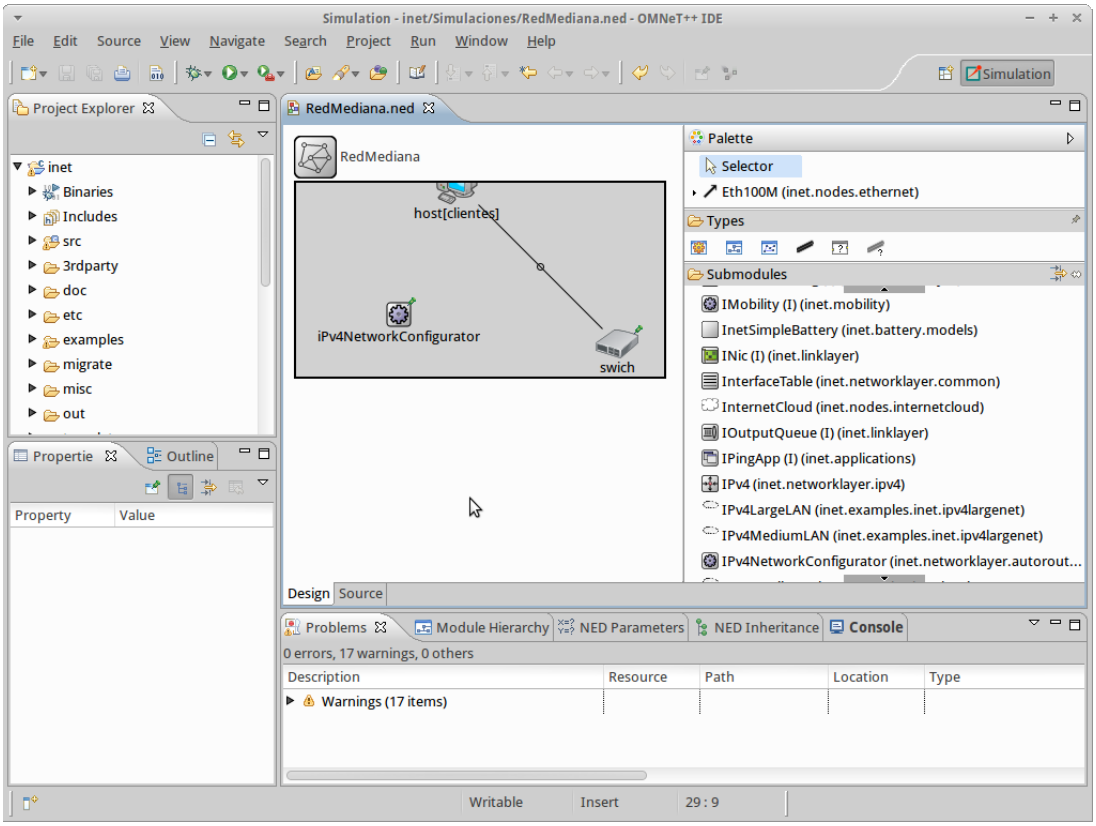


Figura 1: Escenario de la red.

# 3. Descripción de la red

Abre RedMediana.ned. Comprobarás que la red está compuesta por los siguientes componentes:

- Un switch Ethernet (emulado por el módulo EtherSwitch) que interconecta múltiples hosts (módulo StandardHost) mediante una conexión Fast Ethernet (tipo Eth100M).
- El módulo IPv4NetworkConfigurator que proporciona muchas de las funcionalidades para la transmisión de paquetes, como por ejemplo, la asignación de direcciones IP.

El escenario proporcionado se muestra en la figura 1, donde se observa un pequeño circulo sobre la conexión existente entre los hosts y el switch que significa que la conexión se ha definido con múltiples hosts usando un único módulo StandardHost mediante una orden de bucle. El lenguaje NED permite el empleo de un pequeño conjunto de instrucciones de programación (bucles, condicionales, etc.) para facilitar la creación de escenarios.

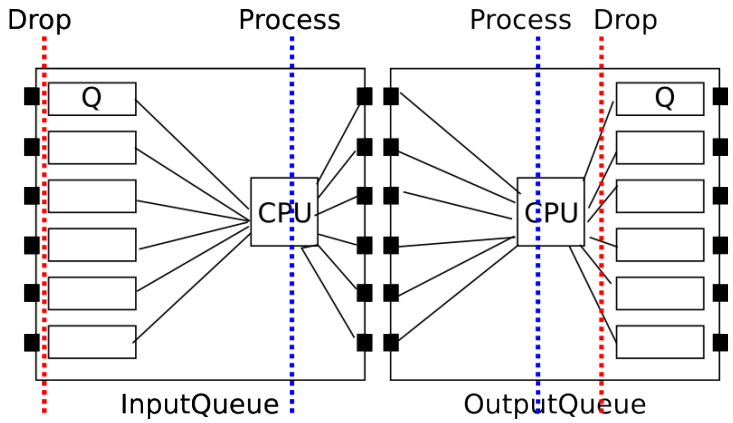


Figura 2: Esquema de las configuraciones. Las lineas roja y azul indican en qué punto del switch se marcan los paquetes como "descartados" y "procesados". Observar que, cuando las colas están a la salida, los paquetes se descartan después de ser procesados.

# 4. Modelos de gestión de colas en switches

Existen dos modelos de gestión de colas en switches, colas a la entrada (*Input Queue*) y colas a la salida (*Output Queue*). En determinadas circunstancias, el tráfico llegado al switch puede superar la capacidad de sus colas, lo que se traduce en una pérdida de paquetes. En la figura 2 se muestra un esquema de ambos modelos, donde la línea azul señala el umbral que superan los paquetes procesados (*processed frames*) y la línea roja marca el punto donde se puede producir la pérdida de paquetes (*dropped frames*), en caso de congestión. Las prestaciones obtenidas con uno y otro modelo variarán dependiendo de diversos factores, entre los cuales estarán el volumen de tráfico generado por los hosts (su número y tasa de transmisión), el destino de los paquetes generados, el tamaño de las colas del switch, el uso, o no, de tramas Pause, etc.

El efecto HOLBlocking afecta a arquitecturas de switch basadas en colas de entrada. Suele presentarse cuando un switch recibe tráfico intenso dirigido a un destino en particular, bloqueando los paquetes dirigidos a otros destinos. En la práctica, se produce un retraso en el reenvío de paquetes hacia los destinos bloqueados, pudiendo provocar que algunas colas de entrada se desborden incluso en situaciones de cargas tráfico no elevadas.

# 5. Configuración de las simulaciones

Se propone llevar a cabo varias simulaciones en las que los host transmiten paquetes UDP al switch con destino a otros host. El objetivo será comparar las prestaciones de las dos implementaciones de gestión de colas: colas de entrada por puerto, y colas de salida por puerto. En ambos casos la memoria total del switch se divide en partes iguales para cada cola. Para ello se probarán diferentes destinos del tráfico generado por los hosts, asi como el uso de tramas Pause, para ver cómo repercute todo ello en la posible pérdida de paquetes:

- Configuración 1: Destino simple, donde el tráfico se dirige a un solo host o a todos los hosts.
- Configuración 2: Destino combinado, donde una mitad del tráfico se dirige a un solo

host y la otra mitad al resto de hosts.

- Configuración 3: Uso de tramas Pause.

Una característica útil de OMNeT++ es poder definir diferentes configuraciones en diferentes secciones de un mismo fichero INI. Entre secciones puede existir una relación de herencia, lo que permite compartir parámetros. Todas las secciones heredan la parametrización de la sección "General".Las secciones se declaran como sigue:

```
[Config Nombre_de_sección]
```

Para heredar la configuración de una sección, hay que añadir la siguiente línea al inicio de la nueva sección:

```
extends = Nombre_de_la_sección_padre
```

Si visualizamos el código del fichero RedMediana.ini suministrado (pestaña "Source"), observaremos que contiene una sección "General" y tres específicas: "SimpleDestination" para la Configuración 1, "CombinedDestination" para la Configuración 2, y "PauseFrames" para la Configuración 3, heredando cada una de ellas los parámetros de la anterior, algunos de los cuales son redefinidos.

En la sección "General" se muestra una configuración básica de las tarjetas de red de los diferentes hosts, así como de algún parámetro del switch. También se observa que se ha reasignado el valor del parámetro "clientes" previamente definido en el fichero NED[1].

**Ejercicio 1**
Examina el código del fichero RedMediana.ini e indica cuántos hosts hay conectados al switch en las configuraciones propuestas, cuál será el tiempo que durará cada simulación, y cuántas aplicaciones UDP se ejecutan en los hosts.

En el switch hay 16 hosts conectados.
Cada aplicación durará 5 segons (sim-time-limit=5s).
En Config SimpleDestination: Se ejecuta una aplicacion por nodo
En Config CombinedDestination: Se ejecutan 2 aplicaciones por nodo.

# 6. Configuración 1: Destino simple

## 6.1. Destino múltiples hosts

Se simulará el comportamiento de la red para la configuración "SimpleDestination" del fichero RedMediana.ini, donde los nodos envían paquetes a todos ellos de manera aleatoria..

[1] La prevalencia en el orden de aplicación de los parámetros se explica en el apartado "Configuración de la simulación, Observaciones" de la práctica 1.

**Ejercicio 2**

Examina el código de la sección "SimpleDestination" e identifica las líneas donde se especifica[a] (situando el ratón sobre un parámetro se muestra el significado de este):

- el destino de los paquetes que generarán los hosts (la linea que comienza por "#" no se ejecutará). en la linea 17, la que empieza por host[*].udpApp[*].destAddress = moduleListByPath("**.host[*]");
- los modelos de colas a la entrada y colas a la salida que se simularán. en switch parameters: input queue y output queue

Mediante el Administrador de configuraciones de simulación[b] (menú principal, "Run" -> "Run Configurations..."), crea una nueva configuración y ejecuta la simulación:

- Crea la nueva configuración con el nombre "SimpleDestination" (botón derecho sobre "OMNeT++ Simulation" -> "New")
- Selecciona "RedMediana.ini", y asigna la sección "SimpleDestination" en "Config name".
- Introduce "*" en "Run number" para ejecutar una simulación (o *Run*) por cada combinación de parámetros especificada en el fichero INI. En nuestro caso, esto nos permitirá simular los dos modelos de gestión de colas des de la misma seccion de configuración. Situa el ratón sobre esta caja de texto para ver el *RunID* que corresponde a cada combinación y anota esta información, ya que será necesaria más tarde para identificar los resultados.
  Run 0: Es de la cua de entrada i la Run 1: Es de la cua de eixida
- Marca la opción "Command Line" para ejecutar la simulación en consola (opcionalmente podemos deshabilitar también logs y debug).
- Pulsa "Apply" para guardar, y después "Run" para lanzar las simulaciones. La salida se puede seguir en la pestaña "Console".
- Una vez finalizada la simulación, genera el fichero ANF de resultados haciendo doble clic en uno cualquiera de los ficheros VEC de esta configuración (carpeta "Resultados"). *Nota: los ficheros VEC pueden tardar un poco a aparecer.*

Consulta los resultados de la simulación en la pestaña "Browse Data" del nuevo fichero ANF. Filtra la columna de parámetros (derecha) para obtener el número de paquetes descartados y procesados en el switch (patrón "*frames"), y rellena los datos de la tabla 1 (si no se producen descartes, todos los paquetes procesados por el switch serán reenviados).

Cuadro 1: Resultados del ejercicio 2.

| Destino | Modelo switch | Run ID | Descartados | Procesados | Reenviados |
|---|---|---|---|---|---|
| a todos | Input Queue | 0 | 0 | 93285 | 93285 |
| | Output Queue | 1 | 0 | 93713 | 93713 |

¿Cuál de los dos modelos de gestión de colas ofrece mejores prestaciones?
Para el estudio realizado donde todos los hosts envian a todos los hosts, cualquiera de los modelos es bueno porque no se han eliminado paquetes ya que no hay opcion de saturacion de colas ni de entrada ni de salida.

Cuando termines cierra la pestaña del fichero ANF.

[a] El significado de todos los parámetros empleados se explica en el anexo 9.

[b] Visto en apartado "El administrador de configuraciones de simulación" de la práctica 1.

## 6.2. Destino un único host

Utilizando la misma sección de configuración "SimpleDestination" mediremos las prestaciones de ambos tipos de switch cuando todos los nodos envían paquetes a un único nodo.

**Ejercicio 3**
Cambia el destino de los mensajes en la sección "SimpleDestination" para que todos los hosts envíen únicamente a "host[0]".
Ejecuta de nuevo la simulación. Consulta los resultados y completa la tabla 2. Con ayuda de la figura 2 deberás calcular también el numero de paquetes reenviados por el switch (entiéndase por reenviados tanto aquellos que ya han salido del switch, como los que esperan en las colas de salida, si las hay).

Cuadro 2: Resultados del ejercicio 3.

| Destino | Modelo switch | Run ID | Descartados | Procesados | Reenviados |
|---|---|---|---|---|---|
| al nodo 1 | Input Queue | 0 | 51547 | 34807 | 34807 |
| | Output Queue | 1 | 52636 | 87758 | 87758-52636 = 35122 |

a) ¿Se ha producido pérdida de paquetes?¿Por qué?
Evidentemente, porque ahora todos los host envian paquetes al host 0 por tanto el switch se satura tanto en colas de entrada como en colas d salida

b) ¿Crees que se ha producido HOL blocking?
Si, en el modelo de colas de entrada

c) ¿Cuál de los dos modelos de gestión de colas ofrece mejores prestaciones?
El modelo de colas de salida porque consigue reenviar mas paquetes, procesa más paquetes y ademas, evita el efecto HOL_Blocking ya que cualquier paquete procesador irá a su cola de salida correspondiente mientras que el efecto HOL_Blocking que se produce en las colas de entrada es porque cuando se satura una cola de entrada y los paquetes quieren entrar van a otra cola y a otra hasta que al final pueden acabar saturandose todas las colas de entrada.

# 7. Configuración 2: Destino combinado

Se simulará la red para la configuración "CombinedDestination" del fichero RedMediana.ini, donde la mitad del tráfico que genera cada host se dirige al host[0], y la otra mitad al resto de hosts. Si examinamos el código de esta sección advertiremos que los hosts ejecutarán ahora 2 aplicaciones UDP. A la nueva aplicación se accede mediante "udpApp[1]".

**Ejercicio 4**

Examina la sección "CombinedDestination" y responde a las siguientes preguntas:

- Identifica las líneas donde se especifica el número de aplicaciones que se ejecutarán y el destino de los paquetes generados por estas. host[*].numUdApps = 2
- ¿Los nodos inyectarán más o menos tráfico en la red que en la Configuración 1?
  El mismo trafico porque ahora el udpApp[*].sendInterval = al doble que en la configuracion 1.

Crea una nueva configuración (siguiendo el procedimiento visto en el ejercicio 2), llámala "CombinedDestination" y ejecútala. Rellena los valores de la tabla 3 a partir de los resultados obtenidos, y contesta las siguientes cuestiones:

Cuadro 3: Resultados del ejercicio 4.

| Destino | Modelo switch | Run ID | Descartados | Procesados | Reenviados |
|---|---|---|---|---|---|
| una aplicacion al nodo 1 y la otra app a todos | Input Queue | 0 | 20356 | 62912 | 62912 |
| | Output Queue | 1 | 11559 | 90658 | 90658-11559 = 79099 |

a) ¿Cuál de los dos modelos de gestión de colas ofrece mejores prestaciones para esta carga de tráfico? El modelo de colas a la salida ya que procesa más y reenvia más y además, en esta simulacion de dos aplicaciones por nodo, descarta menos paquetes.

b) Compara los datos obtenidos con los del ejercicio 3 y contesta:

- ¿Difiere el número de paquetes descartados en ambos modelos? ¿Por qué?
  Si difiere porque ahora hay mas trafico organizado, mas repartido, no todos los nodos envian a uno sino a uno y a todos
- ¿En cuál de los dos modelos se nota más la diferencia? Justifícalo.
  En el modelo de colas de salida actual se nota más la diferencia porque obtiene más paquetes procesados y por tanto, menos paquetes descartados respecto al modelo de colas de salida anterior.
  En el modelo de colas de entrada actual

# 8. Configuración 3: Uso de tramas Pause

En esta simulación mediremos el efecto que tiene el uso de tramas Pause en un switch con colas a la entrada con el propósito de reducir el número de paquetes descartados en las colas.

**Ejercicio 5**
Examina los parámetros de la sección "PauseFrames" y responde:

- ¿Cuál es el parámetro que define las tramas Pause y qué valores toma? ¿Qué ocurrirá con la tasa de inyección de paquetes en la red a medida que aumente dicho valor? relayUnity.pauseUnits = {0,10,300}
  Se reducirá la tasa de inyeccion de paquetes en la red

Crea una nueva configuración y llámala "PauseFrames". Tal como viene especificada la sección "PauseFrames" del fichero INI, se generarán varias combinaciones diferentes de parámetros. Ejecuta solo las combinaciones que correspondan al switch con colas a la entrada (puedes enumerar los runID separándolos por comas en vez de usar "*"). Rellena valores de la tabla 4 a partir de los resultados obtenidos, y contesta las siguientes cuestiones:

Cuadro 4: Resultados del ejercicio 5.

| Destino | Modelo switch | Pause | Run ID | Descartados | Procesados | Reenviados |
|---|---|---|---|---|---|---|
| | | 0 | | | | |
| | | 10 | | | | |
| | | 300 | | | | |

a) ¿Reducen la congestión las tramas Pause?

b) ¿Mitigan el efecto HOLBlocking?

c) ¿Cómo repercute su uso en la prestaciones del switch?

d) ¿Tendría sentido usar tramas Pause en un modelo con colas a la salida?

# 9. Anexo

## 9.1. Configuración de aplicaciones UDP

En la topologia de red planteada se ha considerado la existencia de múltiples host cliente. Para que estos host ejecuten aplicaciones UDP habrá que asignar una aplicación a cada uno de ellos[2].

A continuación se explica el significado de los parámetros relacionados con la configuración de aplicaciones UDP que hay definidos en la sección "ColasEntrada" del fichero "RedMediana.ini".

```
**.host[*].numUdpApps = 1
**.host[*].udpApp[*].typename = "UDPBasicBurst"
```

El parámetro *numUdpApps* define el número de aplicaciones UDP, mientras que el parámetro *typename* define el tipo de éstas. El nombre de la aplicación a ejecutar debe coincidir con alguna de las aplicaciones incluidas en Inet[3]. En nuestro caso utilizaremos la aplicación "UDPBasicBurst", la cual envía una ráfaga de paquetes a un nodo destino cada cierto tiempo. Al estar los nodos organizados en un vector, se puede referenciar un nodo concreto mediante su número, o a todos los nodos del vector empleando el carácter "*".

```
**.host[*].udpApp[0].destAddresses = moduleListByPath("**.host[*]")
**.host[*].udpApp[*].chooseDestAddrMode = "perBurst"
```

Para configurar el destino de los mensajes generados hacemos uso de la función "moduleListByPath()" que lista todos los nodos "host". El parámetro *chooseDestAddrMode* define cuando se selecciona el destino de los mensajes, y puede tomar tres valores: "once", se selecciona aleatoriamente un destino de la lista al inicio de la simulación; "perBurst", se selecciona el destino cada vez que se genera una nueva ráfaga; y "perSend", se selecciona cada vez que se envía un nuevo mensaje. En nuestro caso seleccionamos el valor "perBurst".

```
**.host[*].udpApp[0].destPort = 1000
**.host[*].udpApp[0].localPort = 1000
**.host[*].udpApp[*].messageLength = 1000B
```

Para configurar los puertos de destino, origen y el tamaño de los mensajes, fijamos los parámetros *destPort, localPort* y *messageLength* respectivamente.

[2]Examinando el contenido de la pestaña "Module Hierarchy" podemos ver, entre otras cosas, que el módulo "StandardHost" permite añadir cualquier clase y número de aplicaciones, ya sean TCP, UDP, ICMP o SCTP. Otras vías para la consulta de los parámetros que soporta un módulo son la pestaña "NED Parameters", o durante la edición del fichero INI, como se vió en la práctica 1, apartado "Configuración de la simulación. Observaciones".

[3]Directorio src/applications/udpapp/

```
**.host[*].udpApp[*].startTime = 2s+uniform(0s,0.1s)
**.host[*].udpApp[*].burstDuration = 0.5s+uniform(0s,0.5s)
**.host[*].udpApp[*].sleepDuration = 0.01s
**.host[*].udpApp[*].sendInterval = 0.001s+uniform(-0.0001s,0.0001s)
```

El parámetro *startTime* indica el tiempo de inicio de la aplicación, en nuestra simulación tiene un valor aleatorio uniformemente distribuido entre 2s y 2.1s segundos en todos los nodos. El parámetro *burstDuration* define la duración de las ráfagas, en este caso la duración de cada una de las ráfagas será un valor aleatorio entre 0.5s y 1s. El parámetro *sleepDuration* define el tiempo entre ráfagas, que será siempre de 0.01s. Finalmente el parámetro *sendInterval* define el tiempo entre mensajes, que en este caso será un valor aleatorio entre 0.0009s y 0.0011s.

```
**.relayUnit.pauseUnits = 0
```

Se evita el uso de tramas pause haciendo que el valor de *pauseUnits* (tiempo deespera para el envío de la siguiente trama) sea 0.

## 9.2. Iteraciones en paralelo

El mecanismo "parallel iteration" es útil para evitar la combinación de parámetros incompatibles *(sección 10.4.3 del manual de usuario)*.

Por ejemplo, los parámetros "InputQueuePerPort" y "OutputQueuePerPort" permiten simular un switch con colas a la entrada o con colas a la salida, respectivamente. Ambos se activan o desactivan cambiando el valor de su respectiva variable a *true* o *false*. Dado que el switch solo puede funcionar en uno de los dos modos, las dos opciones no pueden ser ciertas a la vez. Para tener en cuenta esta restricción el switch se configura como sigue:

```
#Run 0
**.switch.**.InputQueuePerPort = ${InputQueue=true,false}
#Run 1
**.switch.**.OutputQueuePerPort = ${OutputQueue=false,true !InputQueue}
```

En la primera línea se declara la variable "InputQueue" sobre la que se iterará en las diferentes "Run". La variable tomará el valor "true" en la primera iteración y "false" en la segunda. En la segunda línea se define la variable "OutputQueue" y se les asignan los valores "false,true" en orden inverso a la anterior. El operador "!" liga el valor de la variable "OutputQueue" con el de la variable "InputQueue", de modo que que las diferentes "Runs" iteran sobre ambas variables en paralelo. El mecanismo se puede aplicar, del mismo modo, a más de dos variables.