IPv6 en INET

Práctica 2 – Diseño de Redes

Instrucciones

Elabora una **memoria**, incluyendo todas las figuras que se piden, así como cualquier figura adicional que consideres importante para explicar los resultados. Las explicaciones deben hacerse de acuerdo a la teoría de la asignatura.

- Cuida la **redacción** y la **ortografía**. Ambas serán tenidas en cuenta.
- El único formato aceptado es **PDF**.
- Utiliza la plantilla **LaTeX Overleaf** oficial del Trabajo Fin de Grado disponible en el Taboleiro FIC.
- Respecto a las **figuras**:
    - **Referencia todas las figuras** en el texto.
    - El contenido de las figuras debe ser legible **sin necesidad de hacer zoom**.
    - Para capturas de la ventana de simulación (Qtenv) o de Wireshark utiliza el **formato PNG**. Para capturas de gráficas de OMNeT++ utiliza el **formato PDF**.

Para la entrega sube a Moodle un único archivo **.zip** con el PDF de la memoria y los ficheros de OMNeT (`*.ned`, `*.ini` y `*.xml`). Utiliza la siguiente estructura dentro del zip:

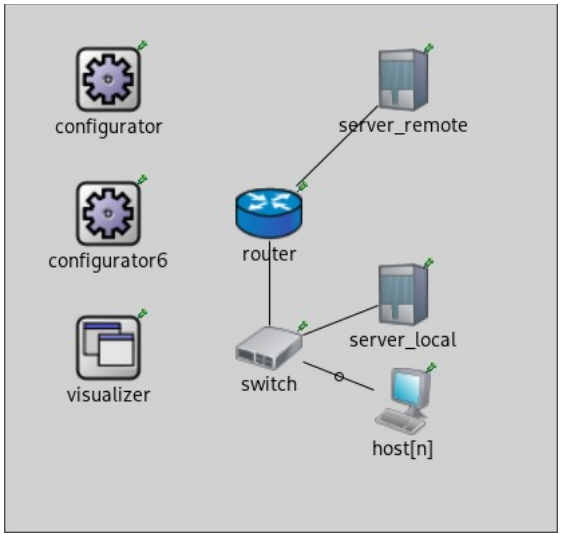
- `memoria.pdf`
- `inet/*.ini`
- `inet/*.ned`
- `inet/*.xml`

Evaluación

- Cada apartado de las preguntas tiene la misma puntuación.
- Las respuestas se calificarán como 0 / 0.5 / 1 punto.
- El total de puntos obtenidos se normalizará sobre 10 puntos.
- La incorrecta legibilidad de las gráficas se penalizará con hasta -1.5 puntos sobre los 10 puntos del total de la práctica.
- Las faltas de ortografía, gramática y sintaxis se penalizarán con hasta -1.5 puntos sobre los 10 puntos del total de la práctica.

Escenario

El escenario *Network.ned*, disponible en Moodle, consiste en un número variable de clientes (vector *host[n]*) conectados junto con un servidor (*sever_local*) en una red local y otro servidor (*server_remote*) situado en otra red.



Crea un nuevo proyecto usando el escenario anterior y un fichero de configuración `omnetpp.ini` basado en la siguiente plantilla (también disponible en Moodle), que configura el módulo *visualizer* para mostrar las direcciones IP y MAC de los equipos de la red, activa la generación de ficheros PCAP para su posterior análisis en Wireshark y define las configuraciones IPv4 e IPv6.

```
[General]
network = Network

*.visualizer.interfaceTableVisualizer.displayInterfaceTables = true
*.visualizer.interfaceTableVisualizer.format = "%N: %m %a"
*.visualizer.interfaceTableVisualizer.nodeFilter = "not(switch*)"
*.visualizer.interfaceTableVisualizer.displayBackground = true
*.visualizer.interfaceTableVisualizer.placementHint = "right"
*.visualizer.interfaceTableVisualizer.displayWiredInterfacesAtConnections = false

*.host[*].numPcapRecorders = 1
**.pcapRecorder[*].pcapFile = "results/" + fullPath() + ".pcap"
**.pcapRecorder[*].alwaysFlush = true
**.crcMode = "computed"
**.fcsMode = "computed"

[Config IPv4]
**.hasIpv6 = false
**.hasIpv4 = true

[Config IPv6]
**.hasIpv6 = true
**.hasIpv4 = false
```

Completa la configuración **[General]** que será común a IPv4 e IPv6:

- Asigna valor a **n** para que haya 3 clientes en la red local.
- Configura tanto en los equipos *server_local* y *sever_remote* una aplicación `TcpGenericServerApp` que acepte conexiones TCP en el puerto 80.
- Añade al equipo *host[0]* una aplicación `TcpBasicClientApp` que se conectará a uno de los servidores a partir de t = 6 s. Configura la longitud de la respuesta (`replyLength`) a 1000 bytes y el intervalo entre conexiones (`idleInterval`) a 4 s.

A continuación completa las configuraciones **[IPv4]** e **[IPv6]** atendiendo a las instrucciones de los siguientes apartados y responde a las cuestiones planteadas.

Configuración IPv4

Añade un servidor DHCP en el equipo *server_local* (`*.server_local.app[1].typeName = "DhcpServer"`, incrementado su `numApps` a 2) y otro al router (`*.router.hasDhcp = true`). Configura en ambos servidores las opciones de DHCP importantes (`numReservedAddresses`, `maxNumClients`, `leaseTime`, etc.). Asegúrate de configurar la opción `gateway` en el servidor DHCP de *server_local* y de que cada servidor asigna direcciones de un rango diferente de la red.

Configura los equipos *host[*]* con clientes DHCP (`*.host[*].app[`**`x`**`].typename = "DhcpClient"`, sustituyendo **x** por el número de aplicación apropiado).

Por defecto el módulo *Ipv4NetworkConfigurator* asignará direcciones IP a todos los equipos, pero los clientes DHCP solo funcionarán si los equipos en los que se ejecutan no tienen ya una IP fija asignada. Para que el *configurator* asigne direcciones IP fijas al router y a los servidores y no a los equipos *host[*]*, crea un fichero `config.xml` con el siguiente contenido:

```
<config>
 <interface hosts='host[*]' names='eth0'/>
 <interface hosts='router' names='eth0' address='192.168.0.1' netmask='255.255.255.0'/>
 <interface hosts='router' names='eth1' address='192.168.1.1'  netmask='255.255.255.0'/>
 <interface hosts='server_local' names='eth0' address='192.168.0.10'  netmask='255.255.255.0'/>
 <interface hosts='server_remote' names='eth0' address='192.168.1.10'  netmask='255.255.255.0'/>
</config>
```

y añade `*.configurator.config = xmldoc("config.xml")` al fichero .ini.

Configuración IPv6

Asigna manualmente direcciones MAC a las interfaces eth0 de *host[*], server_local y server_remote* con `*.`**`equipo`**`.eth[0].address = "`**`mac`**`"`, sustituyendo **equipo** por el nombre del equipo correspondiente y **mac** por la dirección elegida. Para evitar efectos no deseados los últimos 3 bytes de las direcciones deben ser distintos de 0 y los últimos dos bits del primer byte iguales a 0. Elige las direcciones de manera que las de *host[0]* y *host[1]* difieran en los últimos 3 bytes (3 primeros bytes indiferentes).

Cuestiones

IPv4

1.1 Muestra una captura del **tráfico de paquetes** DHCP intercambiados entre el nodo *host[0]* y los servidores DHCP durante el proceso de obtención de su IP, obtenida en **Wireshark** (**Nota:** para que los tiempos mostrados en Wireshark coincidan con los tiempos de simulación, activa *Visualización → Formato de visualización de fecha → Segundos desde 1970-01-01*). Explica lo que ocurre y para qué sirve cada paquete. Para facilitar la captura, configura el `startTime` del cliente DHCP para que se inicie antes en *host[0]* que el resto de equipos.

1.2 ¿Cuál de los servidores proporciona la IP a *host[0]*? ¿Sabe el otro servidor que *host[0]* no cogió la IP ofrecida por él? ¿Cómo? (Muestra el **contenido** de los paquetes relevantes en Wireshark.)

1.3 ¿De qué tipo son los primeros paquetes que se intercambian a partir de t = 6 segundos? ¿Cuál es su objetivo?

1.4 ¿Cuáles son las direcciones origen y destino de estos paquetes (solicitud y respuesta)? ¿Tienen IP origen o destino? ¿Por qué?

1.5 Configura el servidor DHCP en *server_local* con `numReservedAddresses = 10` y el cliente DHCP en *host[2]* para que arranque antes que los otros clientes DHCP. Esto hará que *host[2]* reciba la IP fija asignada a *server_local* (192.168.0.10). ¿Ocurre algún error cuando el *host[2]* recibe la IP de *server_local*? Configura el cliente TCP en *host[0]* para que se conecte a *server_local* y describe paso a paso qué ocurre durante el establecimiento de conexión TCP a partir de t = 6 segundos debido a esta duplicidad.

IPv6

SLAAC

2.1 Muestra una **captura del escenario** en el momento inicial de la simulación en la que se vean todas las direcciones MAC e IP. Utilizando la dirección IP de *host[0]* como ejemplo, explica cómo se construye, destacando los **campos** y **bits** relevantes. Utiliza para explicarlo la notación IPv6 **no abreviada** (16 bytes: xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx).

2.2 Asigna al *host[2]* la misma dirección MAC que al *host[0]* y arranca la simulación. ¿Qué error ocurre antes de que haya transcurrido el primer segundo de simulación? Muestra una **captura del error** que aparece. ¿Qué **paquete** (tipo, origen y destino) provoca el error? ¿Por qué?

2.3 Cambia la MAC del *host[2]* de manera que coincida con la de *host[0]* en los **últimos 3 bytes** y difiera en los **3 primeros bytes (mantén esta MAC para el resto de las cuestiones).** Asigna a *server_remote* la misma MAC que a *host[0]*. ¿Vuelve a ocurrir el error de dirección duplicada con *server_remote* y *host[0]*? ¿Por qué?

2.4 ¿Cuánto tiempo transcurre desde el principio de la simulación hasta que el *host[0]* su IP *link-local* **definitiva** (i.e., fin de DAD)? Muestra la **tabla de interfaces** del nodo *host[0]* en la que se vea su estado antes y después del *DAD timeout* y **explica qué cambia**. (**Nota:** Qtenv muestra toda la información de cada interfaz en una línea; para verla correctamente copia el contenido con botón derecho → *Copy Value* y pégalo en la memoria como texto, en lugar de usar capturas de pantalla.)

2.5 ¿En qué instante de la simulación obtienen los equipos sus direcciones IP **globales**? ¿Cómo obtienen esta última? Muestra la **tabla de interfaces** de nodo *host[0]* en la que se vea su estado antes y después de obtener la dirección global y explica qué cambia.

2.6 Explica cómo se construye la IP global usando el nodo *host[0]* como ejemplo, de nuevo usando la notación IPv6 no abreviada.

2.7. Configura *host[0]* para que se conecte al servidor *server_remote* usando su dirección fe80::x:x:x:x: (asegúrate de que la dirección MAC de *server_remote* es única). ¿Qué ocurre? Repite lo mismo para *server_local*. ¿Qué ocurre?

Vuelve a configurar el cliente TCP en *host[0]* para que se conecte a *server_remote* para las siguientes cuestiones.

Multicast

3.1 En los **primeros 2 segundos** de simulación se envían varios paquetes NS. Muestra una **captura del tráfico de paquetes en Wireshark** en la que se vean las **direcciones IP y MAC origen y destino** de los NS enviados desde cada uno de los nodos *host[*]*. (**Nota**: para mostrar las direcciones MAC añade dos nuevas columnas: ***Hw src addr (unresolved)*** y ***Hw dest addr (unresolved).*** Colócalas a la derecha de las direcciones IP.) ¿De qué tipo son las direcciones IP? ¿Cómo se construyen?

3.2 ¿Coincide alguna de las direcciones IP destino de los diferentes paquetes NS? ¿Por qué? ¿Qué consecuencia tiene esto?

3.3 ¿Por qué se envían los NS a esas direcciones IP?

3.4 ¿Qué **direcciones MAC** destino tienen los paquetes NS anteriores? ¿Cuáles deberían tener según lo visto en clases de teoría? (Utiliza el paquete NS que sale desde *host[1]* como ejemplo y **escribe los 6 bytes** que debería tener la dirección MAC en formato hexadecimal.)

3.5 ¿Qué consecuencia tiene esta diferencia con respecto a lo visto en clase de teoría?

3.6 Muestra las direcciones IP y MAC destino de los mensajes RS y RA que aparecen en torno al segundo 2 de simulación. ¿De qué tipo son las direcciones IP?

3.7 ¿Por qué el RA de respuesta a un RS no usa IP destino unicast?

3.8 ¿Qué direcciones MAC destino deberían tener los RS según lo visto en clases de teoría? ¿Y los RA?

3.9 Aproximadamente en t = 6 s el router envía dos NS. ¿De qué tipo son las direcciones IP destino de estos paquetes? Explica cómo se construyen (notación IPv6 no abreviada).

3.10 ¿En qué equipos llega el **segundo paquete NS** enviado por el router al módulo *ipv6*? ¿Y al submódulo *ipv6.neighborDiscovery*? ¿Por qué? **Nota:** para ver el recorrido del paquete en el módulo *ipv6*, haz doble click en el nodo deseado y luego en el módulo *ipv6*. Puedes mostrar varios nodos a la vez en diferentes ventanas con botón derecho → *Open Graphical View for 'ipv6'* una vez dentro de ese nivel.)

3.11 ¿En qué equipos llegaría el mensaje al módulo *ipv6* si INET implementase direcciones MAC multicast (33-33-xx-xx-xx-xx)? ¿Por qué?

Neighbor Discovery

4.1 ¿Por qué son necesarios los NS que se envían en t = 6 s? Muestra una **captura del log** del nodo que lo envía que muestre el motivo del envío. ¿Qué paquete cumple la misma función en IPv4?

4.2 ¿Por qué no se usa una IP unicast para ese mensaje, si ya es conocida?

4.3 Muestra capturas de la ***neighbor cache*** del nodo que envió el NS un segundo antes de enviarlo, justo después de enviarlo pero antes de recibir el paquete de respuesta y justo después de recibir la respuesta y **explica las diferencias entre los 3 estados.** (**Nota:** para ver la *neighbor cache* haz doble click sobre el nodo → *ipv6* → *neighbourDiscovery*, y a continuación expande *owned objects* en la ventana inferior izquierda. La *neighbor cache* es el atributo *neighbourMap.*)

4.4 ¿Envía el *host[0]* algún otro mensaje NS después del instante t = 6 s? ¿Cuál es su objetivo? Muestra una **captura del mensaje** en Wireshark y una **captura del log** en la que se muestre el motivo del envío.

4.5 ¿Es la IP destino de este NS del mismo tipo que en los NS enviados anteriormente? ¿Por qué?

4.6 Muestra capturas de la ***neighbor cache*** del *host[0]* en los siguientes instantes de tiempo:

- 7 segundos antes del envío del NS de la pregunta anterior.
- 3 segundos antes del envío del NS.
- Justo después del envío del NS y antes de recibir el NA respuesta.
- Justo después de recibir el NA respuesta.

Explica las diferencias observadas entre las capturas.