MANETs en INET

Práctica 3 – 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:

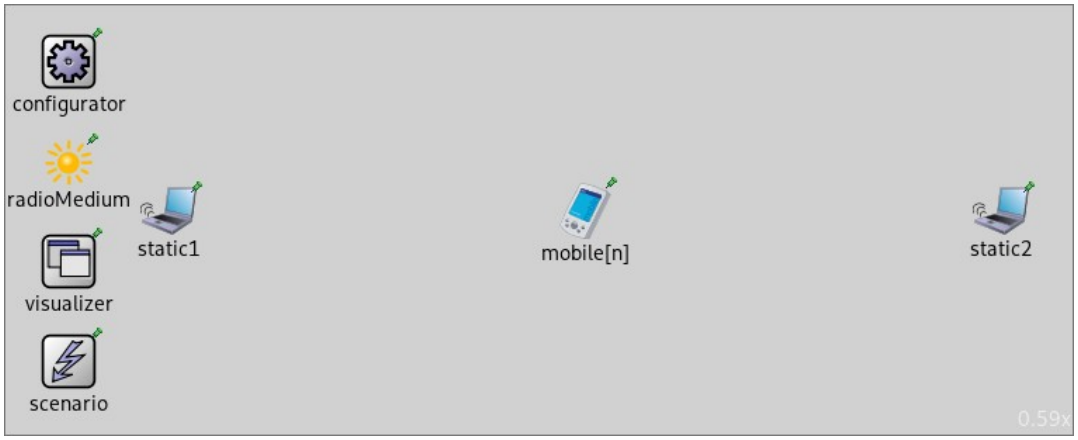
- `memoria.pdf`
- `inet/*.ini`
- `inet/*.ned`
- `inet/*.xml`

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 *Manet.ned,* disponible en Moodle, consiste en 2 nodos estáticos separados por 1000m entre los cuales existe un número variable de nodos móviles. Los nodos estáticos intercambiarán paquetes a través de los móviles, que implementarán dos protocolos de enrutamiento: uno reactivo (AODV) y uno proactivo (DSDV).



Crea un proyecto con el escenario anterior y un fichero de configuración `omnetpp.ini` basado en la siguiente plantilla (también en Moodle):

```
[General]
network = Manet

num-rngs = 2
**.mobility.rng-0 = 1

**.arp.typename = "GlobalArp"

*.configurator.config = xmldoc("config.xml")

**.radio.displayCommunicationRange = true
*.visualizer.networkRouteVisualizer.displayRoutes = true
*.visualizer.networkRouteVisualizer.packetFilter = "UdpBasicAppData*"
*.visualizer.interfaceTableVisualizer.displayInterfaceTables = true
*.visualizer.interfaceTableVisualizer.interfaceFilter = "wlan*"

*.mobile[*].mobility.typename = "RandomWaypointMobility"
*.mobile[*].mobility.speed = 10mps
*.mobile[*].mobility.waitTime = 2s
*.mobile[*].mobility.initFromDisplayString = false
*.mobile[*].mobility.constraintAreaMinX = 200m
*.mobile[*].mobility.constraintAreaMaxX = 1200m
*.mobile[*].mobility.constraintAreaMinY = 50m
*.mobile[*].mobility.constraintAreaMaxY = 450m
*.mobile[*].mobility.constraintAreaMinZ = 0m
*.mobile[*].mobility.constraintAreaMaxZ = 0m

[Config AODV]
**.routingApp.typename = "Aodv"

[Config DSDV]
**.routing.typename = "Dsdv"
**.helloInterval = 3s
```

La plantilla define, en este orden: la configuración de los generadores de números aleatorios; las tablas ARP globales (para evitar tráfico ARP); las direcciones IP fijas para facilitar el seguimiento de la simulación (en el fichero `config.xml`: el nodo *mobile[**x**]* tendrá una IP 1.1.0.**x**); las opciones de visualización (radio de cobertura, rutas seguidas por los paquetes y direcciones IP de los nodos); el tipo de movilidad de los nodos; y, finalmente, las dos configuraciones que se simularán: AODV y DSDV.

Los nodos móviles en ambas redes seguirán un tipo de movilidad aleatoria `RandomWaypointMobility`: cada nodo elige un punto aleatorio del volumen definido por las diferentes variables `constraintArea*` y se desplaza hacia él con una velocidad `mobility.speed`, permanece estático `mobility.waitTime` y vuelve a elegir otro punto aleatorio al que desplazarse. Esto hará que se establezcan y se rompan las rutas entre los nodos estáticos a medida que progrese la simulación.

Para que el movimiento de los nodos sea independiente de otras variables (y de esta forma logremos que el movimiento de los nodos sea idéntico en AODV y DSDV) utilizaremos **dos generadores de números aleatorios** (RNG) distintos (`num-rngs = 2`): uno para la movilidad y otro para el resto de variables. Los componentes usan por defecto el 0, así que es suficiente especificar que la movilidad usará el RNG 1 (`**.mobility.rng-0 = 1`).

Completa la configuración **[General]** que será común a AODV y DSDV:

- Configura en el nodo *static1* una `UdpBasicApp` y en el nodo *static2* una `UdpEchoApp`. El nodo *static1* enviará un datagrama de 100 bytes cada 40 milisegundos hacia *static2*, empezando en `startTime = 10s`.
- Configura **15** nodos móviles (*.n) con una potencia de **2 mW** (`**.radio.transmitter.power`).
- Iguala la semilla aleatoria para el RNG general (`seed-0-mt`) a 0 y **elige al azar** una semilla para el RNG de movilidad (`seed-1-mt`). Puedes ejecutar `echo $RANDOM` en un terminal para obtener un número aleatorio. **Pon la semilla utilizada en la memoria.**

  La semilla elegida debe cumplir que al comenzar la transmisión UDP exista al menos una ruta entre *static1* y *static2*, de manera que **el paquete *UdpBasicAppData-0* llegue a *static2*** tanto en AODV como en DSDV.

Cuestiones

AODV

1.1 ¿Qué nodos reenvían el primer paquete RREQ enviado por *static1*? ¿Y el segundo RREQ? ¿Por qué?

1.2 Elige el nodo intermedio de la ruta que sigue el **primer paquete RREQ que llega a *static2***. Muestra su **tabla de enrutamiento** (vector `<Ipv4route *>` dentro del módulo *ipv4.routingTable*) justo antes y justo después de recibir el primer RREQ. Explica las diferencias y cómo se crean las entradas que aparecen (incluyendo los campos más importantes).

1.3 Haz lo mismo justo antes y justo después del primer RREP.

1.4 Añade las siguientes líneas a la configuración para simular la caída de un nodo en t = 15 s:

```
**.hasStatus = true
```

```
*.scenario.script = xml("<scenario><at t='15'>\
<shutdown module='mobile[x]'/></at></scenario>")
```

Deben existir **al menos 2 rutas alternativas en t = 15 s entre *static1* y *static2* a través de nodos alejados de *static1***. Si la semilla utilizada hasta este punto no lo cumple, elige aleatoriamente otra semilla y **ponla en la memoria**. Haz caer el nodo **x más alejado** de *static1* que permita el restablecimiento de la comunicación entre *static1* y *static2* a través de una ruta alternativa.

¿Cuál es el primer nodo en darse cuenta de la caída? ¿Cómo? Muestra una **captura del log** del nodo que se da cuenta que muestre el motivo. ¿Notifica este nodo la caída del nodo?

1.5 Muestra el **contenido del paquete** RERR en Wireshark explicando los campos más importantes. ¿Qué IP tiene como destino? ¿Por qué?

1.6 Explica cómo se propaga el RERR por la red. ¿Qué nodos lo reenvían? ¿Cómo sabe un nodo si debe reenviar el RERR?

1.7 Muestra capturas de la **tabla de enrutamiento** de un nodo antes y después de recibir un RERR y explica en qué cambia.

1.8 ¿Qué hace *static1* al recibir el RERR? Muestra el **contenido** del siguiente RREQ en Wireshark. ¿En qué cambia con respecto al de la pregunta 1?

DSDV

2.1 Avanza la simulación hasta el instante t = 7 s. Busca **el primer** paquete *Hello* transmitido a partir a ese instante con un valor de *hopdistance* **de al menos 3** y muestra una **captura del contenido**. Explica el significado de los campos *srcAddress* y *nextAddress,* utilizando para explicarlos una captura de **la tabla de enrutamiento** del nodo que **está transmitiendo** el paquete (i.e., **no** el que consta en *srcAddress*).

2.2 ¿Qué valor tiene de *sequencenumber*? ¿Qué quiere decir ese valor?

2.3 ¿Cómo se modifica? ¿Qué nodo lo modifica, y cuándo lo hace?

2.4 Muestra la **tabla de enrutamiento** del nodo que **recibe** el *Hello* de la pregunta anterior justo antes y justo después de recibirlo, relacionándola con el contenido del paquete. Si se actualiza la tabla, explica por qué se actualiza y las entradas que se crean. Si no se actualiza, explica por qué no se actualiza y di qué entrada se crearía (destino, *gateway*, métrica) si se actualizase con la información del paquete.

2.5 Avanza hasta la caída del nodo en t = 15 s. Ten en cuenta que la ruta en ese momento puede ser diferente a la de AODV, y por lo tanto el nodo a desactivar también. ¿Cuál es el primer nodo en darse cuenta de la caída? ¿Notifica la caída del nodo de alguna forma?

Nota: si durante la simulación aparece un error *"Message 'Hello' received when Ipv4 is down"* al caer el nodo, edita el fichero `src/inet/common/lifecycle/OperationalMixinImpl.h` comentando las líneas 77 y 78 responsables del error (*else throw…*) y recompila INET.

2.6 ¿Cómo se repara la ruta entre *static1* y *static2*? ¿En qué momento?

AODV vs. DSDV

3.1 ¿En qué instante se realiza la primera transmisión (de cualquier tipo de paquete) con AODV? ¿Y con DSDV? ¿Por qué?

3.2 ¿En qué instante recibe *static2* el datagrama *UdpBasicAppData-0* con AODV? ¿Y con DSDV? ¿Por qué?

3.3 ¿Cuántos paquetes se pierden como consecuencia de la caída del nodo en AODV? ¿Y en DSDV? ¿A qué se debe la diferencia? (Nota: para calcular el número de paquetes perdidos avanza hasta un instante posterior a la caída en el que en ambos escenarios *static2* vuelva a recibir datagramas y calcula la diferencia entre el número de paquetes recibidos en *static2* con y sin la caída del nodo).

3.4 Con la caída del nodo **desactivada**, simula AODV y DSDV durante 300 s (`sim-time-limit`). Muestra **capturas de los nodos estáticos a nivel de aplicación** (doble click sobre el nodo) al final de la simulación. ¿Qué porcentaje de los *UdpBasicAppData* enviados por *static1* llega a *static2*? Explica los valores y las diferencias observadas.

3.5 ¿Qué porcentaje de **los paquetes devueltos** por *static2* llegan a *static1*? Explica los valores y las diferencias observadas.

AODV vs. DSDV: gráficas

4.1 Obtén una gráfica que muestre el **número de paquetes** recibidos por *static2* **en función de la potencia** tanto para AODV como para DSDV. Para ello añade al .ini el parámetro `repeat` para que realice varios experimentos por configuración (al menos 10), modifica `seed-1-mt` para que genere una semilla diferente para cada experimento y modifica `**.radio.transmitter.power` para que utilice una potencia distinta para cada ejecución: `${inicio..fin step paso}mW`.

Para lanzar la simulación crea una nueva configuración de ejecución (*Run Configurations…*) que utilice *Cmdenv* con el número de CPUs de la máquina virtual (*Allow multple processes*).

Para generar la gráfica añade todos los ficheros de resultado generados durante la simulación al fichero de análisis (.anf) y genera una gráfica seleccionando todos los valores y eligiendo "*Plot using Scatter Chart MPL*". A continuación edita la configuración de la gráfica (*Configure Chart…*) y selecciona como eje X la variable asignada a la potencia y agrupa por "*experiment*".

Explica lo observado en la gráfica.

4.2 Obtén una gráfica similar para el número de nodos. Explica lo observado.

4.3 Obtén una gráfica como las anteriores para la velocidad. Explica lo observado.

4.4 Para DSDV, obtén una gráfica del porcentaje de paquetes perdidos con diferentes valores de *helloInterval*. Explica lo observado.