UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Evaluación de redes IoT-SDN contextuales mediante el simulador SDN-WISE

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Evaluación de redes IoT-SDN contextuales mediante el simulador SDN-WISE

Autor: José Agustín Medina Rodríguez

Tutor: Jaime Galán Jiménez

Co-Tutor/es: Javier Berrocal Olmeda

Índice general de contenidos

Índice de tablas

Índice de ilustraciones

Evaluación de redes IoT-SDN contextuales mediante el simulador SDN-WISE

Resumen

En este momento, los smartphones son dispositivos imprescindibles en nuestra vida cotidiana, que junto a tecnologías como IoT (*Internet of Things*), permiten que nuestro día a día sea mucho más fácil.

Sin embargo, esta tecnología, tiene unas limitaciones bastantes notables referentes a la interacción con los usuarios. Estas carencias, consisten básicamente en que la comunicación que se realiza entre los usuarios y los dispositivos se debe de realizar de forma manual, exigiendo por tanto una inversión de tiempo y esfuerzo, a la hora de querer realizar alguna acción. A parte de esto, estas tecnologías IoT, no toman en cuenta el marco contextual en el que se encuentra, del que se podría obtener información muy valiosa, como la cantidad de personas que hay, sus preferencias, etc. Lo que, ayudaría enormemente a la experiencia de usuario. Por ejemplo, en una habitación con muchas personas, si se detectasen sus preferencias, como la temperatura, las redes IoT podrían actuar en consecuencia e intentar poner una que no le desagrade a ninguno.

Debido a ello, este proyecto propone la creación de redes IoT-SDN contextuales, las cuales, son el resultado de implantar el *Situational Context* en redes SDN y aparecen con el objetivo de solucionar la falta de automatización del entorno a las preferencias de los usuarios. Para ello, se propone una nueva interacción entre los dispositivos IoT y los usuarios de manera más pasiva, en la que los usuarios no deberían de invertir ni tiempo ni esfuerzo, siendo la comunicación más invisible al usuario. En este tipo de redes, el contexto en el que se encuentra toma bastante relevancia, teniendo en cuenta variables físicas del entorno como la cantidad de usuarios, sus preferencias, intereses, etc. que antes no se tenían en cuenta.

En este proyecto por tanto, se define el *Situational Context* en redes SDN, estudiando su impacto en distintos escenarios que se pueden dar en la vida cotidiana. Para ello, se modificara el simulador de redes SDN llamado SDN-WISE para que permita la creación y ejecución de este tipo de redes. A parte de esto, posteriormente se estudiaran los resultados obtenidos en diversos escenarios.

1. Introducción

En la actualidad, en el mundo de la tecnología y la informática hay una palabra que esta resonando en muchas partes, esta es IoT. IoT son las siglas en inglés de Internet of Things o Internet de las Cosas en español y se podría definir, por ejemplo, como:

Un sistema de dispositivos de computación interrelacionados, máquinas mecánicas y digitales, objetos, animales o personas que tienen identificadores únicos y la capacidad de transferir datos a través de una red, sin requerir de interacciones humano a humano o humano a computadora [2]. Así, un elemento en el Internet de las Cosas puede ser, una persona con un implante para monitorizar el corazón, un animal de granja con un transpondedor de biochip, un automóvil que tiene sensores incorporados para alertar al conductor cuando la presión de los neumáticos es baja, o cualquier otro objeto natural o artificial al que se puede asignar una dirección IP y darle la capacidad de transferir datos a través de una red [2]. Pero los elementos de IoT no se limita unicamente a la transferencia de información a traves de la red sino que también son capaces de recibir información y llevar a cabo diversas acciones, por poner un ejemplo, se podria tener un sensor de luminosidad en un domicilio particular el cual detectase la falta de luz y enviase un mensaje al smartphone a los residentes de esta indicándoles si quieren encender las luces o también otro ejemplo bastante claro seria indicarle a un aire acondicionado mediante un smartphone que ponga una temperatura determinada.

Como se puede deducir, la mayoria de las interacciones en las redes IoT actuales necesitan llevar a cabo acciones manuales. Aunque es cierto que se puede programar acciones, estas necesitan una previa inversión de tiempo y esfuerzo por parte de los usuarios. A parte de esto, hay que considerar que estas redes actualmente no tienen en cuenta diversas variables físicas como la posición de los elementos, el desplazamiento, la cantidad, etc. Todas estas circustancias puede llegar a dar informacion muy preciada del entorno o contexto, refiriendonos a este como el conjunto de circunstancia que rodean la red y son capaces de cambiar.

Por tanto, es correcto decir que hay una falta de automatización y tratamiento de la información real referente al contexto, haciendo que esto sea uno de los mayores

incovenientes actuales en los redes IoT. Debido a esto, se ve la necesidad de suplementar estas carencias, surgiendo de esta el concepto de las redes IoT-SDN contextuales.

Las redes IoT-SDN contextuales ofrecen un nuevo tipo de comunicación entre los componentes que forman la red, de manera que trata de solventar los problemas relacionados con la iteracción manual mediante el uso de la información obtenida del contexto, intentando de esta manera la automatización de las acciones. Estas, se realizarian de manera invisible y transparente al usuario, permitiendo una adaptación del entorno en base a la preferencias de los usuarios.

Como se puede ver el eje principal de este proyecto son las redes IoT-SDN contextuales y por ello, en este trabajo se definirá tanto el concepto como el funcionamiento de este tipo de redes. Además, se modificara un simulador para poder realizar tanto el diseño, como la simulación y el analisis de distintos escenarios que involucren este tipo de redes, así como el posterior estudio de los resultados que se han obtenido tras la realización de las simulaciones.

2. Objetivos y Contribuciones

Este proyecto tiene como objetivo principal precisar el término y el funcionamiento de *Situational Context* en redes SDN, apareciendo así las redes IoT-SDN contextuales. También, se realiza la modificación de un simulador que permita la ejecución y estudio de estas redes.

Para este trabajo, se ha visto una gran cantidad de simuladores y se ha concluido que SDN-WISE es el que mejor se adaptaba a estas necesidades ya que permite su alteración. Por tanto lo que se pretende hacer con este simulador es lo siguiente:

- Dotar al simulador de la capacidad de guardar, modificar y calcular contexto: Para ello, se deberá de modificar la implementación de los nodos de este simulador de alguna manera para que sea capaces de retener este contexto, así como también otorgarle al controlador de la red la capacidad de calcular la condición ideal del entorno en el que se encuentra. Debido a esto se ha visto necesario diferenciar entre distintos tipos de nodos aparte del controlador haciendo una pequeña subdivisión en 2 tipos de motes:

    - Nodos *goals*: Son nodos que no modifican el entorno en el que se encuentran, simplemente guardan información sobre cómo les gusta que este el medio en el que se encuentra.
    - Nodos *skills*: A diferencia de los anteriores estos pueden cambiar el estado del marco en el que se encuentran en cambio, estos tampoco pueden guardar sobre preferencias del estado del entorno.
- Diseño y ejecución de distintos escenarios y análisis de los resultados obtenidos: Después de añadir los componentes anteriores a SDN-WISE, se estudiarán distintas situaciones que se pueden dar en la vida real. Para ello, se ha acabado realizando un generador de escenarios el cual crea de forma aleatoria diferentes marcos, intentando imitar de esta manera diversas situaciones en la vida cotidiana para su posterior ejecución y estudio de los resultados.

De esta manera, se pretende traer una herramienta que permita un primer acercamiento a estas redes IoT-SDN contextuales, permitiendo el estudio del comportamiento que tendrían en diferentes situaciones. Intentando de esta forma comprobar si son rentables dado el escenario, el tiempo que tardaría en adaptar el entorno, etc.

3. Redes SDN y Situational Context

Como ya se ha mencionado en apartados anteriores, este trabajo tiene como principal objetivo el estudio del *Sitational Context* en redes SDN, algo a lo que se le ha nombrado como redes IoT-SDN contextuales, pero antes de adentrarnos más en este trabajo, se ve la necesidad de explicar debidamente los que son las redes SDN y el *Situational Context*, así como el concepto de redes IoT-SDN contextuales, además de mostrar la motivación de sus surgimientos.

3.1 Redes SDN

Las redes SDN (*Software-Defined Networks*), son aquellas en las que el control de la red se desliga totalmente del hardware y esta pasa a una aplicación software que usualmente se llama *Sink* o controlador, lo que permite de una manera mucha más fácil la implementación e implantación de servicios de red de forma más dinámica y escalable, aportando numerosas ventajas como:

- Permite la programación del Plano del Control, al estar desacoplado del encaminamiento de los paquetes.
- Como se ha mencionado anteriormente redes más dinámicas, ya que se agiliza el proceso de diseño y modificación de todo el flujo de datos.
- El controlador permite una visión global de la red.
- Se utilizan estándares abiertos, de tal forma que las instrucciones y control se establece por los controladores en lugar de por múltiples protocolos y dispositivos específicos [8].

3.1.1 OpenFLow

Como se ha comentado, en el anterior apartado, las redes SDN, utilizan estándares abiertos. Uno de los más importantes, es el protocolo *OpenFLow* que especifica la interacción entre los planos de control y de datos mencionados anteriormente [8].

El funcionamiento de este protocolo a grandes rasgos se puede dividir en 3 partes:

- *Sink* o Controlador: Que es donde reside el plano de control de las redes SDN.
- Agente OpenFlow: Que se encarga de la creación de las tablas de encaminamiento de la red.
- Protocolo OpenFlow: Que controla la comunicación entre el *Sink* y los demás dispositivos que forman la red.

Como se puede ver, estas redes se pueden trasladar a diversos ámbitos. Uno de ellos las redes IoT y como se ha visto en anteriormente, uno de los ejes principales de este proyecto es la adaptabilidad del entorno. Pero aún falta un problema bastante grande por resolver, el cual, consiste en que esta adaptabilidad del entorno se realice de forma transparente a sus usuarios. Es en este punto, donde entra el concepto de *Situational Context*.

3.2 Situational Context

La aparición de este concepto nace como consecuencia de dar una solución al *Ambient Intelligent*, el cual, basado en la computación ubicua, fundamenta en que cada persona interaccione con una multitud de dispositivos programables [5] para la modificación del entorno en el que se encuentra.

Dada la abundancia de estos dispositivos, necesariamente habrían de ser manejados con mínimo esfuerzo, de forma que la interacción entre sujeto y máquina sea transparente al

usuario. La simple presencia de la persona debería bastar para que la máquina interactuase con ella, sin que el usuario tenga que hacer nada de forma consciente. Además, estos dispositivos estarían presentes en todos los ámbitos de la vida diaria, desde el entorno laboral hasta el desarrollo del ocio [5] y esto es justamente a lo que viene dar solución el *Situational Context*.

El *Situational Context* (contexto situacional), se puede definir como la composición de los perfiles virtuales de todas las entidades involucradas en una situación espacio-temporal [1]. Como se ha podido ver en la definición, es en este momento en el que se empiezan a definir el concepto desde un punto puramente teórico el término de los perfiles virtuales. Estos perfiles virtuales, son estructuras que guardan los datos referentes a los gustos y preferencias, así como las acciones que pueden realizar determinados elementos.

El *Situational Context* hace principalmente una distinción entre 2 elementos de la red:

- *Goals*: Estos hacen referencia a los usuarios, que quieren modificar el entorno a su gusto. En estos nodos, el perfil virtual que se guarda son los gustos y preferencias de las personas.
- S*kills*: A diferencia del anterior, estos hacen referencia a los distintos elementos capaces de modificar el entorno. En este caso, el perfil virtual guarda las acciones que pueden realizar los dispositivos. Las modificaciones que realizan estos aparatos en el entorno, se hacen en base a los gustos de los de los usuarios de la red.

Una vez visto esto, se puede deducir que la adaptación de estos dispositivos ubicuos que nos planteaba el *Ambient Intelligent* a las preferencias de las personas con su simple presencia, puede ser posible mediante el concepto de perfiles virtuales planteando el uso de elementos *Goals* y *Skills* y cuya interacción permite la adaptabilidad del contexto en el que se encuentran de manera automática siguiendo los gustos especificados por los usuarios en los elementos *Goals*, las cuales son recogidas por los *Skills* intentando realizar el cambio del entorno correspondiente.

3.3 Redes IoT-SDN contextuales

Como se ha visto en el apartado de objetivos, en este proyecto se propone la implantación y evaluación de redes IoT-SDN contextuales, las cuales, son el resultado de la unión de la implantación del *Situational Context* en redes SDN.

El objetivo de esta unión, es la automatización del entorno que nos rodea para que se adapte a los gustos de las personas, evitando así la interacción de humano para su adaptación. Para ello, utiliza el dinamismo que proporcionan las redes SDN y el concepto de perfiles virtuales definido en *el Situational Context*

Por tanto, este tipo de redes requieren que los componentes que la forman tengan la capacidad de pasar sus preferencias, así como la capacidad de utilizar algún tipo de algoritmo que permita una adaptación del entorno en la que estén conformes todos sus usuarios. Debido a esto, debemos de realizar una distinción entre 3 tipos de nodos:

- Nodos *goals*: Aquellos que guardan las preferencias de los usuarios
- Nodos *Skills*: Aquellos que tienen la capacidad para modificar el entorno según las preferencias de los nodos *goals*.
- *Sink* o controlador: Aquel que recoge los perfiles de los nodos *goals*, realiza los cálculos oportunos y le indican a los nodos *skills* que cambios deben de realizar en el contexto.

La comunicación en este tipo de redes funciona de la siguiente manera:

- En un principio las preferencias, es decir, el perfil virtual se guardan en los nodos *goals*, el cual como se ha mencionado en la introducción de este proyecto son los smartphones.
- Una vez que se han guardado las preferencias y el *sink* está activado, este desde un primer momento lo que hace es mapear la red, para saber la cantidad de nodos que la forman.
- Después de que la red ha sido mapeada al completo, el *sink* procede a enviar una trama a todos los nodos indicándole que envíen sus perfiles virtuales.
- Cuando el *sink* ha terminado de recibir todos los perfiles virtuales ya sabe de qué tipo de nodo es cada uno, por lo que procede a calcular distintas funciones como por ejemplo la temperatura más agradable para todos los usuarios, la luminosidad, etc. Según las preferencias obtenidas de los perfiles virtuales de los nodos *goals*.
- Una vez calculada las distintas funciones estas son enviadas a los nodos *skills* correspondientes, capaces de modificar el contexto al cual se refiere cada función.

4. Simulador SDN-WISE

El simulador SDN-WISE contiki, creado por el equipo de S. Milardo, pretende ser una solución de red definida por software para *Wireless Sensor Networks*, cuyo principal objetivo es simplificar la gestión de la red, el desarrollo de nuevas aplicaciones y la experimentación de nuevas soluciones de red [6]. Por lo que podemos decir que se trata de un programa que permite simular distintos escenarios con nodos de distintos tipos.

Gracias a que el proyecto SDN-WISE es *Open Source*, su código, el cual, está realizado en Java, se encuentra disponible en Github, permitiendo su descarga y modificación de los elementos que lo componen.

4.1 Nodos de SDN-WISE

En un primer momento, las simulaciones de SDN-WISE cuentan con 2 tipos de nodos:

- *Sink*: Es el nodo controlador del escenario, llevando a cabo, la creación y la supervisión de la red formada en la simulación.
- *Motes*: Estos nodos simplemente tienen la capacidad de crear, enviar y recibir los distintos paquetes que le llegan. Este tipo de nodo, es el que representaría a los distintos dispositivos IoT, dentro de la simulación.

4.2 Paquetes de SDN-WISE

Como se ha indicado en el apartado anterior, los nodos tienen la capacidad de crear, enviar y recibir paquetes. Los principales paquetes usados en las simulaciones son:

- ***Data***: Este paquete sirve principalmente para enviar datos del *Sink* a un *Mote*. Tiene los siguientes campos:
  - *Header*: Es la cabecera de los paquetes de SDN-WISE.
  - *Payload*: Se refiere al contenido de datos.
- ***Beacon***: Se utiliza para ver que nodos se encuentran en la red, a que distancia y que batería tienen. Contiene los siguientes campos:
  - *Header*: Que es la cabecera de los paquete de SDN-WISE.
  - *Distance*: Que indica la distancia desde el *Sink* a los otros nodos en nº de saltos.

  - *Battery*: Nivel de batería del *Sink*.

- ***Report***: Es el paquete de respuesta que envían como respuesta de los paquetes *Beacon* a los nodos al *Sink*. A parte de los campos del paquete *Beacon*, este también incluye:
  - *NeigborsSize*: Que indica la cantidad de nodos vecinos que tiene un nodos.
  - *NeighborAddress*: Muestra la dirección de los nodos vecinos.
  - *LinkQuality*: Que contiene la calidad de la señal de los nodos vecinos.

Destacar, que la cabecera de los paquetes contiene las siguientes secciones:

- **NET**: Indica el identificador de la red.
- **LEN**: Muestra la cantidad de bytes que tiene el paquete.
- **DST**: Hace referencia al nodo destino del paquete.
- **SRC**: Indica el nodo origen del paquete.
- **TYP**: Indica el tipo de paquete que es siendo:
  - 0: paquete de tipo *Data*.
  - 1: paquete de tipo *Beacon*.
  - 2: paquete de tipo *Report*.
- **TTL**: Muestra el tiempo de vida del paquete.
- **NXH**: hace referencia a la dirección del próximo salto.

4.3 Entorno de simulación

Una vez visto que es SDN-WISE, se pasara a mostrar la interfaz de este simulador, empezando por la pantalla de inicio, como se observa en la ilustración 1:

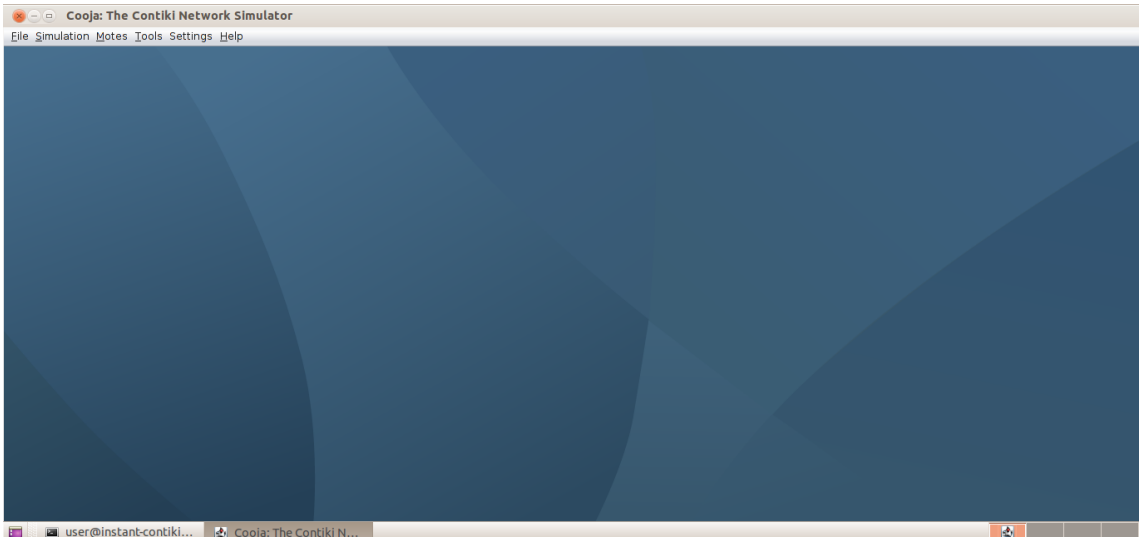

Ilustración 1: Ventana de inicio simulador

Como se puede observar la interfaz no guarda mucho misterio, teniendo en la parte superior los típicos botones de cerrar, minimizar y expandir la ventana, siguiendo en la barra de herramienta se pueden encontrar 6 apartados:

- **File:** su función consiste principalmente en seleccionar 2 categorías, crear una nueva simulación o abrir una ya hecha.
- **Simulation:** cuyo uso radica en tener algunas funciones de la simulación como empezar, recargar, abrir el panel de control de la simulación, etc.
- **Motes:** este apartado tiene la capacidad de seleccionar uno o varios nodos para añadirlos a la simulación o eliminarlos.
- **Tools:** esta sección de la barra de herramientas, nos permite visualizar varias herramientas que pueden ser de utilidad a la hora de realizar una simulación.
- **Settings:** con esta opción se permite modificar la configuración del simulador, así como añadir extensiones, etc.
- **Help:** cuya función radica en aportar una pequeña ayuda al usuario sobre cómo usar el simulador, los atajos del teclado, etc.

Una vez vista la interfaz del simulador de contiki, se mostrara a continuación como crear un escenario básico:

1. Lo primero que se debería de hacer es dirigirse a la opción *File > New simulation…*, con lo que aparecerá una ventana como la que aparece en la ilustración 3 en la cual se debe introducir el nombre de la simulación, etc. Como se aprecia en la ilustración 2.

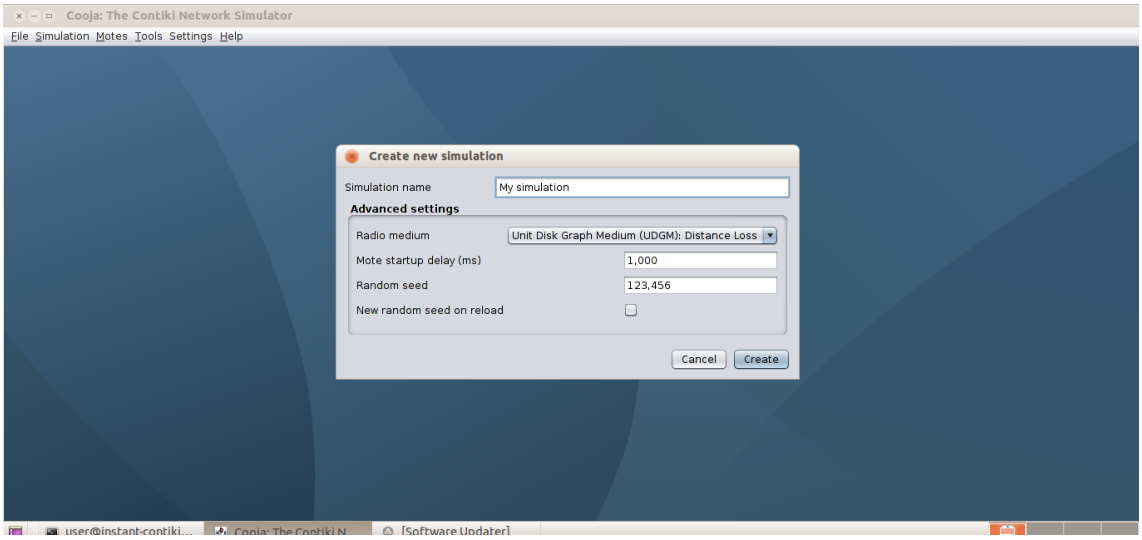

Ilustración 2: creación de simulación 1

2. Una vez que se hayan escogidos los parámetros deseados y después de pulsar sobre el botón de "*Create*", aparecerá una pantalla como la que se muestra en la Ilustración 3, en la cual como se puede observar, tenemos varias ventanas, cada una con una función diferente:
    - ***Network***: En la que se visualizaran los elementos de la simulación.
    - ***Simulation control***: Proporciona los elementos necesarios para controlar varios parámetros de la simulación, como iniciar la simulación, pararla o detenerla, así como la capacidad de limitar la velocidad de esta, mediante la opción de *Speed limit*, localizada en la parte superior, etc.
    - ***Mote output***: Indica los paquetes que van saliendo de cada nodo.
    - ***Timeline*:** Muestra líneas de tiempo de acción de los distintos nodos que se tienen en la simulación.

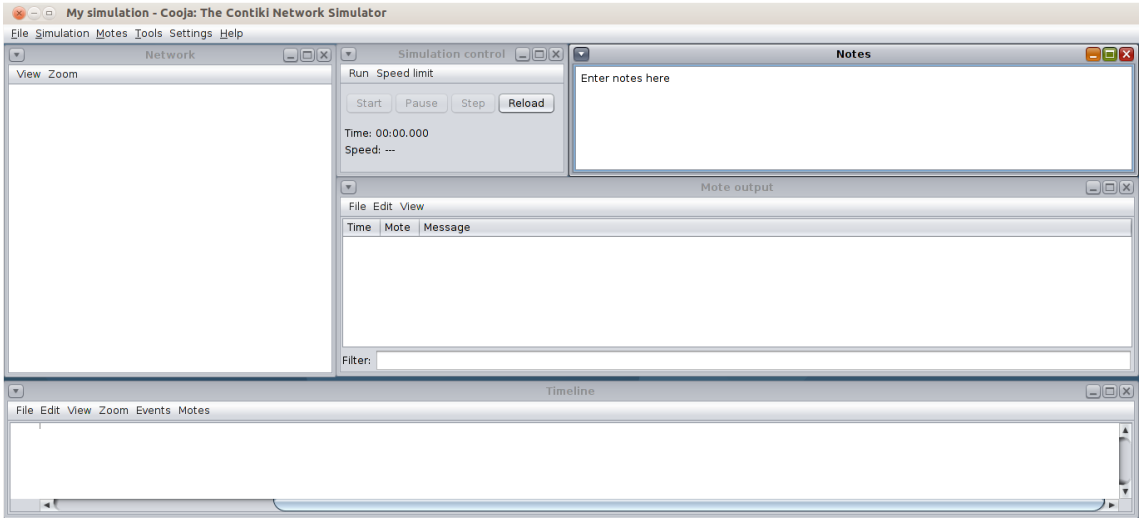

Ilustración 3: creación de simulación 2

Una vez vistos los elementos más importantes que forma la vista principal de la simulación se pasara a mostrar como añadir nodos y modificar un poco la visualización de esta para que sea algo más gráfica y "amigable" para el usuario.

1. Para añadir elementos a la simulación se debe de dirigir al apartado Motes de la barra de herramientas y seleccionar *Add motes* > *Create new mote type* > *Import Java mote*…, con lo que se abrirá una ventana como la que se muestra en la ilustración 4 la cual permite añadir algunas cosas como la descripción que se le quiere dar al nodo, etc.

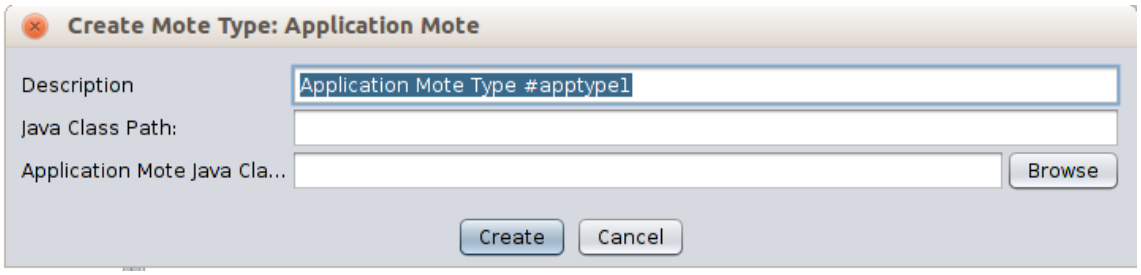

Ilustración 4: creación de simulación 3

2. Una vez se haya añadido una descripción al nodo se procede a seleccionarlo, en una primera instancia, se recomienda siempre que el primero de todos sea el *Sink*, para buscarlo simplemente se debe de pulsar sobre *Browse* y seleccionar *Sink.class* como aparece en la Ilustración 5:

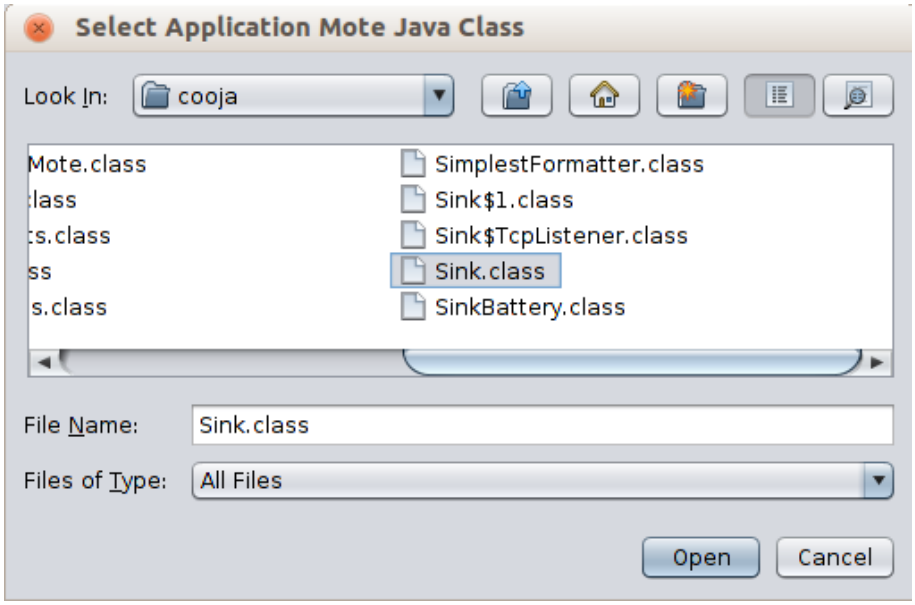

Ilustración 5: creación simulación 4

3. Una vez seleccionado, aparecerá una nueva vista, en la cual se pude modificar la cantidad de nodos que se quieren añadir a la simulación, como se puede apreciar en la Ilustración 6:

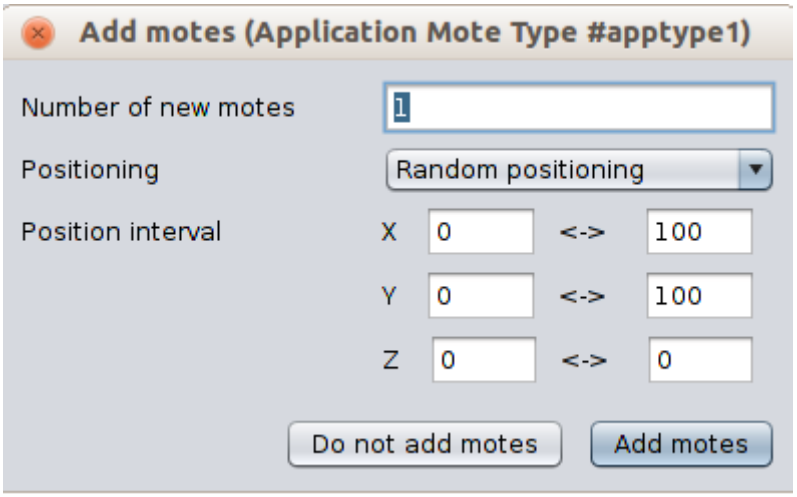

Ilustración 6: creación simulación 5

4. Por último, una vez que se haya pulsado sobre el botón de *Add motes*, aparecerá una última ventana para indicar la dirección IP y el puerto TCP del controlador.

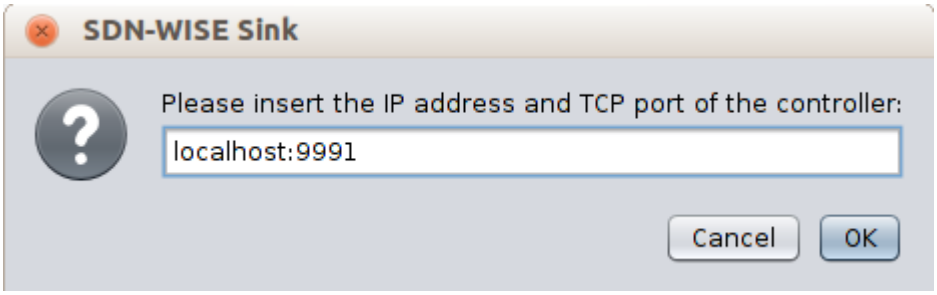

Ilustración 7: creación simulación 6

Una vez realizado esto, se debe de hacer lo mismo para los demás nodos, para ello, en vez de elegir *Sink.class* como aparece en la ilustración 5, se debe de elegir *Mote.class* y posteriormente seguir los mismos pasos que con la creación del *Sink*.

Una vez añadido y creado un nodo aparecerá la simulación como se muestra en la ilustración 8:

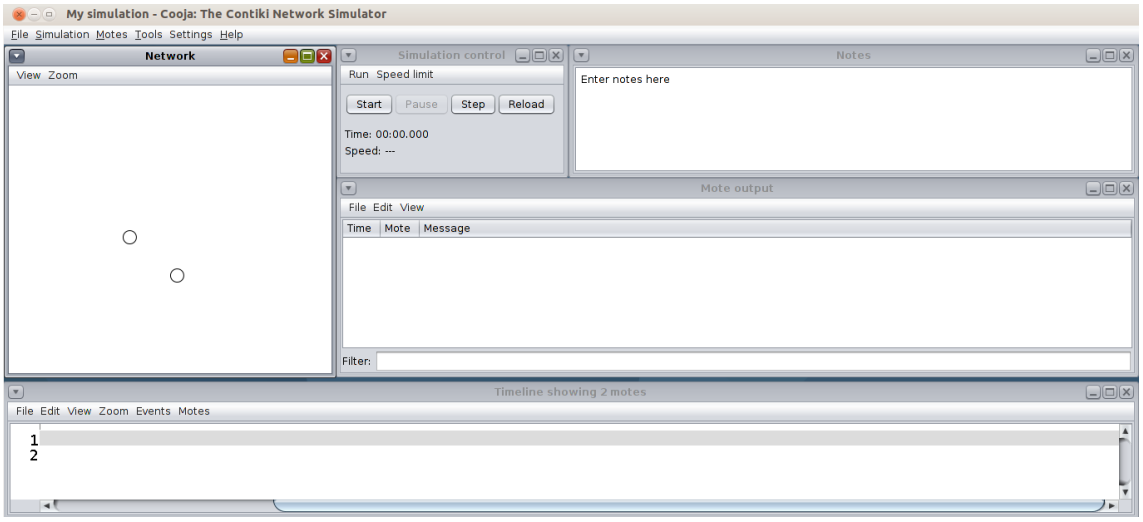

Ilustración 8: creación simulación 7

Como se mencionó anteriormente, se va a modificar un poco la apariencia de este para que sea más amigable para el usuario, para ello, nos vamos a la opción *View* de la ventana *Network* y se seleccionan las opciones de *Mote IDs*, *Radio traffic*, *10m background grid*, *Mote type* y *Radio enviroment*, quedando al final la simulación como se observa en la ilustración 9:

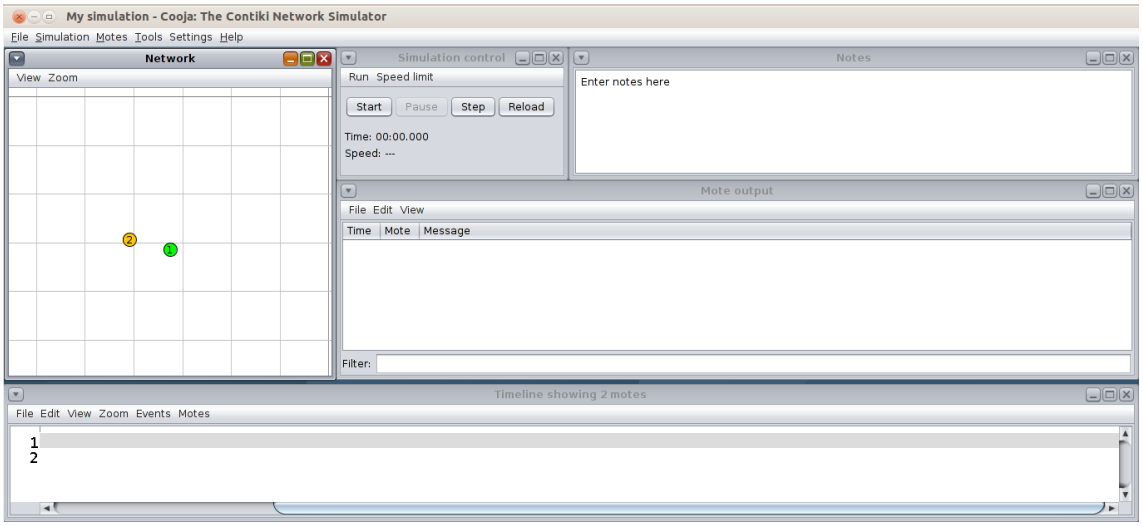

Ilustración 9: creación simulación 8

Destacar también que para las simulaciones que se han planteado, se ha reducido el alcance de los nodos a 10 metros, intentando simular de esta manera la tecnología Bluetooth. Para ello, se pulsa sobre cualquier nodo y se selecciona la opción de *change transmission ranges* y los valores de *TX range* e *INT range* se fijan a 10 y 0 respectivamente.

Una vez realizado para iniciar la simulación, se debe de pulsar el botón *Start*, localizado en la ventana de *Simulation control*, pero antes de eso, se debe de descargar, compilar y ejecutar el código correspondiente al controlador en otra terminal, ya que debido a la estructura de funcionamiento de SDN-WISE, el código referente al controlador se encuentra separado del entorno de desarrollo de simulaciones por lo que se necesita ejecutar de forma separa a este.

4.4 Simulación

Una vez visto el entorno de simulación así como los paquetes más usados, se procede a mostrar la interacción que siguen estos. Una simulación de SDN-WISE se puede dividir en 2 etapas:

1. Una vez empezada la simulación. Se espera un tiempo a que la red "despierte", es decir, a que el controlador mapee la red, como se muestra en la ilustración x.

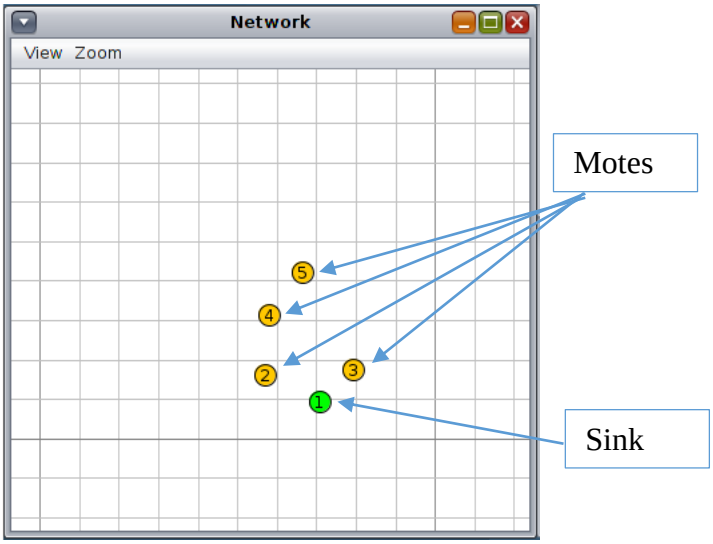

Ilustración 10: Simulación SDN-WISE 1

2. Una vez que el sink mapeado la red, este envía paquetes de tipo Data con la información que quiere enviar, en su interior, como se aprecia en la ilustración

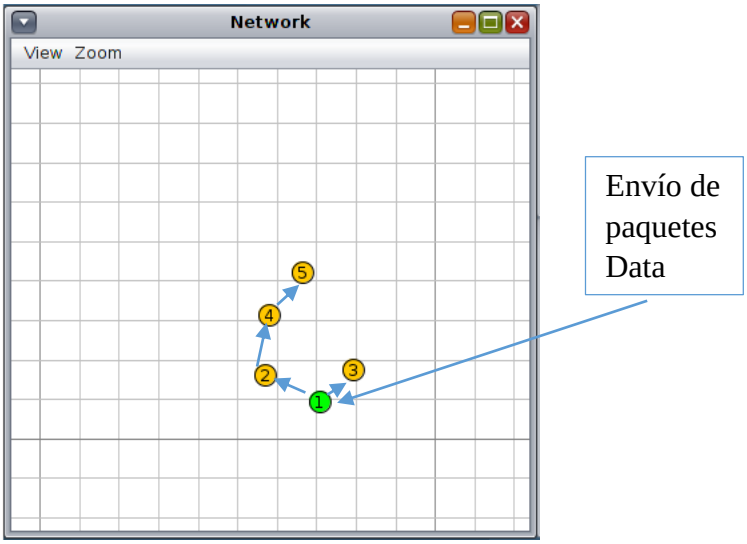

Ilustración 11: Simulación SDN-WISE 2

5. Propuesta implementación Situational Context sobre SDN-WISE

Se ha decidido utilizar el simulador de SDN-WISE, debido a que como se ha visto en la anterior sección, este simulador está programado en Java, es de código abierto y con la capacidad de añadir distintos tipos de nodos, lo que lo hace un candidato ideal para modificarlo y poder aplicar el concepto de *Situational Context* en él.

Esta fase se divide en varias etapas. Se han especificado varios requisitos que se precisa que tenga el simulador. Posteriormente se han realizado los correspondientes análisis y diseño de los requisitos, para acabar procediendo a la implementación de estos, con el objetivo de poder realizar un estudio sobre las redes SDN con *Situational Context*.

Por lo tanto, desde un primer vistazo, esta sección se puede desglosar en:

1. Especificación de los requisitos.
2. Análisis y diseño de los requisitos.
3. Implementación de los requisitos según el diseño propuesto.

5.1 Especificación de los requisitos

Tras analizar las características del simulador, en esta etapa se mostrarán las propiedades que el simulador ya trae y se requieren, así como también indicar aquellas de las que carece y son imprescindibles para poder estudiar y simular las redes SDN contextuales.

Tras el análisis del programa, a grandes rasgos se puede indicar que contiene las siguientes funcionalidades que son necesarias:

- **Creación de escenarios y agregación de nodos:** Esta función encontrada dentro del simulador, permite desde la creación así como la el guardado, la modificación y la eliminación de distintos escenarios, además de otorgar la capacidad de añadir distintos tipos de nodos, cada con sus correspondientes características.
- **Establecimiento de rango de cobertura:** Esta función que ya viene por defecto en el simulador, permite al usuario darle a los nodos un rango de cobertura. Esta acción se puede realizar de 2 formas diferentes, modificando el parámetro de

rango de cobertura que se encuentra dentro del archivo *.csc* donde se guarda el escenario, o bien mediante el entorno gráfico.

- **Envió y recepción de mensajes:** La capacidad de los nodos de recibir y enviar información mediante distintos tipos de paquetes, ya viene incluida en el programa. Además, este envío y recepción de mensajes, utiliza el protocolo TCP.

Aparte de estas funciones básicas ya existentes en el simulador de SDN-WISE, se necesitan muchas otras funcionalidades para poder realizar una simulación de redes contextuales y los cuales, ha sido necesaria su posterior implementación. Estos son:

- **Adición del perfil virtual a los nodos:** Los nodos deben de poder guardar información relacionada con los *goals* o *skills*.
- **Añadir capacidad de envío de datos relacionados con el contexto:** Aunque SDN-WISE ya incorpora la capacidad de envío de mensajes, estos no están adaptados al modelo de envió de datos del envío del perfil virtual que se necesita en las redes contextuales.
- **Incorporar la capacidad de recibir perfiles virtuales y adaptarse al contexto:** De la misma manera que en el anterior requisito, aunque el simulador ya trae por defecto la capacidad de recibir paquetes de datos de los nodos, esto se debe de adecuar para la recepción de los distintos perfiles virtuales que guardan los nodos.
- **Incorporar la capacidad de creación de distintos informes:** El simulador debe de ser capaz de generar distintos informes para así poder realizar un posterior estudio de los resultados obtenidos.

Aunque estos requisitos son los principales a desarrollar, también se ha visto que en acciones como la creación de escenarios o la de los archivos de configuración de los nodos, el usuario puede perder bastante tiempo. Debido a esto, se han implementado las siguientes funcionalidades adicionales para la generación automática de escenarios y pruebas.

- **Generador de escenarios aleatorios:** Para reducir el tiempo en el que se crean las simulaciones, se propone la creación de un generador de escenarios que cree estos de manera automática.
- **Generador de archivo de configuración de nodos:** De la misma manera que en el anterior generador, pretende reducir el tiempo de creación de las simulaciones,

en este caso se pretende reducir el de la creación de archivos de configuración de los nodos.

- **Generador de escenarios con nodos de movilidad:** Parecido al primer generador, este pretende simular el paso del tiempo en un escenario modificando la posición de los nodos.

Una vez que se ha visto lo que se quiere realizar en el simulador de SDN-WISE que se procede a pasar a la siguiente menciona anteriormente en el inicio del apartado.

5.2 Análisis y diseño de los requisitos

Esta fase que es la continuación de la anterior, pretende dar una solución a los requisitos planteados en la etapa anterior. De esta manera, a continuación se indica una solución a cada requisito definido anteriormente.

5.2.1 Adición del perfil virtual a los nodos

Teniendo en cuenta que el perfil virtual de un nodo básicamente son las preferencias (en el caso de los *goals*) o las habilidades (en el caso de las *skills*) que posee dicho nodo. Se ve la necesidad de añadir los parámetros necesarios según sea el caso para poder guardar dicho perfil. De esta manera la solución a este problema pasa por:

1. Dividir los nodos en nodos *goals* y nodos *skills* cada uno independiente al resto de nodos de la simulación.
2. Añadir los atributos necesarios para poder guardar el perfil virtual en los nodos.

5.2.1.1 División de nodos

En una primera instancia la jerarquía de los nodos como se puede ver en la ilustración 12 sigue la siguiente estructura:

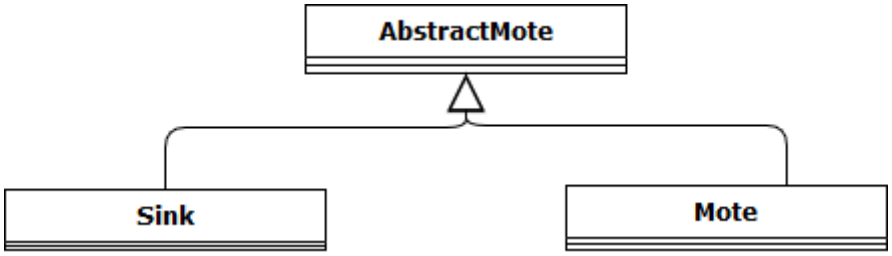

Ilustración 12: Jerarquía de nodos original

Como se puede observar simplemente hay una clase padre llamada *AbstractMote*, de la cual heredan las clases *Sink* y Mote. Mencionar también que obviamente hay muchas más clases y elementos que son utilizas por estas pero debido a que no ha sido necesario su modificación y que en su mayoría contenían constantes, etc. Para simplificar se ha decidido dejarlo de esta manera.

A esta herencia como se puede apreciar en la ilustración 13, se le ha añadido una clase más, la cual se ha llamado *MoteSkills*, quedando ahora de la siguiente manera:

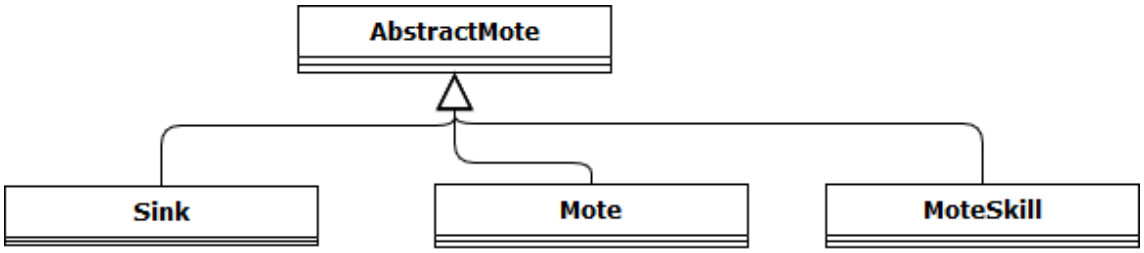

Ilustración 13: Nueva jerarquía de nodos

Aunque tanto los nodos con *goals* como los nodos con *skills* cuentan con una estructura de implementación similar, se ha decidido separarlos en clases diferentes para poder gestionar mejor las acciones que los diferencian, como se puede ver en la ilustración 14. A grandes rasgos, estas diferencias están relacionadas con la introducción de los perfiles virtuales y a la gestión del envió de estos perfiles, así como en el caso de los nodos con *skill*, la capacidad de recibir los datos correspondientes al contexto ideal calculado por parte del controlador.

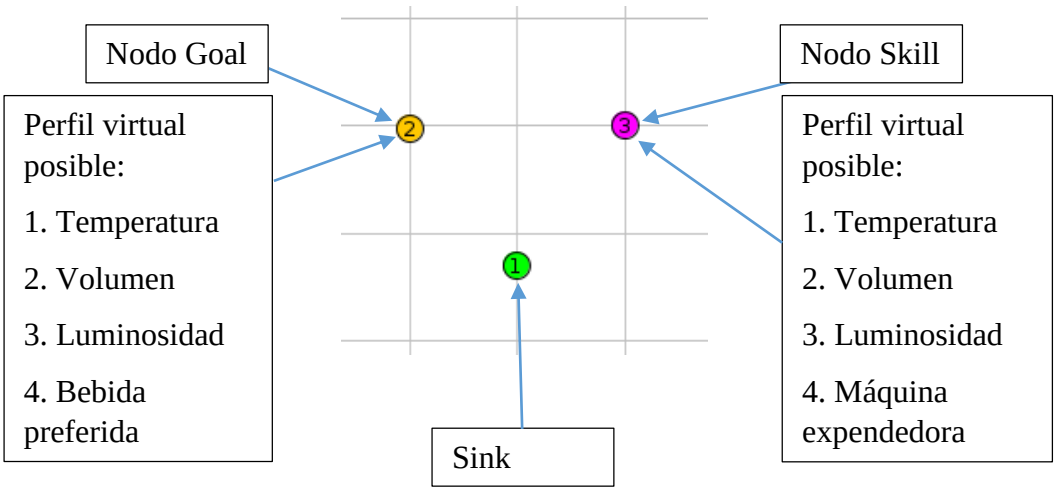

Ilustración 14: Nodos goals y nodos skills

5.2.1.2 Añadir atributos y métodos

Una vez que se ha diferenciado entre los nodos *goals* y *skills*, se ve la implantación de los atributos, así como la creación y modificación de los métodos necesarios para gestionar los correspondientes perfiles virtuales. Por lo tanto, a gran escala, se plantean las modificaciones y creaciones de los siguientes métodos y la implantación de los atributos que se pueden ver a continuación para los distintos tipos nodos:

- Mote:
    - Atributos:
        - *NodeType*: que es un integer.
        - *Function_my*: de tipo integer.
        - *ArrayContext*: conjunto de integer.
    - Métodos:
        - setInformation: para meter la configuración del nodo.
        - writeBeacon: para la generación de informes de paquetes beacon recibidos.
        - rxBeacon: modificando el recibimiento de los paquetes de tipo beacon.
        - prepareReport: modificando la creación y el envió de paquetes de tipo report.
- MoteSkill:
    - Atributos:
        - *NodeType*: de tipo integer.
        - *Function_my*: de tipo integer.
        - *skillType*: de tipo integer.
        - *context*: de tipo integer.
    - Métodos:
        - setInformation: para meter la configuración del nodo.
        - writeBeacon: para la generación de informes de paquetes beacon recibidos.
        - writeData: para la generación de informes de los paquetes data recibidos.
        - rxBeacon: modificando el recibimiento de los paquetes de tipo beacon.

- prepareReport: modificando la creación y el envió de paquetes de tipo report.

Lógicamente cada atributo cuenta con sus respectivos *gets* y *sets*, etc. A grandes rasgos, la diferencia entre estos, radica en que mientras los nodos *goals* tienen un array de contextos en el cual se indica se guardan sus preferencias del entorno, en los nodos con *skills* en cambio tiene un atributo llamado *skillType* el cual guarda el tipo de habilidad que tiene y el atributo *context*, el cual contiene el valor del contexto que tiene actualmente.

5.2.2 Añadir capacidad de envío de datos relacionados al contexto

Después de indicar los cambios realizados sobre las clases motes para la introducción del perfil virtual, en esta función se verá una propuesta sobre cómo llevar a cabo el envío y la recepción de la información relacionada al contexto. Partiendo del funcionamiento y la implementación base otorgada por el equipo de SDN-WISE, se propone la siguiente modificación.

Principalmente consiste en la ampliación de 2 paquetes de datos del simulador SDN-WISE, los cuales son los paquetes *Beacon* y *Report*, que ya se han visto en secciones anteriores. A estos paquetes, se les ha añadido 1 y 2 bytes respectivamente. Estos bytes añadidos como se puede ver en la Tabla 2 y en la Tabla 5, sirven para indicar por parte del *sink* a los nodos el envío de los perfiles virtuales a través de un paquete de tipo *Beacon* y la recepción de estos perfiles a través de *Reports* enviados por los nodos como respuesta al *sink*.

A continuación se mostraran tablas comparando los cambios introducidos a estos paquetes.

- Paquete *Beacon* original

Byte(s)	Nombre	Descripción
0-9	*Header*	Cabecera de paquete SDN-WISE
10	*Distance*	Distancia desde el sink en nº de saltos
11	*Battery*	Nivel de bateria. Entera=0xFF, Vacia = 0x00.

Tabla 1: Paquete Beacon original

- Paquete *Beacon* modificado

Byte(s)	Nombre	Descripción
0-9	*Header*	Cabecera de paquete SDN-WISE
10	*Distance*	Distancia desde el sink en nº de saltos
11	*Battery*	Nivel de bateria. Entera=0xFF, Vacia = 0x00.
12	*Function*	Indica si el paquete beacon va a activar las funciones de los distintos nodos 0 = Nodos "dormidos", 1 = Nodos activados.

Tabla 2: Paquete Beacon modificado

Como se puede observar únicamente cambia en que se ha agregado un byte más al paquete. Este byte extra, tiene el objetivo de decirle a nodos que pueden enviar su perfil virtual, a través de los paquetes *Reports*. Destacar también que la cabecera de un paquete SDN-WISE es la siguiente:

Byte(s)	Nombre	Descripción
0	*NET*	Identificador de la red.
1	*LEN*	Tamaño total del paquete.
2-3	*DST*	Dirección de destino.
4-5	*SRC*	Dirección de origen.
6	*TYP*	Tipo de paquete.
7	*TTL*	Nº de saltos restantes.
8-9	*NXH*	Dirección del siguiente salto.

Tabla 3: Cabecera paquetes SDN-WISE

Seguidamente se mostrará el paquete *Report* tanto el original como el modificado y se pasara a explicar estos cambios introducidos:

- Paquete *Report* original

Byte(s)	Nombre	Descripción
0-9	*Header*	Cabecera de paquete SDN-WISE.
10	*Distance*	Distancia desde el sink en nº de saltos.
11	*Battery*	Nivel de bateria. Entera=0xFF, Vacia = 0x00.
12	*NeigborsSize*	Número de nodos en la lista de vecinos.
13-14	*NeighborAddress 1*	Dirección del primer nodo en la lista.
15	*LinkQuality 1*	Rssi entre el primer nodo y la fuente.
…-…	*NeighborAddress n*	Dirección del nodo n en la lista.
…	*LinkQuality n*	Rssi entre el nodo n y la fuente.

Tabla 4: Paquete Report original

- Paquete *Report* modificado

Byte(s)	Nombre	Descripción
0-9	*Header*	Cabecera de paquete SDN-WISE.
10	*Distance*	Distancia desde el sink en nº de saltos.
11	*Battery*	Nivel de bateria. Entera=0xFF, Vacia = 0x00.
12	*Function*	Indica si se ha activado el nodo 0 = Nodo “dormido”, 1-10 = Nodo activo, indica el tipo de contexto que se envia (temperatura, volumen, luminosidad, etc),

		11 = Nodo activo, indicación de nodo con skill
13	*Information*	Si campo Function = 1-10, indica el valor del contexto correspondiente del nodo. Si campo Function = 11, indica el tipo de skill que tiene el nodo.
14	*NeigborsSize*	Número de nodos en la lista de vecinos.
15-16	*NeighborAddress 1*	Dirección del primer nodo en la lista.
17	*LinkQuality 1*	Rssi entre el primer nodo y la fuente.
…-…	*NeighborAddress n*	Dirección del nodo n en la lista.
…	*LinkQuality n*	Rssi entre el nodo n y la fuente.

Tabla 5: Paquete Report modificado

Como se puede apreciar a simple vista en las tablas 4 y 5, se han añadido 2 bytes los cuales como se indican en la tabla el campo *function* indica si el nodo ya puede enviar información al *sink* para lo cual tomaría valores del 1-11 como se muestra en la tabla o no en cuyo caso el valor de este campo es 0. El siguiente campo a tener en cuenta es el campo *information* el cual, según el valor que se tenga en el campo explicado anteriormente toma un valor u otro, si el campo *function* toma valores de 1 a 10 este indicara el valor que se quiere para ese contexto, mientras que si el valor *function* toma el valor de 11, este campo indicara con valores de 1 a 10 el tipo de *skill* que tiene. Cabe destacar que aunque se ha dejado en una primera instancia 10 espacios para definir diferentes contextos, por simplicidad, en este trabajo únicamente se han utilizado 4, los cuales se muestran en la tabla 6:

Número	Contexto
1	Temperatura
2	Volumen
3	Luminosidad
4	Máquina expendedora

Tabla 6: Relación número contexto

Una cosa a tener en cuenta de esto es que mientras el campo *Information* de los paquetes report pasan una información de carácter puramente numero por ejemplo la temperatura del aire acondicionado a 23ºC, con respecto a la máquina expendedora, este campo lo que representa son opciones de compra como se puede apreciar en la tabla 7:

Número	Opción
1	Agua mineral
2	Refresco de naranja
3	Refresco de limón
4	Refresco de cola
5	Refresco de té

Tabla 7: Relación número opción

Destacar que estos contextos y funciones creados son totalmente configurable y el usuario pude cambiarlos, creando los suyos propios.

Una vez visto esto, se debe de destacar que aunque no se haya modificado el paquete de *Data* de SDN-WISE, también se hace uso de este para enviarle a los nodos con *skills* las

órdenes por parte del controlador. Estos paquetes tienen la estructura mostrada en la tabla 8.

Byte(s)	Nombre	Descripción
0-9	*Header*	Cabecera de paquete SDN-WISE.
10-…	*Payload*	Contenido de datos

Tabla 8: Paquete Data

5.2.3 Incorporar la capacidad de recibir y calcular el contexto ideal

Teniendo en cuenta que el *sink* ya tiene la capacidad para la recepción y el envío de los distintos paquetes que se incluyen en el simulador de SDN-WISE se propone lo siguiente:

1. Añadir capacidad de guardar los distintos datos referentes al contexto que se reciben.
2. Creación de un algoritmo que gestione el recibimiento de los datos así como el cálculo y el envió del contexto ideal calculado a los nodos *skills*.

5.2.3.1 Gestión de datos recibidos

Para satisfacer estas necesidades, se plantea la modificación del código del controlador. Para ello, básicamente se propone la introducción de un par de atributos de tipo lista (uno para los nodos *goals* y otro para los *skills*), en el cual se vayan recogiendo los distintos datos referentes a los perfiles virtuales que van llegando, así como también comprobar que no se metan datos repetidos del mismo mote.

5.2.3.2 Creación del algoritmo

La gestión del recibimiento de los datos así como el cálculo y el envío del valor del contexto ideal debería de seguir los siguientes pasos:

1. Cada vez que el *sink* recibe un paquete de tipo *report* comprobar mediante el campo *function* de este si esta “dormido” o “activado”.
2. Si está activado comprobar a qué tipo de nodo pertenece y guardar su valor, comprobando con anterioridad que este no sea repetido.
3. Una vez obtenido todos los datos referentes al contexto que se presentan en la red, se procede calcular las diferentes funciones disponibles referentes al contexto

como por ejemplo temperatura ideal que se podría realizar mediante una media de los datos obtenidos que hacen alusión a esta.

4. Después de realizar los correspondientes cálculos comprobar si hay algún nodo con *skill* disponible en la red que permita adecuar el entorno a lo deseado. En caso afirmativo enviar los datos obtenidos de las funciones a todos los nodos correspondientes.

Si se pusiera en forma de pseudo-código, quedaría algo como:

```
gestionPaquetes (Paquetes p, Lista listaValores){
si p.tipo es igual a report  entonces
   si p.contexto y p.valor no se encunetra en listaValores
         listaValores.Añadir(p.contexto,p.valor)
   finsi
finsi
}

calculoTemperatura (Lista listaValores){
var aux
var cont
mientras i menor a listaValores.tamañoLista entonces
   si listaValores.posicion(i).contexto igual a temperatura entonces
         aux igual a aux + listaValores.posicion(i).valor
         cont igual a cont más 1
   finsi
finmientras
aux= aux dividido entre cont
mientras i menor a listaValores.tamañoLista
   si listaValores.posicion(i).contexto igual a skill temperatura
          enviarPaqueteData(aux)
   finsi
}


recibirPaquete(Paquete p, Lista listaValores, numero tamañoRed){
mientras listaValores.tamaño menor que tamañoRed entonces
         gestionPaquetes(p)
finmientras
calcularTemperatura(listaValores)
}
```

5.2.4 Incorporar la capacidad de creación de distintos informes

Teniendo en cuenta que uno de los objetivos principales de este proyecto es la evaluación de redes SDN y el simulador SDN-WISE no aporta ningún dato de salida, se propone la creación de distintos informes en los cuales se recojan distintos tipos datos. Estos informes contendrán información útil como la cantidad de bytes que han circulado por la red para la ejecución de un determinado tipo de *goal*, el tiempo que tarda en llegar los paquetes *Reports*, el tiempo que tarda el controlador en calcular las distintas funciones, etc.

5.2.5 Generador de escenarios aleatorios

Teniendo en cuenta que la creación de un escenario toma bastante tiempo debido a que se deben de registrar los nodos así como posteriormente seleccionarlos, indicar la cantidad que se desean de cada uno, modificar el entorno para adaptarlo a las necesidades que se desean, etc. Se propone la creación de un pequeño programa en java mediante el cual el usuario indicando la cantidad de nodos que quiere de cada tipo y el nombre del fichero, este genere de forma aleatoria un fichero *.csc* en el cual ya se tenga todo lo necesario para iniciar la simulación.

5.2.6 Generador de archivo de configuración de nodos

Debido al cambio que se plantea en el diseño anteriormente explicado se deberá indicar de alguna manera el perfil virtual que se desea en cada nodo, por lo que en este requisito extra, se propone la implementación de un script en java el cual indicándole la cantidad de nodos con *skill* y *goals* que tiene el escenario, este genere un archivo que contenga los distintos perfiles virtuales de los nodos.

5.2.7 Generador de escenarios en el tiempo

Una de las limitaciones que presenta el simulador es la carencia de movimiento en los nodos. Para ello, se ha propuesto e implementado un modelo de movilidad basado en el movimiento de peatones para tener que evitar que mover los nodos de forma manual, se procede a proponer otro programa java el cual indicándole el archivo *.csc* en el que se quiere simular el movimiento de los nodos, así como también indicando el tiempo que se quiere recoger y los distintos instantes en los que se quiere observar la simulación, este crea variaciones del escenario original, cambiando de sitio los nodos *goals*.

5.3 Implementación de los requisitos

Una vez analizado los requisitos y realizar su correspondiente diseño, se pasa a la etapa de implementación de los mismos. Esta parte del proyecto tiene como finalidad llevar las distintas propuestas de diseño de solución de los requisitos a código java, debido a que es con esta tecnología con lo que está hecho SDN-WISE.

Las implementaciones a realizar según los diseños explicados anteriormente son por tanto son:

- Introducción de un perfil virtual en los nodos, en el cual se deben de realizar las siguientes implementaciones y modificaciones:
    - Clase AbstractMote.
    - Clase mote.
    - Clase moteSkill.
- Añadir capacidad de envío de datos relacionados al contexto, en este caso, la implementación consistirá principalmente en la modificación de:
    - Clase Beacon.
    - Clase Report.
- Incorporar la capacidad de recibir y calcular el contexto ideal, para esta caso, ha sido necesario la modificación de los siguientes componentes de SDN-WISE:
    - Clase Controller.
    - Clase SdnWise.
    - Paquete sdn-wise-api-3.6.0
- Incorporar la capacidad de creación de distintos informes, esta implementación se hará en conjunto a las 3 anteriores.
- Generador de escenarios aleatorios, se ha necesitado la creación del siguiente programa java:
    - Generador.
- Generador de archivo de configuración de nodos, como en el caso anterior se ha necesitado la creación del siguiente programa java:
    - GeneradorInfNodos.
- Generador de escenarios en el tiempo, para este caso, se ha creado el siguiente programa:

- GeneradorTiempo.

5.3.1 Introducción de un perfil virtual en los nodos

Como se ha propuesto en la etapa de diseño, la implementación de este requisito pasa por:

- Implementar la clase *moteSkill*: En ella, se introducen los siguientes atributos:
  - *NodeType:* para identificar el tipo de nodo que es.
  - *Function_my:* el cual se pondrá a 1 cuando llegue el *beacon* de activación indicando que ya se puede enviar su perfil virtual al *sink*.
  - *skillType* indicando el tipo de *skill* que tiene el nodo.
  - *context* el cual guarda el valor del contexto.

- Modificar la clase *mote*: Solo se diferencia de la clase *moteSkill* en que en vez de tener los atributos *skillType* y *context*, tiene un atributo llamado *ArrayContext*, el cual es un array de contextos que guarda las preferencias.

- Modificación de la clase *AbstractMote*: En este caso, se ha modificado sobre todo el recibimiento de los paquetes, haciendo que compruebe cada vez que se recibe un *beacon* que este es el de activación y comprobando si se ha activado la variable *function_my* a la hora de enviar paquetes de tipo *reports*, en caso afirmativo este enviara los datos correspondientes al contexto según el tipo de nodo que se trate. Por ultimo mencionar que también se ha agregado la capacidad de generar diversos informes como *BeaconX.txt* o Data.txt, los cuales se explicaran con mayor detenimiento en apartados siguientes de este documento.

5.3.2 Añadir capacidad de envío de datos relacionados al contexto

Como se especificó en la etapa de diseño la implementación de este, pasa por la modificación de las clases *Beacon.java* y *Report.java* localizadas en la librería *sdn-wise-api-3.6.0* de contiki.

5.3.3 Incorporar la capacidad de recibir y calcular el contexto ideal

Para implementar esta capacidad ha sido necesario la modificación de la clase *Controller.java* creando diversos métodos como *calcTempMedia, calcVolMedia, calcLumMedia* o *ExpulsarBebida* las cuales calculan el contexto ideal según los datos recibidos de los perfiles virtuales. Así como también agregarle la capacidad de generar

diversos informes como *outputBeacon.txt*, *data.txt*, *neighbours.txt*, *outputRTT.txt*, *outputRTTAcciones.txt* y *reports.txt* los cuales, se explicaran con mayor detenimiento en apartados siguientes de este documento.

También ha sido necesario la implementación de métodos como *getNetworkGraph* y *getPath* que consigue la cantidad de nodos que forman la red, así como la generación de un informe llamado *neighbours.txt* respectivamente.

Otros métodos como *managePacket* se han tenido que modificar, en concreto este método el cual va recogiendo los distintos paquetes que le llegan, se le ha añadido la capacidad de recolección de los datos relacionados a los perfiles virtuales que llegan junto a los pauqetes *reports* una vez que se ha activado la red. También ha sido necesario la modificación de la clase *SdnWise.java*, en el que se ha modificado simplemente el proceso *startExample* que permite la creación del paquete *beacon* de activación de la red una vez que ya se sabe la cantidad de nodos que la forman.

Por último se ha tenido que modificar una de sus librerías, la cual es la correspondiente con las estructuras de los paquetes llamada sdn-wise-api-3.6.0.

5.3.4 Generador de escenarios aleatorios

La implementación del generador funciona de una manera bastante simple, el usuario simplemente debe de tener un fichero llamado InformacionGenerador.txt con un aspecto como el mostrado en la ilustración 15:

```
1 NodosSkills=4
2 NodosGoals=16
3 NombreEscenario=Prueba
```

Ilustración 15: Generador de escenarios

Una vez que se ejecuta el programa este leerá la información del fichero y creara una simulación como se ha indicado.

5.3.5 Generador de archivo de configuración de nodos

Similar a la implementación anterior, esta vez cuando se ejecuta el programa este le pedirá al usuario la cantidad de nodos con *goals* y *skills* que desea y posteriormente crea un fichero de configuración de los nodos como se puede apreciar en las ilustraciones 16 y

17. Cabe mencionar también que como se ve en la Figura 17 la estructura que sigue este archivo es la siguiente:

- Para nodos *skills*:
    - Id del nodo
    - Skill del nodo
- Para nodos *goals*:
    - ID del nodo
    - Goal de temperatura
    - Goal de volumen
    - Goal de luminosidad
    - Goal de bebida preferida

```
Introduce numero de nodos skills:
4
Introduce numero de nodos goals:
64
```

Ilustración 17: Generador de archivo de configuración

```
0.2
goal=1
0.3
goal=2
0.4
goal=3
0.5
goal=4
0.6
tem=1
vol=2
lum=1
exp=2
0.7
tem=1
vol=1
lum=1
exp=1
```

Ilustración 16: Archivo de configuración

Destacar, que la interacción del simulador con este fichero de configuración consiste en que una vez que iniciamos la simulación, cada nodo se va a la carpeta donde se encuentra el archivo *InformacionNodos.txt*, el cual, es */user/home* y empieza a recorrer el fichero, deteniéndose en su ID y leyendo sus datos.

En relación a los nodos *goals*, indicar que el valor 1 indica que se quiere poner un valor a ese contexto, mientras que el valor 2 indica que no se quiere valor. En caso de querer un valor, este se le otorga de manera aleatoria.

5.3.6 Generador de escenarios en el tiempo

Similar también a las implementaciones anteriores, esta vez cuando se ejecuta el programa llamado GeneradorTiempo.java este pedirá al usuario el nombre del escenario del que desea realizar el movimiento, así como el tiempo total (en minutos) y los lapsos de tiempo (en minutos) en los que quiere ver la simulación, como muestra la ilustración 18 creando diferentes escenarios en base al original, estos nuevos escenarios terminan con un numero indicando el orden de los lapsos de tiempo, así por ejemplo si se tiene un escenario llamado *escenarioPrueba.csc* y un tiempo total de 30 min con lapsos de 5 min se generaran los escenarios *escenariosPrueba.csc2*, *escenariosPrueba.csc3*, *escenarioPrueba.csc4*, etc. Un ejemplo de esto es el que se muestra en la ilustración 19:

```
Introduce nombre del escenario:
escenarioPrueba.csc
```

Ilustración 18: Generador tiempo 1

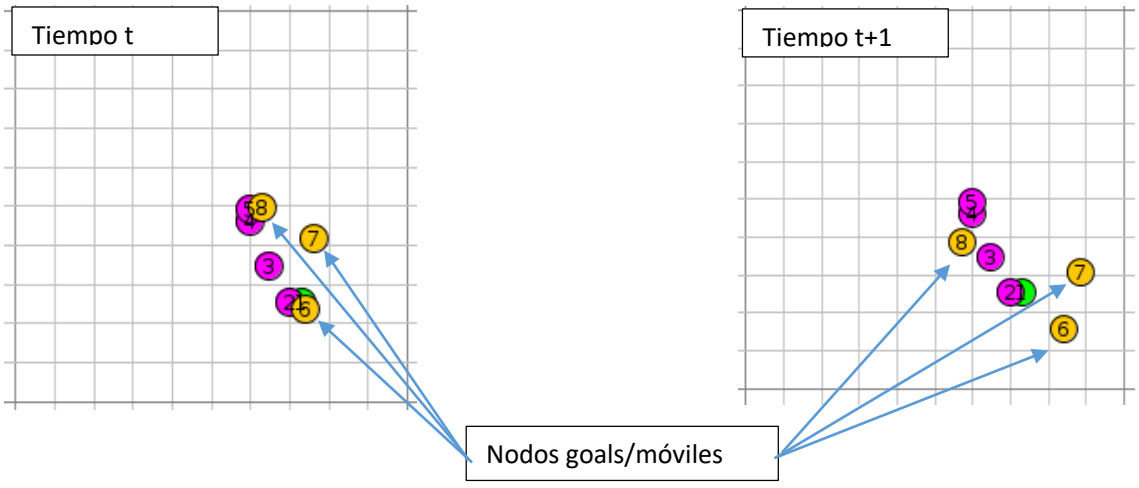

Ilustración 19: Generador tiempo 2

6. Resultados experimentales

Mientras que en el apartado anterior tiene como objetivo el desarrollo de las implementaciones realizadas sobre el simulador SDN-WISE. En este punto se llevarán a cabo el planteamiento y simulación de diversos escenarios para posteriormente analizar los resultados obtenidos.

Por tanto, las principales tareas a desarrollar en esta fase del documento son las siguientes:

1. Planteamiento de los escenarios: Mediante la ayuda de un generador de escenarios, el cual se comentó anteriormente, se ha generado diversos escenarios intentando representar situaciones que se pueden dar de manera cotidiana en diferentes lugares y ver como las redes SDN, pueden adaptar el entorno de una manera fácil sin la necesidad de la intervención de una persona.

2. Simulación de escenarios: Tras la creación de los escenarios, se lleva a cabo la ejecución y el posterior estudio de los resultados.

6.1 Planteamiento de los escenarios

Los distintos escenarios que se han generado tienen como objetivo intentar reflejar diferentes situaciones que se pueden llegar a dar en la vida cotidiana, así como ver la forma en que esta tecnología de redes contextuales puede ayudar en el día a día. Dicho esto, cabe decir que los escenarios están compuestos por distintos tipos de nodos que tienen diferentes características y funcionalidades:

- **Nodos Goal:** Tiene como función principal la recopilación de los datos preferentes a su usuario, en la vida real, sería su Smartphone.

- **Nodos Skill:** Tiene como función principal la de adaptar el entorno en base a los datos que le llegan del *Sink*. Este nodo en la vida real, se podría decir que son los distintos aparatos electrónicos que se usan en la vida cotidiana, como por ejemplo un aire acondicionado, una máquina expendedora, una lámpara, etc.

- **Sink:** Este nodo llamado *Sink* o controlador, es el más importante de todos ya que mapea la red para saber qué elementos se encuentran en ella, recopila los datos de los nodos *goals* y envía los resultados obtenidos a sus respectivos nodos *skills*.

6.2 Simulación de escenarios y análisis de resultados

La simulación de los escenarios y su posterior análisis permite conocer en mayor medida los resultados obtenidos en cada escenario. Se puede decir que los escenarios se han agrupados en 4 tipos diferentes según la cantidad de nodos de tipo *skill* que poseen, siendo estos de 1, 2, 3 y 4 nodos con *skills*. Además de esto, los escenarios se han probado con distintos nodos *goals* empezando desde 1 único nodo con *goal* a llegar a escenarios con un máximo de 64.

Para un estudio más riguroso de estas redes SDN, como se comentó con anterioridad en el apartado del desarrollo, se ha implementado la capacidad de sacar diversos informes *.txt* con un estilo *.csv*, para una fácil exportación de estos.

Una vez visto esto, se pasara a explicar cómo funciona una simulación así como ver los diversos informes que se generan.

6.2.1 Ejecución de la simulación

La nueva ejecución de una simulación en el entorno de SDN-WISE se podría dividir en 4 etapas concretas:

1. Una vez empezada la ejecución de la simulación. Se espera un tiempo a que la red "despierte", es decir, a que el controlador mapee la red que se ha creado antes de enviar el *beacon* de activación de los nodos. En este instante los *beacon* y los *reports* toman valor 0 en los campos *function* e *information* respectivamente. Esto se puede ver en la ilustración 20.

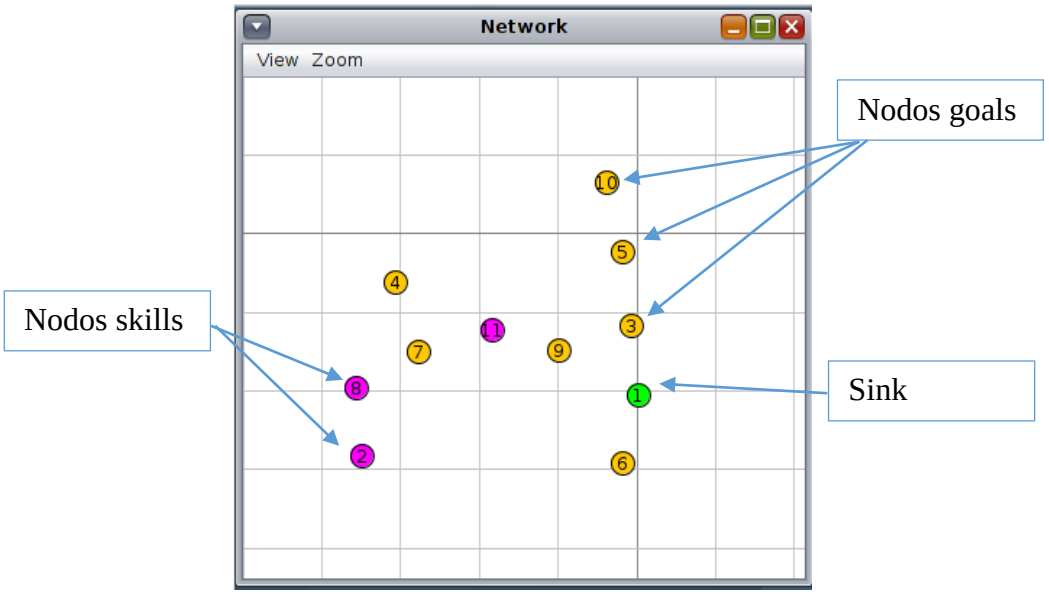

Ilustración 20: Ejecución de simulación 1

2. Una vez que la red se ha despertado, y se sabe la cantidad de nodos que la forman, se han formado la lista de vecinos de cada nodo, etc. Se procede a enviar un *beacon* con valor en el campo *function* de 1 como muestra la ilustración 21, lo que le indicara a los distintos nodos de la red que ya pueden enviar los diversos datos sobre *goals* y *skills* que tienen.

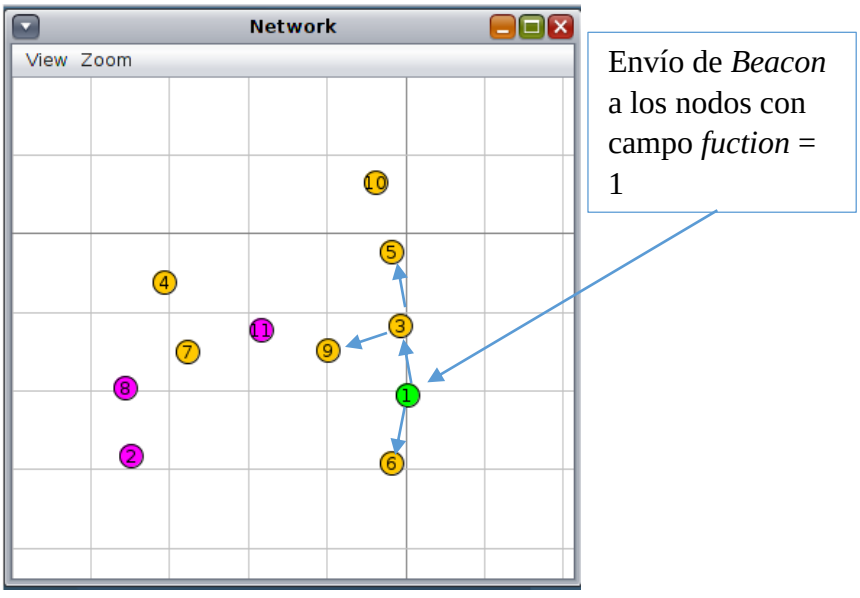

Ilustración 21: Ejecución simulación 2

3. Cuando un nodo recibe el paquete *beacon* creado anteriormente, este se "activa" y a partir de este momento, se procede a crear paquetes *report* indicando en el tipo de contexto que tiene y su valor o en caso de un nodo con *skill* este indicara, habilidad tiene. Como se puede apreciar en la ilustración 22.

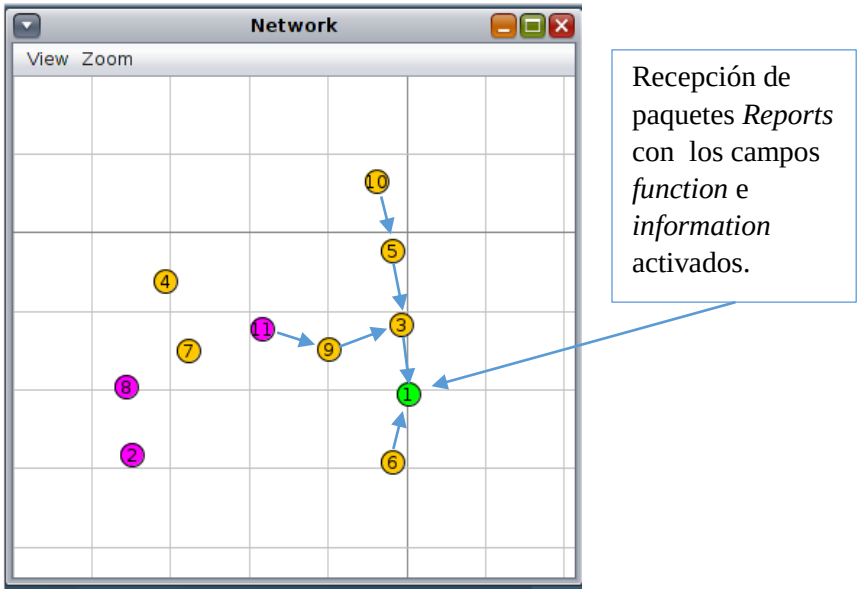

Ilustración 22: Ejecución simulación 3

4. Una vez que el *sink* ha recibido todos los *reports*, procede a calcular las diversas funciones y los resultados obtenidos se envían a sus respectivos nodos con *skills* mediante un paquete de tipo data como los visto en la parte de desarrollo, terminando así la simulación, como se muestra en la ilustración 23.

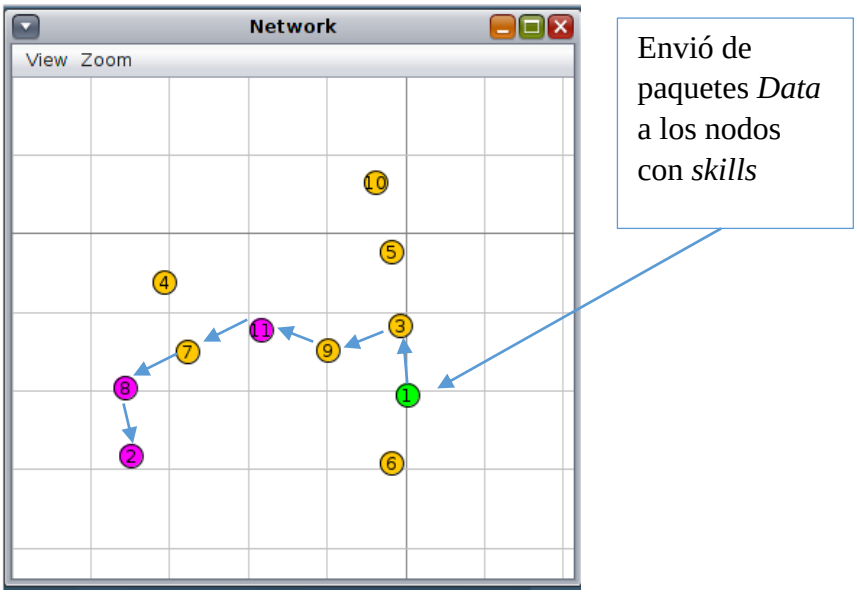

Ilustración 23: Ejecución simulación 4

Tras esta breve explicación sobre las etapas que ocurren en una simulación se procederá a indicar los distintos informes que se generan así como la estructura de su contenido.

6.2.2 Informes de la ejecución

Los informes que se generan en cada simulación se pueden dividir en 2 dependiendo de si se generan en la parte de los nodos *goals*/*skills* o si en cambio es generado por parte del *sink* o controlador.

6.2.2.1 Informes nodos goals/skills

Los informes generados por estos nodos son los siguientes:

- BeaconX.txt: estos informes en los cuales la X hace referencia al id del nodo en cuestión que lo ha generado para saber a qué nodo pertenece, trata simplemente de mostrar los distintos paquetes de tipo *beacon* que han pasado por el nodo en cuestión. Este informe tiene los siguientes campos:
    - packetLength: contiene la longitud del paquete contando la cabecera de este.
    - netId: indica la red a la que pertenece el paquete.
    - srcNode: indica el nodo de origen del paquete.

  - dstNode: indica el nodo destino del paquete, en este caso al tratarse de paquetes de tipo *beacon* siempre será 255.255 la cual es la dirección de broadcast haciendo alusión a todos los paquetes referentes a la red.
  - Type: hace referencia al tipo de paquete que es, al ser de tipo beacon los paquetes que se recogen en este informe siempre tendrá un valor de 1.
  - Ttl: indica el tiempo de vida del paquete antes de que se deseche de la red si no consigue llegar a su destino.
  - nextHop: indica que nodo es el siguiente salto del paquete.
  - Function: campo que indica si se puede enviar los datos referentes al contexto o no.
  - Distance: hace referencia a la distancia a la que se encuentra con respecto al nodo origen en número de saltos.
  - Battery: se refiere a la cantidad de batería del nodo en cuestión que ha enviado el *beacon*.
  - Tiempo_recibido: el tiempo en el que se ha recibido dicho paquete.

- DataX.txt: estos informes en los cuales la X hace referencia al id del tipo de *skills* del nodo en cuestión que lo ha generado para saber a cual pertenece, trata simplemente de mostrar los distintos paquetes de tipo data que se han enviado a dicho nodo con *skill*. Estos informes, tienen la siguiente esrtuctura:
  - packetLength: contiene la longitud del paquete contando la cabecera de este.
  - netId: indica la red a la que pertenece el paquete.
  - srcNode: indica el nodo de origen del paquete.
  - dstNode: indica el nodo destino del paquete.
  - Type: hace referencia al tipo de paquete que es, al ser de tipo data los paquetes que se recogen en este informe siempre tendrá un valor de 0.
  - payload: indica la información que transporta, en este caso llevara los distintos resultados obtenido de los algoritmos del sink.
  - Tiempo_recibido: el tiempo en el que se ha recibido dicho paquete.

6.2.2.2 Informes nodos sink

Los informes generados por estos nodos son los siguientes:

- Data.txt: este informe contiene los paquetes de tipo data que genera el controlador y que posteriormente enviara a los nodos con *skills*. Este informe se compone de los siguientes apartados:
    - packetLenght: como en casos anteriores, este campo contiene la longitud del paquete contando la cabecera.
    - netId: indica la red a la que pertenece el paquete.
    - srcNode: indica el nodo de origen del paquete.
    - dstNode: indica el nodo destino del paquete.
    - Type: hace referencia al tipo de paquete que es, al ser de tipo data los paquetes que se recogen en este informe siempre tendrá un valor de 0.
    - payload: indica la información que transporta, en este caso llevara los distintos resultados obtenido de los algoritmos ejecutados anteriormente.
    - Tiempo_envio: el tiempo en el que se ha enviado dicho paquete.

- Neighbours.txt: este tipo de informe guarda la cantidad de nodos vecinos que tiene cada mote. Contiene los siguientes campos:
    - nodeId: que recoge el id del nodo en cuestión.
    - Neighbours: que indica la cantidad de vecinos que tiene cada nodo perteneciente a la red.

- outputBeacon.txt: este archivo, guarda los distintos tipos de paquetes *beacon* que se generan en el *sink* para su posterior envío. Su estructura es similar a BeaconX.txt siendo la siguiente:
    - packetLength: contiene la longitud del paquete contando la cabecera de este.
    - netId: indica la red a la que pertenece el paquete.
    - srcNode: indica el nodo de origen del paquete.
    - dstNode: indica el nodo destino del paquete, en este caso al tratarse de paquetes de tipo beacon siempre será 255.255 la cual es la dirección de broadcast haciendo alusión a todos los paquetes referentes a la red.
    - Type: hace referencia al tipo de paquete que es, al ser de tipo beacon los paquetes que se recogen en este informe siempre tendrá un valor de 1.
    - Ttl: indica el tiempo de vida del paquete antes de que se deseche de la red si no consigue llegar a su destino.

  - nextHop: indica que nodo es el siguiente salto del paquete.
  - Function: campo que indica si se puede enviar los datos referentes al contexto o no.
  - Distance: hace referencia a la distancia a la que se encuentra con respecto al nodo origen en número de saltos.
  - Battery: se refiere a la cantidad de batería del nodo en cuestión que ha enviado el beacon.

- Reports.txt: este informe guarda los diversos tipos de paquetes *reports* que llegan al controlador por parte de los demás nodos. Su contenido se muestra de la siguiente forma:
  - packetLength: contiene la longitud del paquete contando la cabecera de este.
  - netId: indica la red a la que pertenece el paquete.
  - srcNode: indica el nodo de origen del paquete.
  - dstNode: indica el nodo destino del paquete, en este informe al tratarse siempre de los *reports* recibidos por el *sink*, siempre tendrán un valor de 0.1
  - type: hace referencia al tipo de paquete que es, al ser de tipo *report* los paquetes que se recogen en este informe siempre tendrá un valor de 2.
  - ttl: indica el tiempo de vida del paquete antes de que se deseche de la red si no consigue llegar a su destino.
  - nextHop: este campo contiene el siguiente nodo al que debe ir para poder llegar al nodo destino, al tratarse de los paquetes que tienen como destino el *sink*, este campo siempre tendrá un valor de 0.1.
  - function: indica el contexto al que corresponde el campo *context* si el nodo es de tipo *goal*, en cambio sí es de tipo *skill*, este campo indicara que el nodo origen es de tipo *skill*.
  - distance: hace referencia a la distancia a la que se encuentra con respecto al nodo origen en número de saltos.
  - battery: se refiere a la cantidad de batería del nodo en cuestión que ha enviado el *report*.
  - context: contiene el valor del contexto de los nodos *goal*, o el tipo de *skill* que tiene si el *report* corresponde a un nodo *skill*.

  - neighboursSize: indica la cantidad de vecinos que tiene el nodo.

- outputRTT.txt: este fichero guarda los distintos tiempos en los que se envía el *beacon* de "activación" y se reciben los *reports* correspondientes. Este fichero muestra los datos de la siguiente manera:
  - nodo: este campo hace alusión al nodo al cual pertenece el *report*.
  - Tiempo_envio_beacon: instante del tiempo en el que se envió el *beacon* de activación por parte del *sink*. Este campo siempre tendrá el mismo valor.
  - Tiempo_recibido_report: instante del tiempo en el que se ha recibido el *report*.
  - Diferencia_de_tiempo: el tiempo que ha tardado en llegar el *report* en segundos.
  - Distancia: la distancia en saltos en la que se encontraba el nodo origen con respecto al *sink*.

- outputRTTAcciones.txt: este archivo contiene información relacionada con el tiempo en que el controlador tarda en calcular las distintas acciones relacionadas con los diversos contextos que se tienen. Su estructura es la siguiente:
  - tipo_calculo: indica el cálculo que se va a realizar.
  - Tiempo_inicio: indica el instante en el tiempo en que se inició el cálculo.
  - Tiempo_fin: indica el instante en el tiempo en que se envió el paquete data correspondiente.
  - Diferencia_tiempo: muestra el tiempo que se ha tardado en realizar el cálculo.

6.2.3 Análisis de resultados

Como se comentó anteriormente se han realizado diversas simulaciones con una gran variedad de nodos *goals* y *skills*. Una vez terminada las ejecuciones se han agrupado los diversos datos obtenidos sintetizándolos y creando varias gráficas de las cuales se pueden sacar varias conclusiones sobre estas redes.

6.2.3.1 Análisis del Overhead de la red

El objetivo de esta gráfica es el de analizar la cantidad de bytes que han sido necesarios para realizar la simulación y la cantidad de nodos presentes para observar así la cantidad de información que es necesaria enviar en las distintas simulaciones. Debido a los cambios que puede presentar estas redes móviles en relación a la cantidad de nodos presentes en la simulación y la cantidad de nodos reales en la red, se han realizado tres ejecuciones independientes, con el objetivo de calcular intervalos de confianza al 95%. Los resultados obtenidos son los siguientes:

- Tabla obtenida de la simulación 1 (cada celda hace referencia al número de bytes totales enviados en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	243	572	655	347	2537	2114	4015	21320
2	293	902	825	601	1129	1413	9977	61368
3	1150	1140	727	2263	1058	12604	9229	72892
4	3869	3644	4409	5515	7744	19409	18344	35700

Tabla 9: Tabla overhead simulación 1

- Tabla obtenida de la simulación 2 (cada celda hace referencia al número de bytes totales enviados en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	1419	1535	5247	4430	8436	21410	21062	17434
2	2185	3438	3457	6451	9941	15024	35367	127104
3	2708	4792	7863	6945	11204	16685	51182	127448
4	3719	5864	6363	4743	14004	9993	32930	103241

Tabla 10: Tabla overhead simulación 2

- Tabla obtenida de la simulación 3 (cada celda hace referencia al número de bytes totales enviados en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	1105	1558	1417	2655	11411	17364	30404	113380
2	2164	3625	4747	7698	15990	19277	18047	94545
3	3136	3983	8806	6908	8301	13837	28748	25140
4	2577	7603	9924	3666	7497	12513	50432	146398

Tabla 11: Tabla overhead simulación 3

Como se puede apreciar en las tablas de datos anteriores las variaciones de un escenario a otro pueden ser bastantes significativas, de esta manera, teniendo por ejemplo una simulación de tipo 4 nodos *goals* y 1 nodo de tipo *skill* que se pude dar de manera bastante habitual en varios escenarios de la vida cotidiana como por ejemplo un domicilio particular haciendo referencia a los 4 nodos *goals* a los componentes de una familia y al nodo de tipo *skill* un electrodoméstico como un aire acondicionado se han producido los siguientes casos:

- Caso simulación 1:

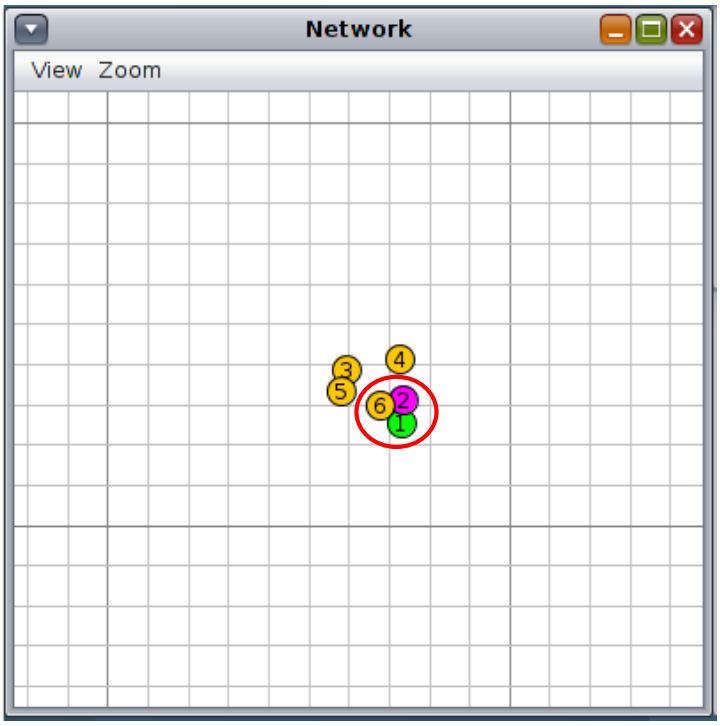

Ilustración 24: Simulación overhead 1

Como se pude ver en la ilustración 24, aunque la simulación contiene 4 nodos *goals* y 1 nodo *skill*, en realidad, la red únicamente está formada por un nodo *goal* y un nodo *skill*, dando por tanto una red de dos nodos y ambos a un único salto de distancia del controlador por lo que se puede llegar a comprender el bajo nivel de bytes enviados en esta simulación, más concretamente 347.

- Caso simulación 2:

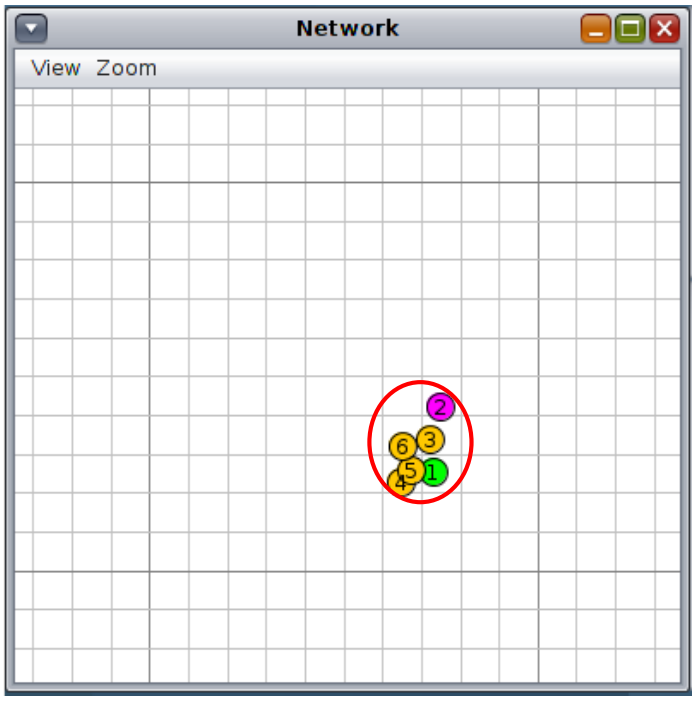

Ilustración 25: Simulación overhead 2

Al contrario del caso anterior, en este, como se ve en la ilustración 25 se ha dado la casualidad de que en la red de la simulación han participado todos los nodos de está formando una red de 4 nodos *goals* y 1 nodo *skill*, dando razones suficientes para entender el alto contenido de bytes enviados en esta red con respecto al caso anterior dando un total de 3340 bytes enviados.

- Caso simulación 3:

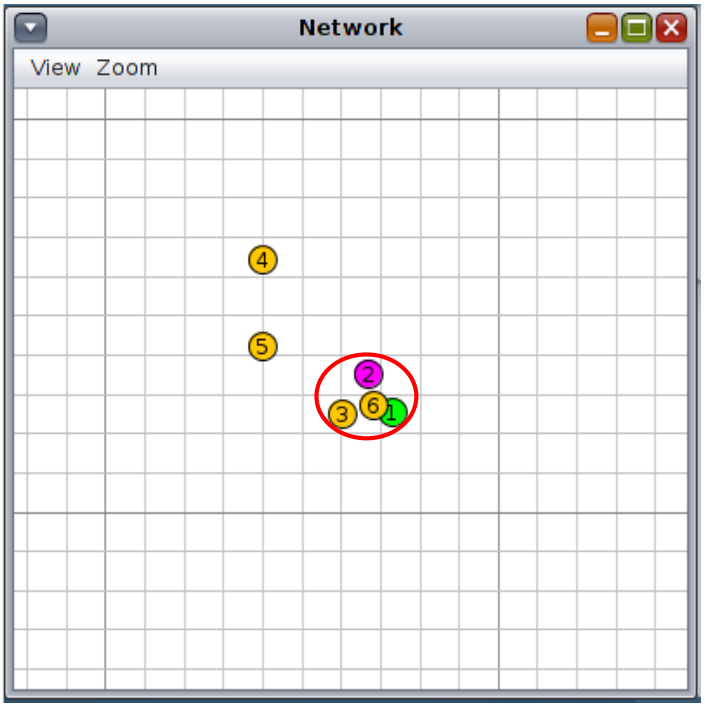

Ilustración 26: Simulación overhead 3

En esta simulación, como se puede aprecias en la ilustración 26, a diferencia de los 2 casos anteriores, en esta ocasión ha salido una red de un tamaño comprendido entre las 2 mencionadas, dando una red de un total de 3 nodos 2 nodos *goals* y 1 nodo *skill*, aparte de que como se puede ver hay 2 nodos que, se encuentra a más de 1 salto de distancia del *sink* a lo que acarrea la necesidad de enviar más bytes de control y como se puede apreciar en su tabla, se da también un resultado acorde a lo dicho con anterioridad y comprendido también entre los casos anteriores con un total de 2655 Bytes enviados.

Por último indicar que los datos necesarios para la realización de esta tabla se pueden encontrar en los informes *BeaconX.txt*, *data.txt* y *reports.txt*, en ellas, como se describió anteriormente, se muestra los distintos paquetes que se han enviado por la red y han sido recibidos por los diversos nodos.

Una vez obtenido los distintos datos se procede a calcular el intervalo de confianza y a introducirlos en la siguiente grafica perteneciente a simulación 1 para tener una visión más global sobre los posibles resultados que se pueden obtener dando como resultado:

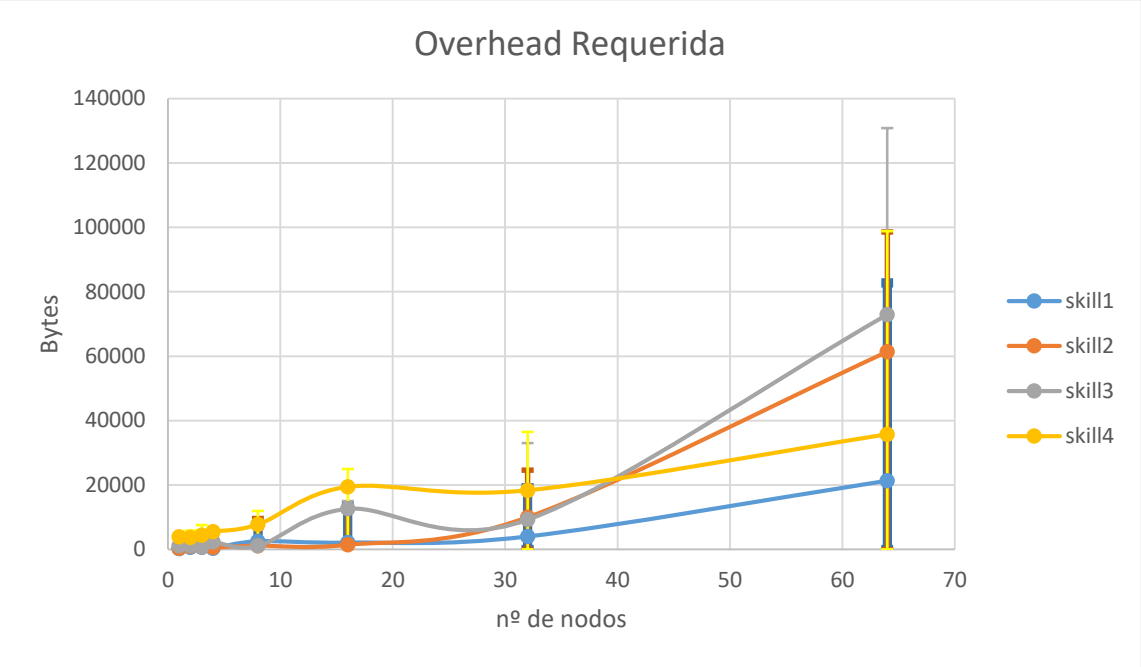

Ilustración 27: Overhead requerida

En la ilustración 27, se puede observar por tanto que la tendencia que sigue la gráfica es a mayor número de nodos en la red mayor número de bytes enviados en esta. Esto se puede traducir como que a una mayor cantidad de nodos en la red más recursos se precisan para el correcto funcionamiento de esta.

6.2.3.2 Eficiencia

La gráfica de eficiencia pretende mostrar, el tanto por ciento de bytes del total que se han enviado en la red, y que son útiles para el cálculo de las diferentes funciones en relación al contexto, que realiza el controlador con la cantidad de nodos que presenta dicha red. La fórmula utilizada para este cálculo es:

$$\frac{Cantidad\ de\ bytes\ referentes\ al\ perfil\ virtual}{Total\ de\ bytes\ enviados\ por\ la\ red}$$

Como en el caso anterior, estas redes móviles pueden variar mucho dependiendo de lo cerca o alejando que pueden estar los distintos nodos entre ocasionando en algunas situaciones dejar incluso a varios nodos fuera de la red debido a la distancia en la que se encuentran. Por ello, también ha sido necesario la creación de un intervalo de confianza, para poder observarlas y sacar conclusiones de una manera más determinada.

Después de realizar las 3 simulaciones, se han obtenido los siguientes datos:

- Tabla de la simulación 1 (cada celda hace referencia a la eficiencia calculada en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	0,032921811	0,017482517	0,024427481	0,023054755	0,029168309	0,018921476	0,018929016	0,015196998
2	0,006825939	0,028824834	0,00969697	0,016638935	0,015943313	0,01556971	0,007817981	0,018837179
3	0,006086957	0,007894737	0,006877579	0,008837826	0,005671078	0,034274833	0,016036407	0,009740438
4	0,004135436	0,009879254	0,023134498	0,013417951	0,015495868	0,012159308	0,008667684	0,01070028

Tabla 12: Tabla eficiencia simulación 1

- Tabla de la simulación 2 (cada celda hace referencia a la eficiencia calculada en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	0,007047216	0,011726384	0,011435106	0,007223476	0,009957326	0,007239608	0,00797645	0,009980498
2	0,008237986	0,005817336	0,01099219	0,00589056	0,014485464	0,008120341	0,01046173	0,006042296
3	0,005169867	0,010434057	0,007121964	0,010079194	0,008211353	0,008270902	0,006408503	0,007093089
4	0,004302232	0,004092769	0,007543612	0,004638415	0,011568123	0,012608826	0,006559368	0,003002683

Tabla 13: : Tabla eficiencia simulación 2

- Tabla de la simulación 3 (cada celda hace referencia a la eficiencia calculada en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	0,009049774	0,011553273	0,007057163	0,006779661	0,019104373	0,006680488	0,013156164	0,009137414
2	0,005102041	0,006025609	0,002949231	0,017926734	0,015634772	0,018675105	0,007203413	0,007975038
3	0,005169867	0,010434057	0,006359 3272	0,004053272	0,008191784	0,011270412	0,013635731	0,007093089
4	0,004302232	0,005787189	0,004232164	0,023458811	0,019207683	0,017901383	0,013205901	0,006817033

Tabla 14: Tabla eficiencia simulación 3

Los datos necesarios que se necesitan para la realización de esta tablas son los mismos que los del apartado anterior, teniendo en cuenta que cuando el *sink* envía el *beacon* de

activación los *reports* que se envían a partir de ese instante, llevan 2 bytes de información útil. Si seguimos el ejemplo anterior, se puede ver como se han procesado los datos.

Para ello, se tomara como referencia la simulación 3 debido a que este tiene una complejidad intermedia comparada con las otras 2, al estar formada la red por 3 nodos. Una vez calculado la cantidad de bytes totales que se han transmitido en la red, nos dirigimos al informe reports.txt, en él, como se puede observar en la ilustración 28, es a partir del *report* número 23 en el que se empieza a recibir los *reports* con información del contexto, teniendo en cuenta que son 9 *reports* con información útil los que se envía por parte del nodo y los recibe el *sink*, por los que podemos decir que hay un total de 18 bytes de datos útiles, ahora siguiendo la fórmula, cantidad de bytes información útil/cantidad de bytes totales, obtenemos 18/2655, lo que en efecto es 0,006779661.

```
23  24,1,0.6,0.1,2,98,0.1,1,1,213,29,3
24  24,1,0.6,0.1,2,97,0.1,2,1,213,15,3
25  24,1,0.6,0.1,2,96,0.1,3,1,213,25,3
26  24,1,0.6,0.1,2,95,0.1,4,1,213,5,3
27  21,1,0.3,0.1,2,98,0.1,1,2,213,25,2
28  21,1,0.3,0.1,2,97,0.1,2,2,213,255,2
29  21,1,0.3,0.1,2,96,0.1,3,2,213,79,2
30  21,1,0.3,0.1,2,95,0.1,4,2,213,2,2
31  21,1,0.2,0.1,2,96,0.1,11,2,213,1,2
```

Ilustración 28: Datos reports

Una vez calculado los datos referentes a las tablas, se procede a calcular los distintos intervalos de confianza y a introducirlos en la siguiente grafica perteneciente a simulación 1 para tener una visión más global sobre los posibles resultados que se pueden obtener dando como consecuencia:

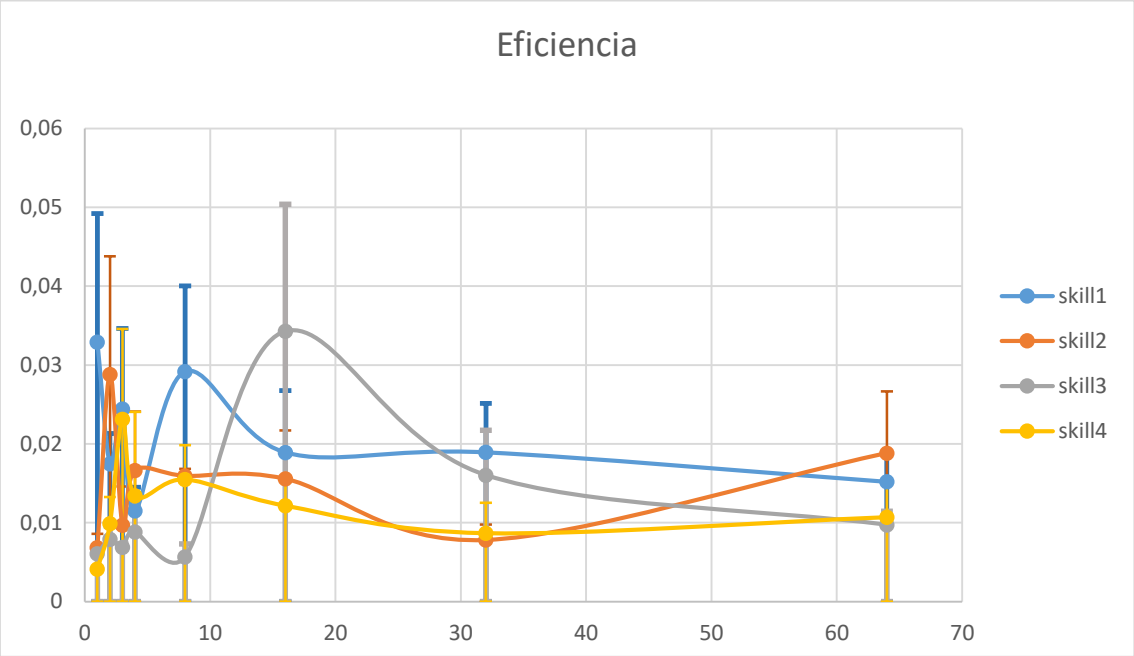

Ilustración 29: Eficiciencia

Como se puede observar en la gráfica que aparece en la ilustración 29, la eficiencia de la red no ronda a más del 0,03 debido a que como se comentó con anterioridad esto se debe a que la mayoría de los datos que se envían en esta red están relacionados principalmente al mapeo de la red y solo 2 bytes por *reports* nos son útiles para el cálculo de los contextos, aparte de que estos solo se envían cuando la red ya se ha formado y el *sink* la conoce. En este sentido se puede decir que estas redes, tiene una baja eficiencia en relación a funcionalidades relacionadas con el cálculo de los diferentes contextos. Una conclusión bastante importante que se puede sacar de esta gráfica es que alcanza su mayor eficiencia entre las simulaciones de 1 nodo a 16 nodos *goals*.

6.2.3.3 Tiempo de cómputo de funciones skills

Esta gráfica llamada tiempo requerido, tiene como misión relaciona el tiempo que tarda en calcular y enviar los paquetes tipo data correspondientes a la salida de las funciones que calculan el contexto ideal según los usuarios y la cantidad de nodos que presenta la red, una vez que ya se han dado a conocer todos los datos en relación al contexto. Como en los dos casos anteriores se han realizado 3 simulaciones de cada tipo para calcular su intervalo de confianza, de esta forma tenemos los siguientes datos obtenidos:

- Datos simulación 1 (cada celda hace referencia al tiempo requerido en calcular los contextos ideales en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	0	0,00461853	0,008122104	0,001806869	0,020345669	0,000552743	0,002960214	0,00035663
2	0	0,008402798	0,008447374	0,00002764	0,004240301	0,00527292	0,015560522	0
3	0	0,00029542	0,00020732	0,000428212	0,000046059	0	0	0,000489423
4	0,000976344	0,006373752	0,00052757	0,00479768	0,008232235	0,01928841	0,03174881	0,00276985

Tabla 15: Tabla tiempo de computo simulación 1

- Datos simulación 2 (cada celda hace referencia al tiempo requerido en calcular los contextos ideales en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	0	0,000312407	0,00005731	0,006644047	0,001834928	0,000311714	0,000658924	0,002997733
2	0	0,000895368	0,00092072	0,000815428	0,01504097	0,000983796	0,000891779	0,000552889
3	0,00057425	0,003462717	0,019015894	0,001258649	0,001334385	0,001420561	0,001408368	0,002117304
4	0,000890311	0,00101358	0,024711188	0,001970291	0,000883289	0,008231962	0,001171993	0,000625276

Tabla 16: Tabla tiempo de computo simulación 2

- Datos simulación 3 (cada celda hace referencia al tiempo requerido en calcular los contextos ideales en la simulación):

skills\nodos	1	2	3	4	8	16	32	64
1	0,000571802	0,000112393	0,001883779	0,001174463	0,006342008	0,000560775	0,000982802	0,002997733
2	0,001203185	0,001654623	0,001877782	0,000815428	0,004044392	0,00212239	0,003429236	0,000552889
3	0,00412088	0,007966121	0,019015894	0,001167818	0,004013216	0,000554316	0,001073475	0,000700303
4	0,002343863	0,005045028	0,001067937	0,000729329	0,00060271	0,00190569	0,026750982	0,017062112

Tabla 17: Tabla tiempo de computo simulación 3

Cabe destacar que aquellos datos que tienen un valor de 0 significan que no se calculó ni envió ninguna función de tipo data debido a diversas razones como por ejemplo que en la red no se encontraba ningún nodo de tipo *skill* debido a que estaba alejado, etc.

La gráfica como resultado de los datos anteriores y los intervalos de confianza es la mostrada en la ilustración 30:

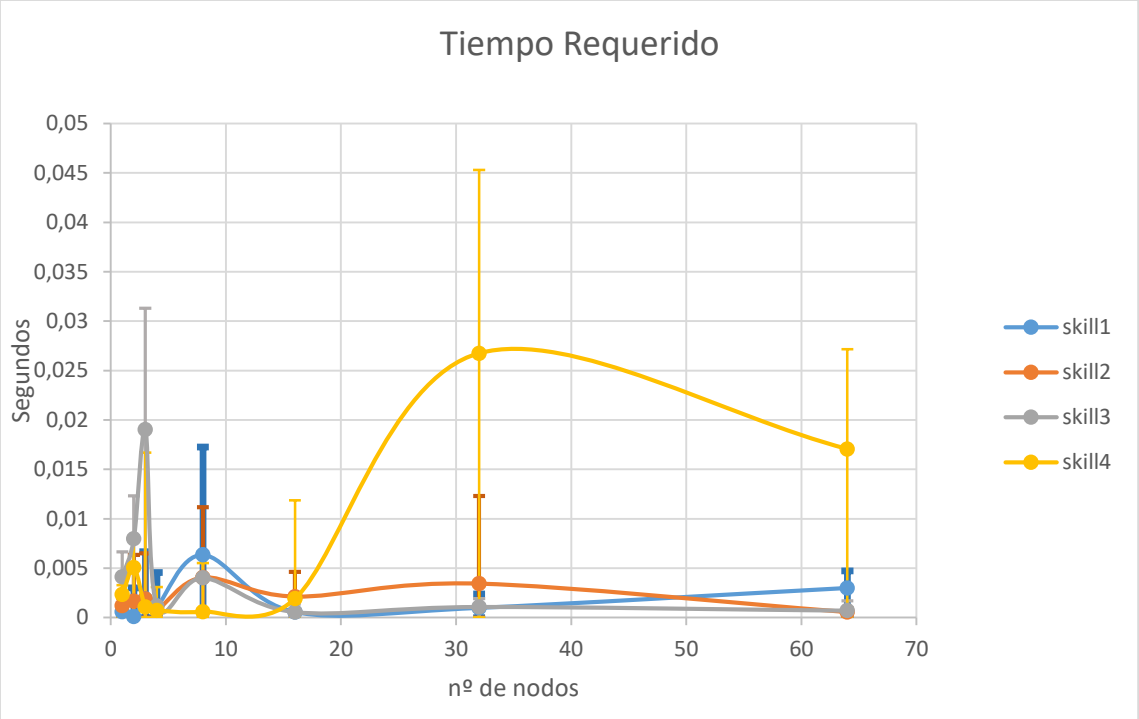

Ilustración 30: Tiempo requerido

Las conclusiones que se pueden sacar de esta gráfica es que de manera general a mayor número de *skills* y *goals* en una red mayor tiempo se necesita emplear para calcular las diversas funciones, aun así gracias a los procesadores de hoy en día el tiempo que se emplea en estos cálculos no llega ni a 0,1 segundos.

6.2.3.4 Función CDF

Esta gráfica correspondiente a la función de distribución acumulada tiene como objetivo mostrar las probabilidades de llegada de los distintos paquetes *reports* según en tiempo en el que tardan en llegar y la cantidad de saltos necesarios a dar para llegar al *sink*. Debido a la inmensa cantidad de datos necesarios para su realización no se mostrarán

todos, pero aquí se dejara una pequeña muestra de observación de los primeros datos obtenidos en los paquetes con los 3 primeros saltos.

Hops = 1	%	Hops=2	%	Hops=3	%
1,000000000	0,003773585	7,350117278	0,005025126	10,380014804	0,010638298
13,622506990	0,00754717	7,383638078	0,010050251	10,388490041	0,0212765957
13,651263869	0,011320755	7,393585385	0,015075377	10,762642750	0,0319148936
13,654127356	0,01509434	7,403944210	0,020100503	10,763827411	0,0425531915
13,672458004	0,018867925	8,189440085	0,025125628	10,764796648	0,0531914894
14,729457198	0,022641509	8,231756623	0,030150754	12,907410949	0,0638297872
14,749843143	0,026415094	8,240008859	0,035175879	20,000000000	0,0744680851
14,763586500	0,030188679	23,805513304	0,040201005	24,115474288	0,0851063830
14,774369395	0,033962264	23,809079874	0,045226131	24,121208297	0,0957446809
14,775210323	0,037735849	23,813877463	0,050251256	28,000000000	0,1063829787
14,780051029	0,041509434	23,822712468	0,055276382	31,166043601	0,1170212766
14,780901461	0,045283019	24,000801322	0,060301508	37,074582300	0,1276595745
14,790545079	0,049056604	24,006896386	0,065326633	37,077055372	0,1382978723
14,946764564	0,052830189	24,009282182	0,070351759	39,401953216	0,1489361702
14,982672792	0,056603774	24,016199732	0,075376884	39,423160170	0,1595744681
14,988616452	0,060377358	24,771776613	0,08040201	39,468917278	0,1702127660
14,992922113	0,064150943	24,775582049	0,085427136	39,469280745	0,1808510638
14,994120042	0,067924528	24,778404663	0,090452261	40,302808256	0,1914893617
15,006989657	0,071698113	24,781945812	0,095477387	40,305677499	0,2021276596
15,008293170	0,075471698	25,011436791	0,100502513	40,396977225	0,2127659574

Tabla 18: Tabla muestra función CDF

Una vez realizado obtenido los datos y haber calculado su tanto por ciento de probabilidad de la llegada del paquete en ese tiempo, se procede a insertar los datos en la gráfica dando como resultado:

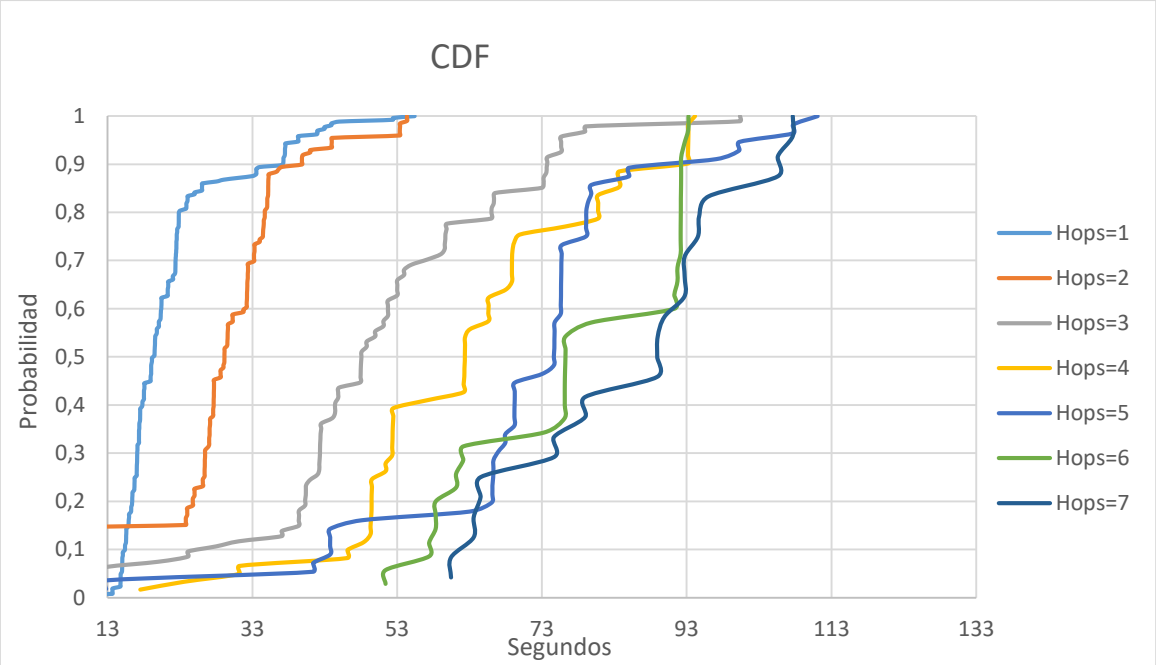

Ilustración 31: Función CDF

Como se puede observar esta gráfica, que se muestra en la ilustración 30, es bastante interesante ya que por ejemplo, nos muestra que para el tiempo aproximadamente 55 segundos, ya se han recibido todos los *reports* referentes a los nodos que se encuentran a un salto de distancia, mientras que para ese mismo tiempo, están empezando a llegar los paquetes que están a 6-7, nodos de distancia. Otras de las observaciones más importantes a realizar en esta gráfica es el aumento de tiempo que se requiere en la simulación cuando se tienen nodos a más de 2 saltos de distancia acarreando que casi se duplique el tiempo de simulación.

6.2.3.5 Tabla Movimiento

Como se ha visto en los sub-apartados anteriores de esta sección, la cantidad de información que se envía por la red, así como la eficiencia y el tiempo requerido a la hora de calcular y enviar los resultados de los contextos ideales calculados, pueden variar de una simulación a otra debido principalmente a la cantidad de nodos que forman la red SDN.

En este punto se muestra por tanto una tabla creada a partir de un escenario que contiene 4 nodos skills y 16 nodos goals, al cual, se le ha aplicado el generador de escenarios en el tiempo, descrito anteriormente. Con la cual se pretende mostrar las variaciones que puede haber en una de este tipo a medida que pasa el tiempo.

Para la generación de los escenarios, se ha utilizado un tiempo total de 30 minutos y un lapso de 5 minutos dando lugar a 6 simulaciones totales en los que se han obtenido los siguientes datos:

Simulación	Overhead	Eficiencia	Tcomp
1	9993	0,012608826	0,008231962
2	13120	0,015396341	0,004016496
3	10506	0,019036741	0,001255947
4	12925	0,017176015	0,001410614
5	11877	0,019196767	0,01143041
6	2485	0,012877264	0,001281548

Tabla 19: Tabla Movimiento

	Overhead	Eficiencia	Tcomp
Máximo	13120	0,019196767	0,01143041
Medio	10151	0,016048659	0,004604496
Mínimo	2485	0,012608826	0,001255947

Tabla 20: Tabla movimento máximo y mínimo

Como se puede apreciar en las tablas anteriores, existe una gran diferencia entre los valores máximos y mínimos que se han obtenido, esto como ya se ha comentado y se pretende resaltar, se debe a que aunque en los escenarios haya la misma cantidad de nodos, en realidad, la red no está formada por todos ellos. Como se puede apreciar en las ilustraciones 32 y 33, los cuales hacen referencia a las simulaciones 2 y 6 que corresponden a aquellas en los que se ha obtenido los valores máximos y mínimos respectivamente:

- Simulación 2:

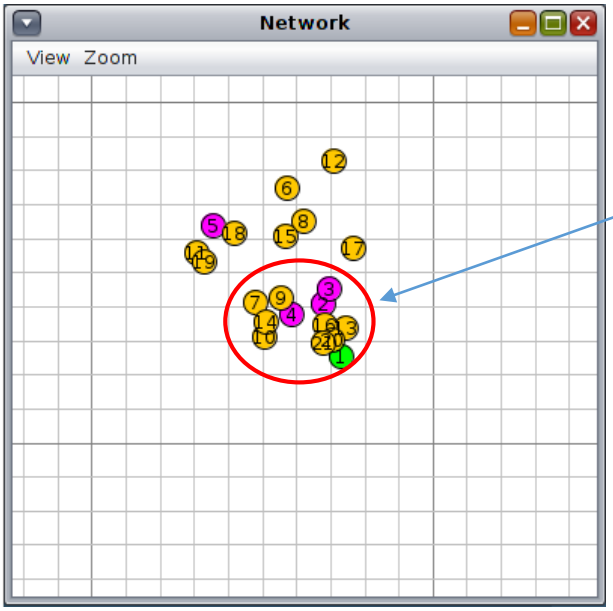

Ilustración 32: Movimiento 1

- Simulación 6:

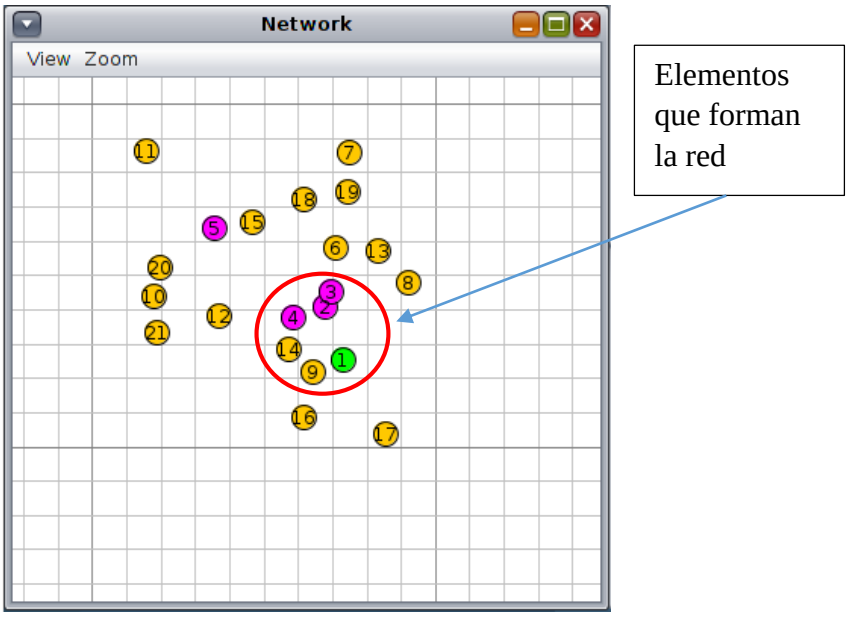

Ilustración 33: Movimiento 2

7. Conclusiones y trabajos futuros

Una vez concluido el proyecto, se puede decir que la implantación sobre el simulador de SDN-WISE permite a los usuarios conocer en un primer momento como serian esas redes IoT-SDN contextuales que se pretenden instalar pudiendo incluso pensar y comprobar si la distribución tanto del *Sink* como la de los nodos *skills* se adapta a las necesidades que se pretenden solucionar, ya que debido a la capacidad tanto de diseño de escenarios como la recolección de datos en los distintos informes, se ha conseguido crear una herramienta bastante completa.

De esta forma las conclusiones que se pueden sacar de este proyecto, en relación a los objetivos establecidos en un primer momento son:

- Explicación de redes IoT-SDN contextuales: Se ha conseguido dar una definición sobre este tipo de redes, indicando la necesidad actual que hay sobre ellas, así como las ventajas y las novedades que se aportan frente a los otros tipos de redes.
- Modificación del simulador de SDN-WISE para la introducción de redes IoT-SDN contextuales: Como uno de los puntos principales del proyecto, se ha conseguido modificar el simulador proporcionado por el equipo de S. Milardo, permitiendo la creación de distintos escenarios, así como la obtención de distintos datos de interés.
- Creación, ejecución y análisis de los escenarios: Mediante los generadores creados para ayudar a minimizar el esfuerzo empleado en la realización y ejecución de los escenarios, se consiguen varios resultados organizados en distintos informes, que sirven como datos para la creación de distintas gráficas permitiendo ver diversos aspecto de la red creada en el escenario como la eficiencia, los tiempos en los que se reciben los paquetes *reports*, etc. Con lo que se ha podido observar que las redes IoT-SDN contextuales funcionan mejor si son de un tamaño más bien pequeño y no hay más de 3 saltos de distancia entre el *Sink* y los otros tipos de nodos.

Desde otro punto de vista, esta herramienta para la creación y el análisis de redes IoT-SDN contextuales, crea un punto de partida para futuros estudios como:

- Implementar una automatización de la simulación: Como se ha visto actualmente se tiene que añadir la simulación al entorno de SDN-WISE y darle al botón de

Start, con lo que se pierde tiempo, por lo que se propone la implementación de algún script capaz de automatizar esta tarea.

- Implementar nuevas funciones a los nodos relacionadas con el contexto: Mediante la creación de nuevas funciones, se pueden crear nuevas implementaciones que permitan la modificación del contexto y su correspondiente agregación a los nodos *goals* y *skills*, como por ejemplo la implementación del envío de coordenadas en caso de emergencia.
- Añadir la capacidad de que haya más de un nodo *Sink*: Con la introducción de otros controladores en la red se pretende intentar dividir grandes redes en otras más pequeñas, intentando así, bajar la complejidad de estas y mejorando la eficiencia, el tiempo de cálculo de las distintas funciones, etc.

8. Bibliografía

[1] Javier Berrocal, José García-Alonso, Carlos Canal, Juan M. Murillo. Situational-Context: A Unified View of Everything Involved at a Particular Situation. En SPRINGER-VERLAG. Lecture Notes in Computer Science. Springer, 2018.

[2] Margaret Rouse, Ivy Wigmore. Internet de las cosas (IoT) [consulta: 14 noviembre 2018]. https://searchdatacenter.techtarget.com/es/definicion/Internet-de-las-cosas-IoT

[3] Feamster N., Rexford J., Zegura E. "The Road to SDN: An intellectual history of programmable networks. http://queue.acm.org/detail.cfm?id=2560327

[4] Patrick Jhon McGee. The Web of Things with Mozilla Open Badges [consulta: 14 noviembre 2018]. *"SlideShare The Web of Things with Mozilla Open Badges"*

[5] Innowiki Inteligencia Ambiental. [Consulta: 14 noviembre 2018]. http://185.5.126.23/innowiki/index.php/Inteligencia_ambiental

[6] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. Proc. of IEEE INFOCOM 2015. April 2015.

[7] S. Milardo. Introduzione a SDN-WISE. May 2015.

[8] David Mariscal. Software-Defined Networks (SDN), el futuro en las redes [consulta 20 de noviembre 2018]. https://www.beeva.com/beeva-view/tecnologia/software-defined-networks-sdn-el-futuro-en-las-redes/