Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

IMPLEMENTACIÓN DE SOFTWARE INTERACTIVO BASADO EN AODV

Alumno: **Jose Ramón Rodríguez Rodríguez**

Tutor: Prof. D. Ildefonso Ruano Ruano
Depto.: Ingeniería de Telecomunicación

UNIVERSIDAD DE JAÉN

ESCUELA POLITÉCNICA SUPERIOR DE LINARES

Departamento de Ingeniería de Telecomunicación

CURSO ACADÉMICO

2020-2021

TRABAJO FIN DE GRADO

TÍTULO DEL TRABAJO

Implementación de software interactivo basado en AODV

Grado en Ingeniería Telemática

AUTOR: Jose Ramón Rodríguez Rodríguez

TUTOR: D. Ildefonso Ruano Ruano

Linares, septiembre de 2021

1. ÍNDICE

2. RESUMEN

Cuando se tiene una necesidad de comunicación específica de duración limitada y no se cuenta con la infraestructura requerida para ello, la resolución del problema pasa por la formación de una red. Estas redes son conocidas como redes Ad-hoc, pueden formarse de manera cableada o inalámbrica (caso más usual) y sus principales características son su fácil montaje, flexibilidad y que no necesitan un nodo central o padre para su funcionamiento.

Este trabajo fin de grado (TFG) está centrado en este tipo de redes, más concretamente en las redes ad-hoc formadas por nodos que poseen capacidad libre de movimiento, conocidas como redes MANET (*Mobile Ad-hoc Network*) y más específicamente en el protocolo de enrutamiento para redes MANET conocido como AODV (*Ad hoc On-demand Distance Vector*). En las redes MANET, las topologías establecidas por los nodos son muy dinámicas debido a la capacidad de movimiento de los mismos. Además, en estas redes los nodos suelen colaborar entre sí para enviar información enrutando paquetes salto a salto. Estas características hacen que el enrutamiento de las redes MANET sea bastante diferente y más complejo al estudiado tradicionalmente. Existen múltiples protocolos de enrutamiento que pueden ser utilizados en una red MANET, uno de los más conocidos y populares es AODV.

Este Trabajo Fin de Grado se ha centrado en el estudio del protocolo AODV y el desarrollo de una serie de recursos docentes que permitan la comprensión del mismo. Estos recursos se han integrado como recursos de e-learning utilizando el estándar SCORM (*Sharable Content Object Reference Model*) para incluirse en plataformas de docencia virtual. Entre ellos destaca la programación de una serie de simulaciones gráficas interactivas que permiten observar y comprender el funcionamiento de una red MANET que usa AODV como protocolo de enrutamiento. El hecho de mostrar gráficamente el movimiento de los nodos de una red MANET y el funcionamiento detallado de AODV (incluso paso a paso), facilita mucho el aprendizaje y comprensión de estos conceptos, por los que los recursos desarrollados en el ámbito de este TFG resultan ser muy útiles a todas las personas interesadas, especialmente estudiantes.

3. INTRODUCCIÓN

3.1 Redes Ad-hoc

Entre las definiciones empleadas por la RAE para describir el término Ad-hoc se encuentran la de “Expresión usada para referirse a lo que se dice o hace solo para un fin determinado” o “Aquello adecuado, apropiado o dispuesto especialmente para un fin”. Basándose en estas definiciones, una red Ad-hoc es aquella que se forma con el fin de resolver un problema de comunicación específico y añadir que, para su funcionamiento no requiere de infraestructura de red ni un nodo central, por lo que todos los componentes de esta red están en igualdad de condiciones. Se puede decir que una red Ad-hoc es la forma más sencilla de montar una red de comunicación.

Este tipo de redes pueden formarse de forma inalámbrica, en períodos de duración limitada, permitiendo a varios dispositivos establecer una comunicación para un propósito específico, siendo cada uno de estos dispositivos los responsables de las operaciones de red, y todo esto sin ningún tipo de infraestructura. Las características más comunes que suelen tener las redes Ad-Hoc son:

- Destinadas a ser utilizadas por un periodo de tiempo limitado.
- Son redes fáciles de montar.
- Flexibles.
- Todos los terminales que conforman la red son independientes.
- Las comunicaciones son punto a punto (sin nodo central ni cualquier otra infraestructura de red).
- Ofrecen un servicio personalizado para una aplicación en específico.
- Emplean protocolos personalizados, dependiendo de la aplicación que motivó la creación de la red.
- Permiten la movilidad de los terminales que forman la red.
- Soporta multidifusión.

En cuanto a las principales limitaciones que suelen presentar las redes Ad-hoc se encuentran la falta de seguridad, tiempo de vida limitado (baterías) y una velocidad de transmisión de datos poco estable y posiblemente lenta.

Conociendo la definición, características y limitaciones de este tipo de redes, las situaciones en las que se pueden formar redes Ad-hoc son las siguientes:

- En ámbitos militares como campos de batalla o defensa nacional.
- En recuperaciones de desastres (Naturales o no).
- En aquellos entornos donde es imposible el montaje de infraestructuras.
- Cuando el montaje de infraestructuras supone un gasto excesivo.
- Para establecer comunicaciones entre vehículos móviles.
- Para ocio circunstancial.

Antes de continuar, es conveniente definir los siguientes tipos de redes, ya que todas ellas están relacionadas con las redes Ad-hoc, bien porque presentan características similares o porque son tipos particulares de redes ad-hoc:

- Redes en Malla (Mesh Network): Son aquellas redes que constan de una topología en la que cada nodo está conectado a varios nodos, permitiendo que todos ellos puedan transmitir y recibir sus datos además de reenviar datos del resto de nodos. Es decir, todos los nodos actúan como routers reenviando datos de los otros nodos, esta es una característica muy común en las redes ad-hoc.
- Redes MANET (Mobile Ad-hoc Network): Se tratan de redes compuestas por nodos capaces de desplazarse independientemente en cualquier dirección, esto produce que se modifiquen dinámicamente las condiciones de los enlaces entre los nodos que forman la red.
- Redes WSN (Wireless Sensors Network): Se trata de redes compuestas por pequeños ordenadores equipados con sensores, todos estos encargados de la realización de una tarea común. Algunas redes WSN pueden considerarse redes ad-hoc, o incluso MANET.
- Redes VANET (Vehicular Ad-hoc Network): Se tratan de redes MANET vehiculares, por lo que los nodos que forman la red serían vehículos de cualquier tipo.

En este TFG se profundiza en las redes MANET.

3.2 Redes MANET

Como se ha comentado anteriormente, las redes MANET son aquellas compuestas por nodos que poseen capacidad de movimiento libre. Se puede ver un ejemplo de cómo se forman este tipo de redes gracias a la siguiente figura:

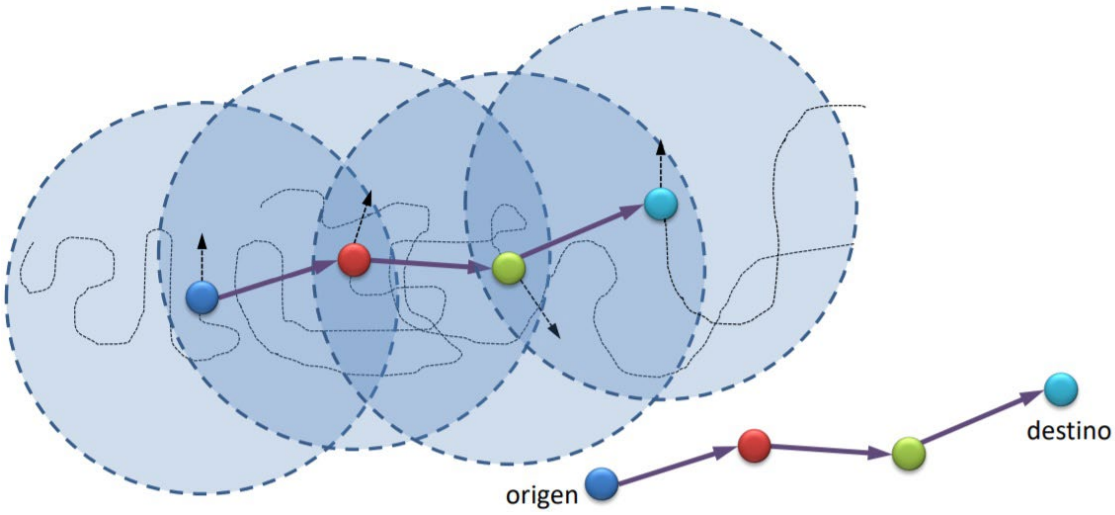


Figura 1: Formación de redes MANET. Fuente: https://dv.ujaen.es/goto_docencia_file_533989_download.html

Cada uno de los nodos que aparecen actúan como routers, permitiendo establecer rutas multisalto entre nodos y sin necesidad de un control central.

Entre las características que presentan las redes MANET, se encuentran:

- Presentan topologías dinámicas, lo que provoca que se creen/destruyan enlaces dinámicamente.
- Poseen un ancho de banda limitado.
- Es importante el ahorro de energía debido a que la alimentación de los nodos es por baterías (Energía limitada).
- Presentan problemas de seguridad ya que es un medio de acceso libre.
- No requieren de ninguna infraestructura para su formación.

A continuación, se hablará sobre los diversos protocolos de encaminamiento que las redes MANET pueden utilizar.

3.2.1 *Protocolos de Enrutamiento MANET*

En cuanto a los protocolos de encaminamiento a utilizar por este tipo de redes, hace unos años se evaluaron cerca de 60 propuestas diferentes [1], aunque en la actualidad sólo unas pocas propuestas han resistido la fuerte competencia. Algunas de aquellas propuestas que sobrevivieron poseen una RFC, como es el caso del protocolo AODV[2], en el cual se profundizará en este documento. También existen propuestas que se encuentran en desarrollo activo, por lo que no se ha impuesto ningún estándar real (AODVv2 posee un draft)[3].

El grupo de trabajo IETF[4] en MANET es el encargado de estandarizar la funcionalidad del protocolo IP en aplicaciones de enrutamiento inalámbrico con topologías estáticas y dinámicas con dinámica aumentada debido al movimiento del nodo u otros factores.

Las RFCs desarrolladas por este pueden ser de carácter experimental (representar el posible inicio de un futuro estándar), informacional (para ser inspeccionado), obsoleto (en desuso) o con un estándar propuesto. [5]

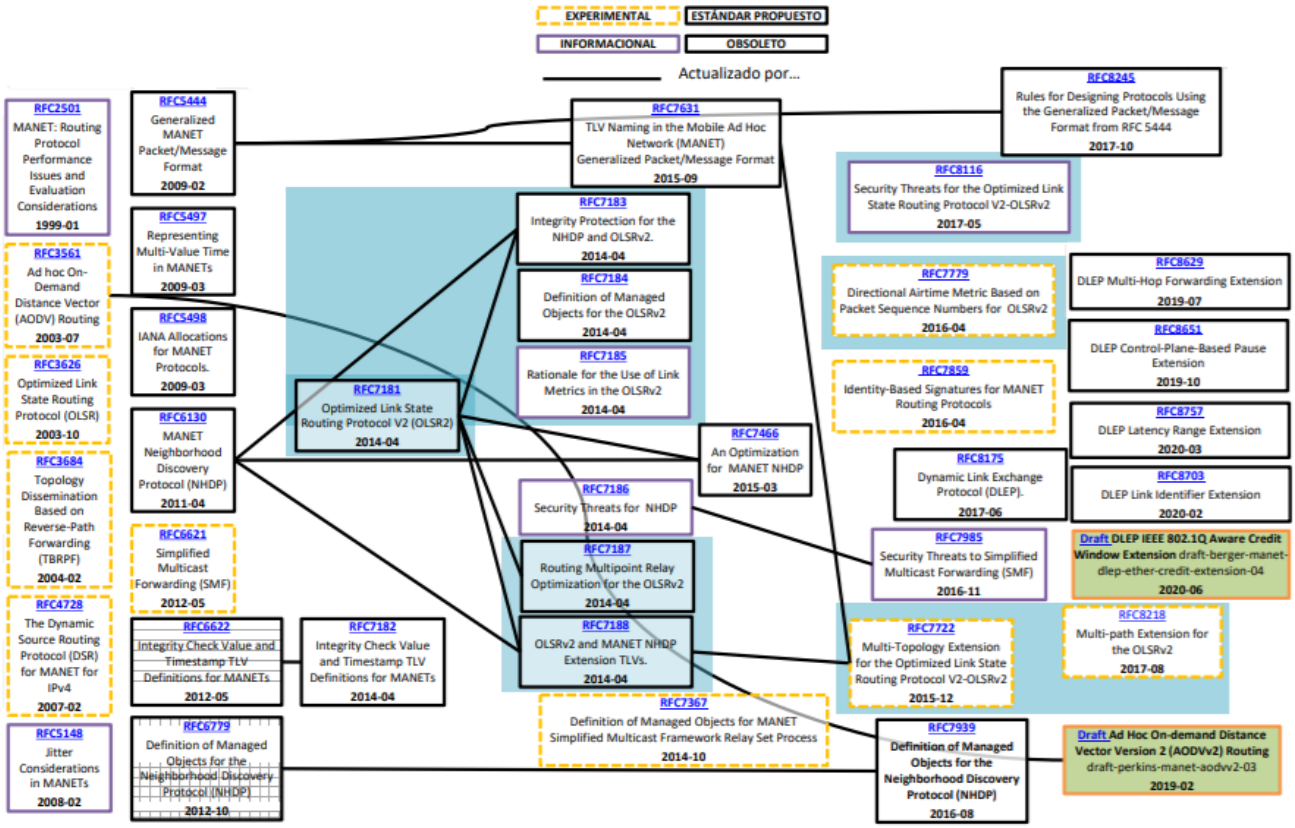


Figura 2: RFCs desarrolladas para redes MANET. Fuente: https://datatracker.ietf.org/wg/manet/documents/

El principal objetivo del encaminamiento en MANET es el de encontrar rutas óptimas en relación con un parámetro o conjunto de parámetros determinado. Otros de los objetivos que se presentan son:

- Minimización de costes inherentes a través de la reducción de la cantidad de mensajes de control intercambiados y la carga computacional de operaciones.
- Eliminación de bucles entre nodos.
- Establecer un mantenimiento dinámico de la topología.

Los protocolos de encaminamiento que utilizan las redes MANET están organizados dependiendo de varios factores como su estructura o procedimientos que emplean para el descubrimiento de rutas y su mantenimiento. Dentro de esta organización caben destacar los protocolos reactivos y proactivos[1]. Se dice que son reactivos (bajo demanda) cuando las rutas que obtienen los nodos se construyen únicamente en el momento en el que se necesita establecer una comunicación. En cambio, un protocolo de encaminamiento es proactivo (basado en tablas) cuando, desde el momento inicial, todos los nodos que componen la red poseen la información necesaria para comunicarse con el resto de nodos.

Tras esta breve descripción, el protocolo AODV estaría encasillado como un protocolo reactivo, además de uniforme, ya que todos los nodos que conforman la red desempeñan funciones iguales y poseen las mismas características.

3.2.2 Protocolo de enrutamiento AODV

Ad hoc On-Demand Distance Vector, mejor conocido como AODV, es un protocolo de enrutamiento diseñado para redes MANET, estandarizado por el IETF en la RFC 3561.

Este protocolo soporta poblaciones que pueden ir desde decenas hasta miles de nodos móviles, además puede manejar tasas de movilidad bajas, moderadas y relativamente altas. Es necesario que todos los nodos que conforman la red confíen entre sí, ya que este protocolo no presenta seguridad.

Como protocolo reactivo, AODV solo construye rutas cuando un nodo necesita establecer una comunicación. Para ello, cuando un nodo desea obtener una ruta hacia otro, el protocolo inicia una fase de descubrimiento de ruta a través de la inundación de mensajes de petición de ruta, que concluye al recibir una respuesta del nodo destino, el cual proporciona la ruta deseada.

3.3 JavaScript

Para explicar de dónde surge JavaScript[6], hay que remontarse a los años 90, donde se empezó a desarrollar la programación de páginas web, a las que se les comenzaron a incluir aplicaciones web. El problema surgió cuando estas aplicaciones web fueron aumentando de complejidad dentro de las páginas que las albergaban, pero la velocidad de conexión no aumento o mejoró al mismo ritmo. Por ello, surgió la idea de desarrollar un lenguaje de programación capaz de ser ejecutado en el lado del cliente en lugar del servidor, permitiendo así reducir los tiempos de espera.

JavaScript consiste en un lenguaje de programación capaz de ofrecer una mayor interactividad y dinamismo a las páginas web. Uno de los puntos positivos que posee JavaScript es que no se requiere de un compilador cuando se ejecuta en un navegador web, ya que este sería el encargado de leer directamente el código implementado.

JavaScript conforma junto con HTML (HyperText Markup Language) y CSS (Cascading Style Sheets) los tres lenguajes nativos de la web.

Como se ha dicho anteriormente, es un lenguaje de programación posicionado del lado del cliente, al cual le permite desarrollar efectos y animaciones sin ninguna interacción, o respondiendo a eventos causados por el propio usuario tales como botones pulsados y modificaciones del DOM (Document Object Model), el cual se comentará en que consiste más adelante. El código de programación de JavaScript se ejecuta en los navegadores, ya sean de escritorio o móviles.

3.4 HTML

HTML[7] (HyperText Markup Language) fue inventado por Tim Berners-Lee, un físico del instituto de investigación CERN en suiza. Es un lenguaje informático con el que se han desarrollado la mayoría de páginas web y aplicaciones en internet.

Un hipertexto se trata de un sistema de organización y presentación de datos basado en la vinculación de fragmentos textuales o gráficos a otros fragmentos, mientras que un lenguaje de marcado consiste en una serie de marcas que indican a los servidores web la estructura y el estilo del documento.

Al no poder crear funcionalidades dinámicas, HTML no es considerado un lenguaje de programación, pero en su lugar, gracias a HTML, los usuarios pueden implementar y estructurar secciones, párrafos y enlaces mediante elementos, etiquetas y atributos.

Actualmente, HTML es considerado un estándar oficial de la web, siendo el W3C (World Wide Web Consortium) el encargado de mantener y desarrollar las especificaciones de HTML, además de proporcionar actualizaciones periódicas.

Algunos de los usos más comunes de HTML son:

- Desarrollo web: Los desarrolladores utilizan el código HTML para diseñar la forma en la que un navegador muestra los elementos de una página web.
- Navegación por internet: Los usuarios pueden navegar fácilmente entre páginas ya que HTML soporta la capacidad de incrustar hipervínculos.
- Documentación web: HTML permite organizar y dar formato a los documentos.

3.4.1 Estructura básica del documento HTML

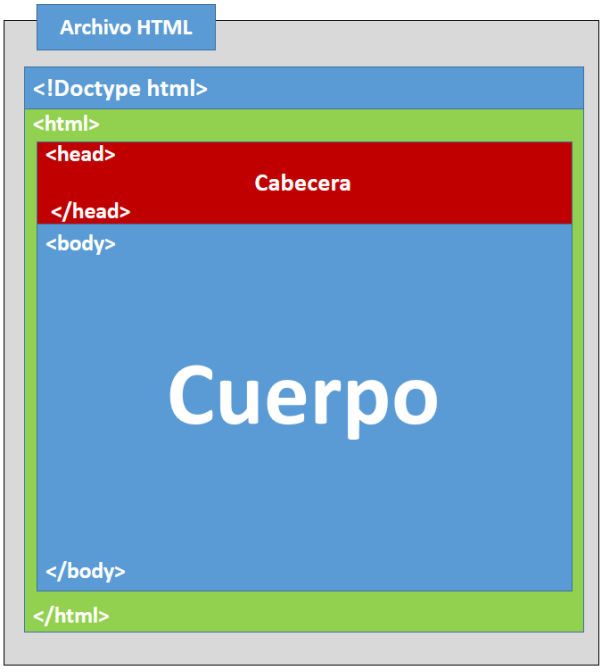


Figura 3: Estructura básica de un documento en HTML. Fuente: https://studentplace98.blogspot.com/2019/04/EstructuraBasica.html

Como se puede apreciar en Figura 3, la estructura de un documento HTML[8] está compuesta por los siguientes componentes:

- <!Doctype html>: Etiqueta empleada para notificar al navegador el tipo de documento que se va a crear, así como su versión.
- <html>: Etiqueta utilizada para indicar al navegador el principio y final de un documento.

- <head>: Es la cabecera del documento HTML. Dentro de estas etiquetas se encuentran datos como el título de la pestaña, los estilos de diseño (CSS), código JavaScript, título del documento, etc.
- <body>: Etiqueta que indica el origen y el final del cuerpo de la página, donde se desarrolla texto, incrusta imágenes, tablas, etc. Se implementa todo aquello que el usuario verá.

3.5 CSS

Se trata de un lenguaje conocido como hoja de estilo en cascada o CSS[9] (Cascading Style Sheets) el cual se utiliza para definir la presentación de un documento estructurado implementado en HTML y derivados. Se les conoce como estilos en cascada porque se aplican de arriba abajo. El W3C se encarga de desarrollar la especificación de las hojas de estilo, las cuales servirán de estándar para los navegadores web.

La idea de CSS es la de emplear el concepto de separación de presentación y contenido, intentando que los documentos HTML incluyan solo información y datos, mientras que todos los aspectos relacionados con el estilo (diseño, colores, formas, etc..) aparezca implementado en un documento CSS.

Figura 4: Implementación de una página web utilizando HTML y CSS. Fuente: https://lenguajecss.com/css/introduccion/que-es-css/

Esto ofrece las siguientes ventajas:

- Las modificaciones visuales se realizan en un solo lugar, sin tener que editar el documento HTML.

- Se reduce la duplicación de estilos.
- Resulta más sencillo implementar diferentes versiones de presentación para otros tipos de dispositivos.

3.6 DOM

DOM[10] (Document Object Model) es una plataforma e interfaz de lenguaje neutro que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de un documento. Es un estándar W3C (World Wide Web Consortium) que se divide en tres partes diferentes:

- Core DOM: Modelo estándar para todos los tipos de documentos.
- XML DOM: Modelo estándar para documentos XML.
- HTML DOM: Modelo estándar para documentos HTML.

Gracias a HTML DOM, JavaScript puede acceder y cambiar todos los elementos de un documento HTML. Cuando se carga una página web, el navegador crea un DOM de la página.

El modelo HTML DOM se construye como un árbol de objetos:

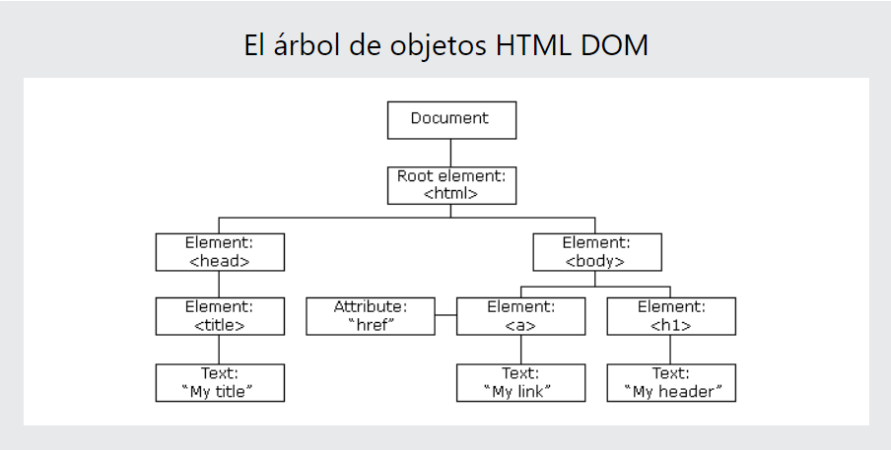


Figura 5: Árbol de objetos HTML DOM. Fuente: https://www.w3schools.com/js/js_htmldom.asp

Así JavaScript obtiene todas las herramientas para desarrollar HTML dinámico, permitiendo insertar o modificar todos los elementos HTML de una página, modificar atributos HTML de una página, modificar los estilos CSS de una página y eliminar tanto elementos como atributos HTML ya creados.

3.7 XML

Es un subconjunto de SGML (Estándar Generalised Mark-up language), simplificado y adaptado a Internet.

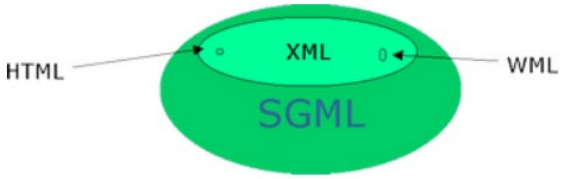


Figura 6: Estructura del estándar SGML. Fuente: https://www.mundolinux.info/que-es-xml.htm

Consiste en una especificación para diseñar lenguajes de marcado que permite definir etiquetas personalizadas para descripción y organización de datos. La tarea que tiene XML[11] es la de representar información de forma estructurada en la web, de manera que dicha información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por diversos tipos de aplicaciones y dispositivos.

Entre las principales características que presenta XML, destacan:

- Extensibilidad
- Estructura
- Validación

Las ventajas que proporciona XML son:

- Es fácilmente procesable
- Separa el contenido del formato de presentación
- Diseñado para cualquier lenguaje y alfabeto

3.8 QTI

Las especificaciones de las preguntas y exámenes (Question & Test Interoperability (QTI) [12]) establecen la estructura básica necesaria para la representación de preguntas y exámenes (items y assessment). Gracias a esta especificación se permite el intercambio

de preguntas y exámenes entre sistemas para la gestión del aprendizaje (Learning Management Systems (LMS)). La especificación del QTI se define en lenguaje XML.

El grupo de trabajo encargado de QTI tiene como principales objetivos:

- Proporcionar exámenes o bancos de preguntas a los usuarios de los entornos virtuales de enseñanza.
- Utilizar exámenes y bancos de preguntas procedentes de diversas fuentes no vinculadas a un solo sistema virtual de enseñanza
- Dar soporte para las herramientas que permiten desarrollar nuevos exámenes y preguntas
- Capacidad de generar informes en base a los resultados de las pruebas realizadas

Un documento QTI posee la siguiente estructura:

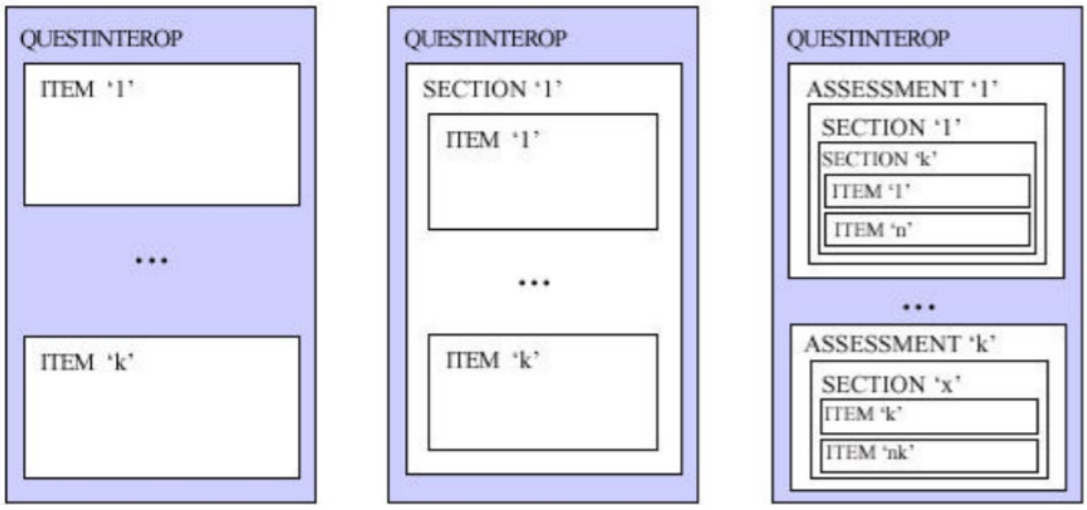


Figura 7: Esquema de la estructura básica adoptada por el QTI. Fuente: http://pdi.topografia.upm.es/m.manso/docs/Estandar_QTI_Descripcion.pdf

- ITEM: Son las unidades mínimas que pueden ser intercambiadas usando QTI
- Sections: Un archivo QTI-XML puede contener varias secciones. Estas secciones pueden contener más secciones o items
- Assessment: Un assesstment es un examen dentro de un documento QTI-XML. Un archivo QTI-XML puede contener más de un examen. Cada examen debe contener al menos una sección.

  En cuanto a un sistema de exámenes, este presenta el siguiente esquema:

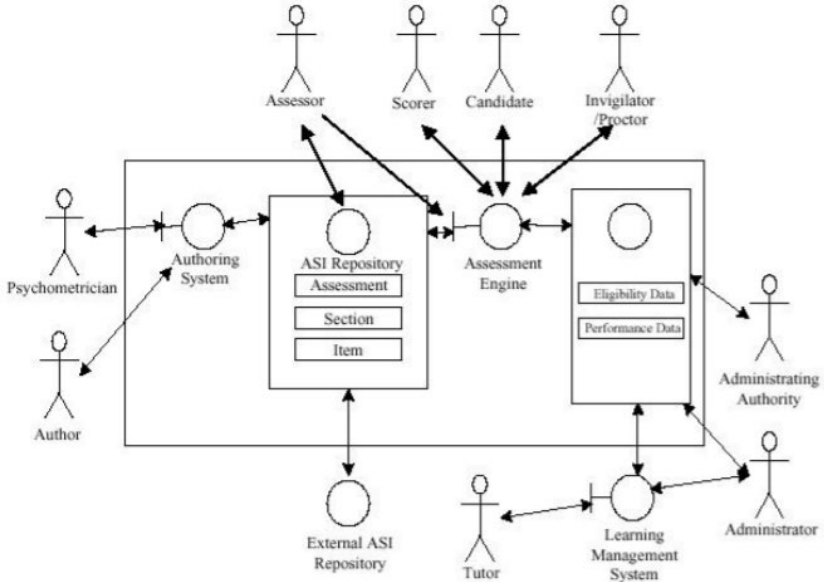


Figura 8: Casos de uso del sistema de exámenes. Fuente: http://pdi.topografia.upm.es/m.manso/docs/Estandar_QTI_Descripcion.pdf

- Authoring system: Es el proceso que soporta la creación y edición de exámenes, secciones y preguntas.
- Assessment engine: Es el proceso que soporta la evaluación o calificación de las respuestas de forma que genere calificaciones y consejos.
- Learning Management System: Sistema o proceso responsable de la gestión completa de la arquitectura de enseñanza.
- Candidate and Repository: Base de datos con las especificaciones para el candidato (alumno).
- ASI repository: Base de datos local de exámenes, secciones y preguntas.
- External ASI repository: Base de datos externa que debe ser importada haciendo uso de las especificaciones del QTI.

3.9 E-Learning

En la actualidad, gracias a internet se puede tener acceso a todo tipo de información en cualquier momento y desde cualquier sitio. El E-Learning[13] se nutre de esto, ya que se trata de la enseñanza y aprendizaje recibido a través de internet y las tecnologías, permitiendo acceder a dicha enseñanza en función de las diversas necesidades personales de cada uno, ya que consiste en una modalidad de formación que ofrece flexibilidad y personalización en los procesos de aprendizaje.

En esta modalidad de enseñanza online, el usuario aprendiz puede interactuar con el material de aprendizaje. Para lograr esto, es necesario el uso de un entorno de aprendizaje virtual, este es comúnmente conocido como LMS (Learning Management System).

LMS es donde se desarrolla el curso o temario, se encarga de gestionar los contenidos del mismo y permite a los docentes conocer las interacciones realizadas por el alumno.

En resumidas cuentas, el E-Learning constituye una herramienta de aprendizaje fácil de utilizar, capaz de soportar diversos formatos multimedia (texto, audio, video, imagen), resulta económico para el alumnado, además de interactivo y flexible, ya que su uso no depende de horarios fijos ni sitios determinados.

3.10 SCORM

SCORM [14] (Sharable Content Object Reference Model) consiste en un conjunto de estándares y especificaciones los cuales permiten el desarrollo de módulos de aprendizaje estructurados. SCORM podría entenderse como el formato estándar de contenidos E-Learning.

SCORM concibe la arquitectura[15] del aprendizaje bajo la filosofía Cliente-Servidor. El cliente es la combinación del estudiante y el objeto educativo. El servidor es el entorno de ejecución que normalmente es soportado por la plataforma de aprendizaje, LMS. El término LMS implica un entorno cliente-servidor que contiene la inteligencia suficiente para gestionar y distribuir los objetos educativos a los estudiantes. Es el LMS quien determina que debe entregar a cada alumno y cuándo hacerlo dependiendo de las interacciones de cada alumno a través de los contenidos. Simplificando, la relación que existe entre LMS y SCORM es que un módulo SCORM equivaldría a un DVD, y el LMS sería el lector de DVD.

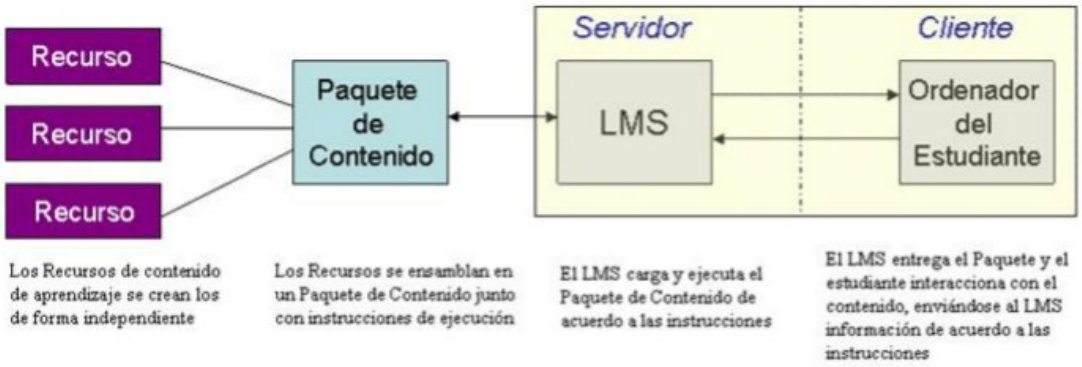


Figura 9: Arquitectura del modelo SCORM. Fuente: https://www.unpa.edu.ar/sites/default/files/descargas/Administracion_y_Apoyo/4.%20Materiales/2015/T093/SCORM_Standar.pdf

Gracias a la aparición del formato SCORM, creado allá por el año 1999 por los laboratorios ADL (Advanced Distributed Learning), se resolvió el problema que impedía el intercambio de contenidos entre plataformas de formación, ya que estas contaban con sus propios formatos. Ahora la mayoría de los LMS están diseñados para la lectura del formato SCORM.

Desde la creación de SCORM, este ha contado con varias versiones:

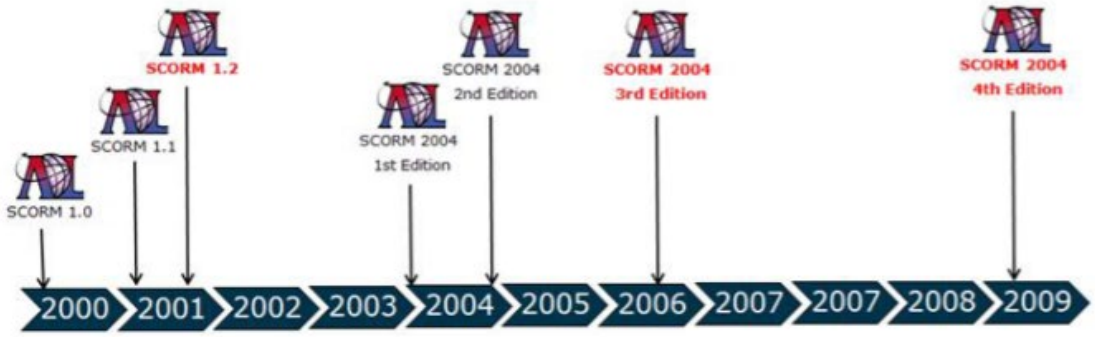


Figura 10: Evolución de las diferentes versiones de SCORM. Fuente: https://www.unpa.edu.ar/sites/default/files/descargas/Administracion_y_Apoyo/4.%20Materiales/2015/T093/SCORM_Standar.pdf

De entre las distintas versiones que podemos observar en la figura 10, destacan la versión 1.2, actualmente la más empleada y es soportada por la mayoría de los LMS que existen, aunque presenta como desventaja que carece de especificaciones de secuenciación y navegación[16], por lo que no se puede especificar cómo el estudiante progresa por los contenidos de aprendizaje.

La otra versión destacada es la 2004, la cual introdujo la funcionalidad conocida como "Secuenciación y navegación" que permite generar reglas para conocer cómo los usuarios

pueden navegar a través de los contenidos de aprendizaje. Estos contenidos de aprendizaje son conocidos como SCO (Shareable Content Objects).

Cuando se genera un paquete SCORM, entre los ficheros que este posee, encontraremos el imsmanifest.xml, el cual es el encargado de manejar el contenido del módulo SCORM, estableciendo el orden de aparición de los SCOs.

Las ventajas que SCORM nos ofrecen son:

- Accesibilidad: Alumnos y profesores pueden acceder a la formación desde cualquier lugar, siempre que se disponga de una conexión a internet.
- Adaptabilidad: Los SCORM son personalizables a las necesidades del alumno.
- Durabilidad: Al tratarse de un sistema sencillo, no requiere de grandes reconfiguraciones, por lo que una vez adquirido se mantiene perfectamente a lo largo del tiempo.
- Interoperabilidad: Al ser un formato mundialmente aceptado permite que cualquier LMS esté listo para su reproducción.
- Reusabilidad: Soporta flexibilidad a la hora de añadir nuevos componentes de aprendizaje dentro de los módulos ya desarrollados.
- Reducción de costes: Los paquetes SCORM, gracias a las ventajas vistas anteriormente, permiten reducir drásticamente los costes de formación de las organizaciones.

3.11 Herramientas de simulación de redes

En la actualidad, existes diversas herramientas[17] capaces de simular el funcionamiento de los protocolos de encaminamiento utilizados en redes ad-hoc. A continuación, se describirán algunas de estas herramientas:

- ns[18]: Su desarrollo comenzó en 1989. Se trata de un simulador de redes basado en eventos discretos utilizado principalmente en ambientes educativos y de investigación. Este permite simular tanto protocolos unicast como multicast y se emplea sobre todo para la investigación en el ámbito de las redes ad-hoc. Es un software libre que cuenta con las versiones ns-2 y ns-3.

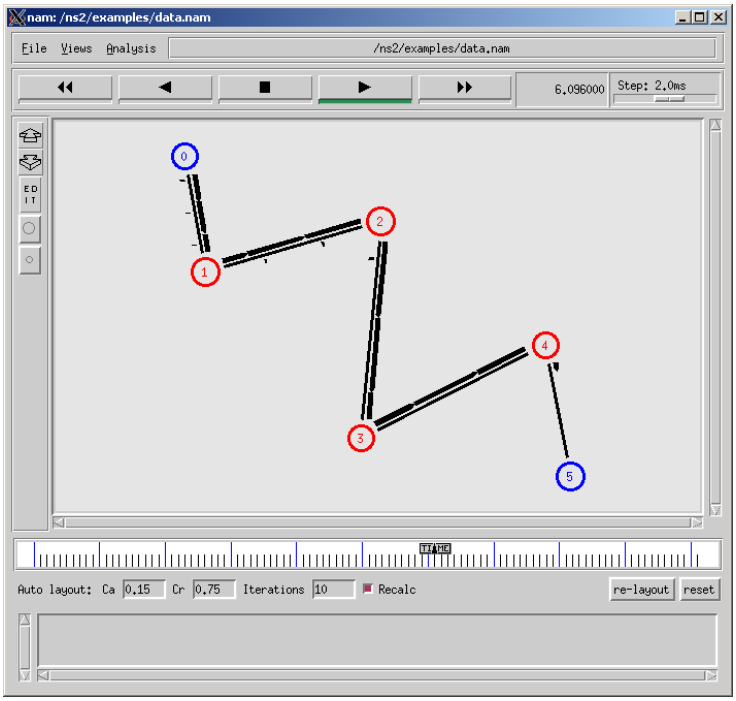


Figura 11: Interfaz del simulador ns-2. Fuente: https://unix.stackexchange.com/questions/64159/network-simulator-ns-2-modeling-of-fragmentation-in-the-network-with-different

- OMNET: Es un simulador modular, el cual está tomando protagonismo en los últimos años gracias a su intuitiva y sencilla interfaz gráfica y a su poderosa GUI para animación y depuración. Presenta como desventaja más significativa la carencia de protocolos disponibles en sus librerías, por lo que los usuarios deben implementarlos.

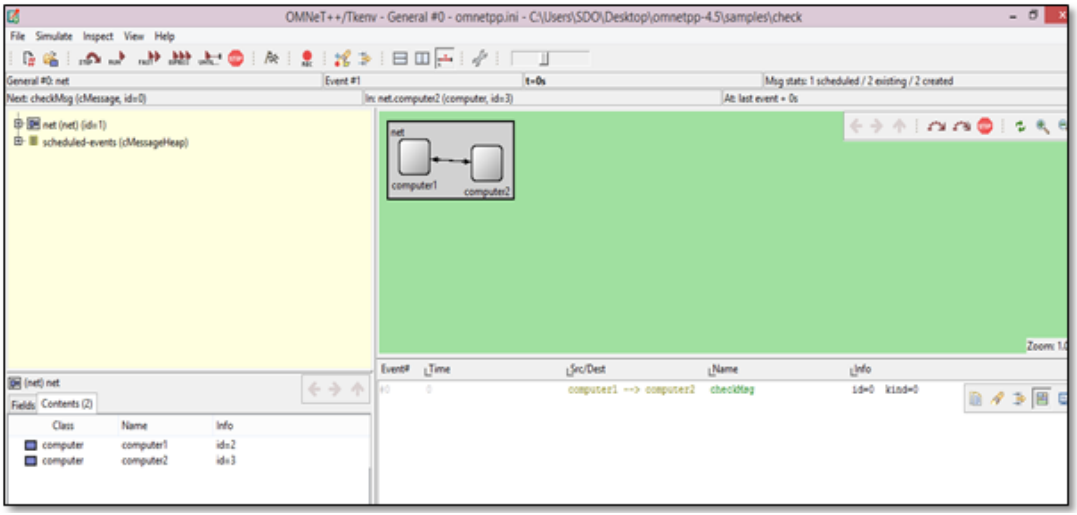


Figura 12: Interfaz del simulador OMNET. Fuente: http://inalambricas-lte4g.blogspot.com/2014/08/simulador-omnet.html

- J-sim: Es un simulador desarrollado completamente en Java. La mayor ventaja que proporciona J-Sim es la gran cantidad de protocolos soportados. Los modelos en J-Sim son reusables e intercambiables ofreciendo una amplia flexibilidad.

-

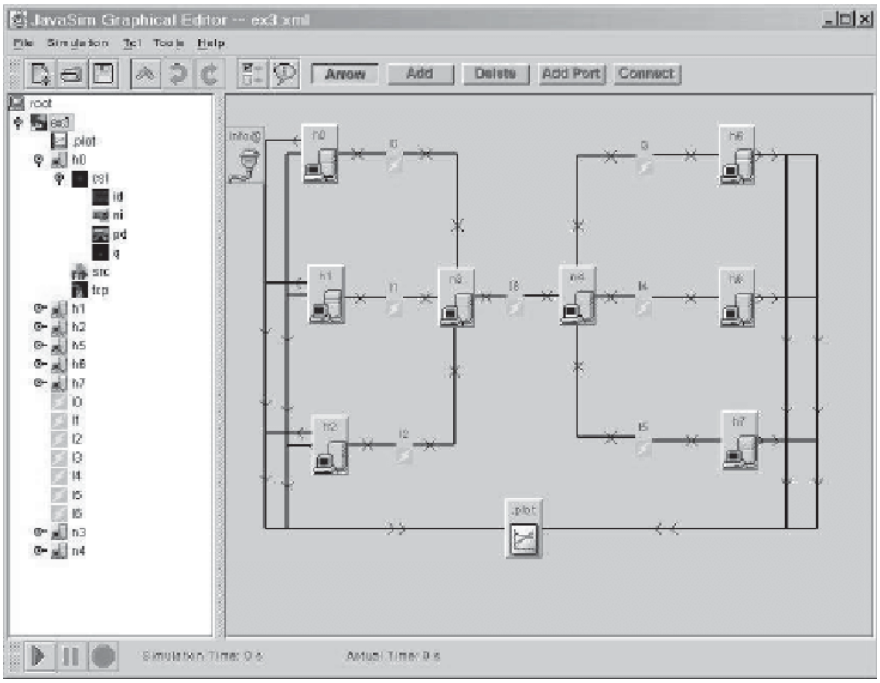


Figura 13: Interfaz del simulador J-Sim. Fuente: https://www.researchgate.net/figure/J-Sim-gedit-user-interface_fig6_320650642

- OPNET: Esta herramienta de simulación proporciona un entorno virtual de red que modela el comportamiento de una red por completo, incluyendo sus pasarelas (routers), conmutadores (switches), protocolos, servidores y aplicaciones en red. Esta herramienta es ampliamente usada en ambientes profesionales de diseño de redes.

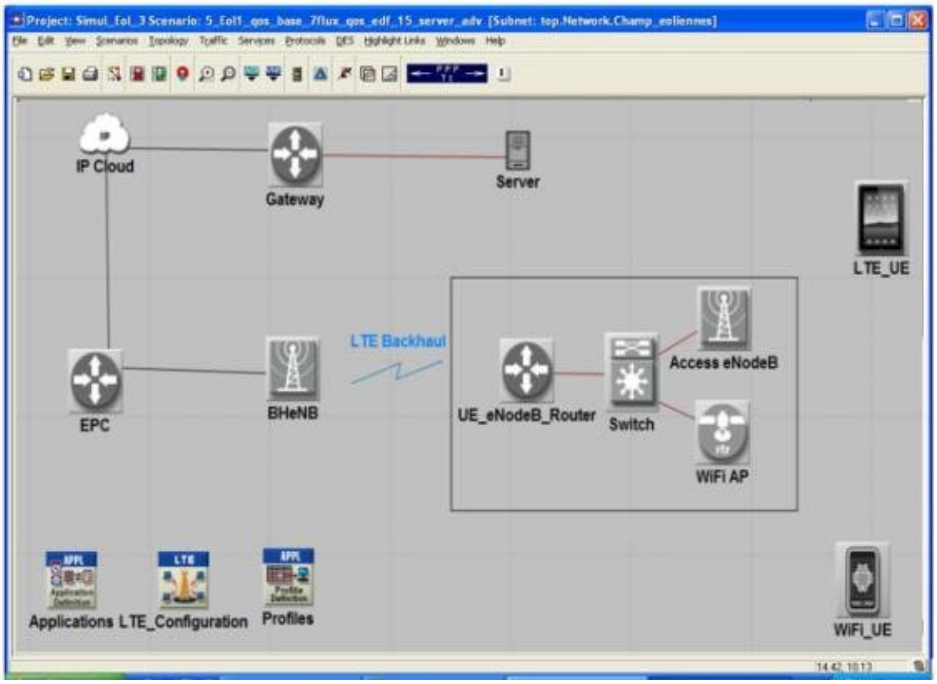


Figura 14: Interfaz del simulador OPNET. Fuente: https://www.researchgate.net/figure/OPNET-Modeler-Interface_fig1_340261604

4. MOTIVACIÓN Y OBJETIVOS

Existen diversos recursos que permiten conocer los distintos aspectos y características del protocolo AODV. A través de su RFC, materiales teóricos que ofrece la universidad o gracias a una simple búsqueda en internet, se puede encontrar infinidad de información sobre este protocolo. Además, existen simuladores de red como los vistos anteriormente capaces de simular el funcionamiento de AODV, permitiendo conocer cómo reacciona el protocolo ante diversas topologías y tasas de movilidad, entre otras variables, y compararlo así con otros protocolos de enrutamiento. Todo esto está muy bien, pero desde el punto de vista del alumno, llegar a comprender correctamente los aspectos y el funcionamiento de este protocolo a través de la extensa información que proporcionan estos recursos de internet, puede resultar complicado.

De aquí surge la motivación para desarrollar este Trabajo Fin de Grado. La idea consiste en el desarrollo de una simulación interactiva implementada a través de la herramienta de software EJS (Easy Java Simulation), la cual permite ver gráficamente cómo actúa el protocolo AODV para cada una de las tareas que este realiza, observando todo paso a paso, a la vez que es explicado por una consola que posee la propia simulación. Además, la simulación nos mostrará a tiempo real, el formato de los mensajes que AODV está utilizando y cómo evolucionan las tablas de enrutamiento de todos los nodos involucrados en la red. Incluso, si el alumno lo desea, puede cambiar el idioma de

esta simulación, ya que está disponible tanto en inglés como en español. Además, se ha creado un módulo SCORM, el cual al estar integrado en LMS, podrá recabar información variada sobre las interacciones de los alumnos dentro de dicho módulo. Este ofrece una teoría básica y clara sobre redes Ad-Hoc y MANET, hasta llegar al protocolo AODV. También contiene unos test que permitirá a los administradores del módulo conocer el proceso de aprendizaje de los alumnos. Todo esto estará estructurado alrededor de la simulación descrita anteriormente. Además, se ha añadido a este SCORM un conjunto de actividades/tutorial que nos enseña a utilizar esta simulación antes de acceder a ella, incluyendo de un vídeo explicativo sobre la misma.

Para desarrollar todo esto, se han debido de lograr una serie de objetivos:

- Para la comprensión del tema:
  - Estudio previo de las redes Ad-Hoc, redes MANET y protocolos de enrutamiento de las redes MANET, profundizando en AODV.
- Para el desarrollo de la simulación:
  - Comprensión del lenguaje de programación JavaScript.
  - Comprensión de la herramienta EJS (Easy Java Simulations) utilizada para la creación de la simulación.
- Para la creación del módulo SCORM en ILIAS:
  - Estudio de normas y estándares de SCORM.
  - Desarrollo de páginas web por medio de HTML y CSS.
  - Introducción de información teórica en base al estudio previo realizado sobre el protocolo AODV.
  - Introducción de preguntas test en algunas de las páginas web desarrolladas.
  - Desarrollo de un video explicativo que facilite la comprensión del uso del simulador AODV.
  - Inclusión de nuevo código sobre los HTML y XHTML para conseguir que trabaje SCORM, así como la adición de las librerías necesarias para su funcionamiento.
- Para la creación de preguntas tipo test:
  - Estudio del estándar QTI.
  - Comprensión de las librerías JavaScript que permiten la inclusión de los bancos de preguntas a las páginas web desarrolladas.
  - Desarrollo de banco de preguntas para test de evaluación inicial.
  - Desarrollo de banco de preguntas para test de evaluación final.

Con la consecución de estos objetivos, se obtiene como resultados:

- Desarrollo de un módulo SCORM funcional que permite conocer al usuario externo el funcionamiento del protocolo AODV.
- Desarrollo de una simulación independiente capaz de recrear el comportamiento del protocolo AODV.

5. HERRAMIENTAS Y MÉTODOS

5.1 EJS

Para el desarrollo de la simulación nombrada, la cual facilitará la comprensión del protocolo AODV, se utilizará una herramienta de software diseñada para la creación de simulaciones gráficas de forma sencilla, conocida como EJS (Easy Java Simulations). EJS permite exportar los proyectos creados con formato SCORM. Además, estos proyectos pueden ser creados en Java o JavaScript. En el caso de la simulación desarrollada, se ha utilizado el lenguaje de programación JavaScript.

La primera interfaz que se nos muestra cuando se accede a EJS es la siguiente:

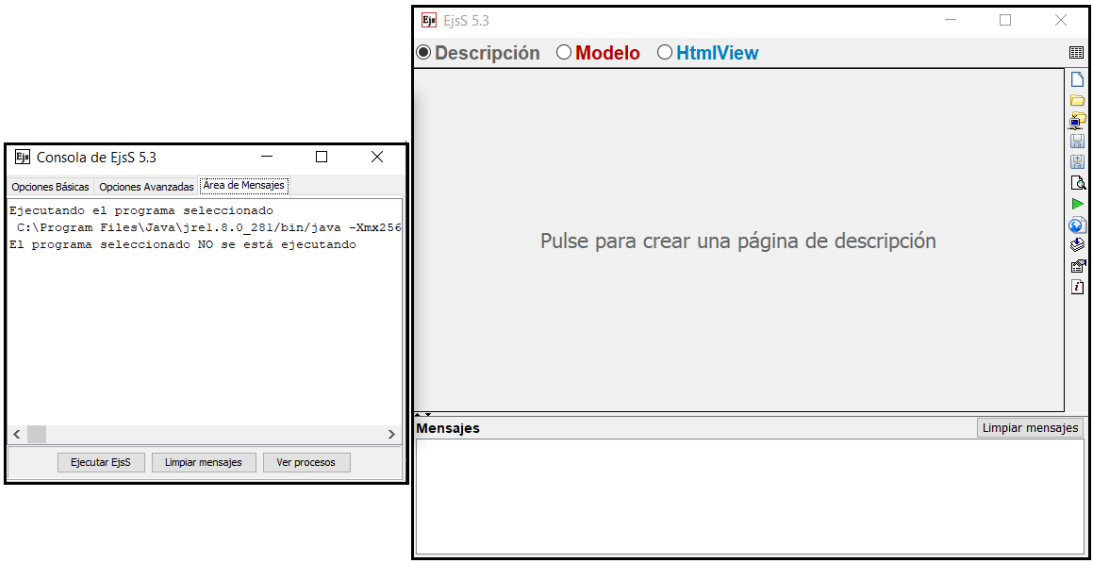


Figura 15: Primer vistazo a la interfaz gráfica de EJS. Fuente: Propia

La pestaña de la izquierda se trata de la consola de EJS. En esta, se permite configurar el programa, escoger el directorio donde se almacenará la simulación, así como definir el lenguaje de programación empleado. La interesante es la pestaña de la parte derecha, donde se desarrolla toda la simulación. Esta posee tres apartados:

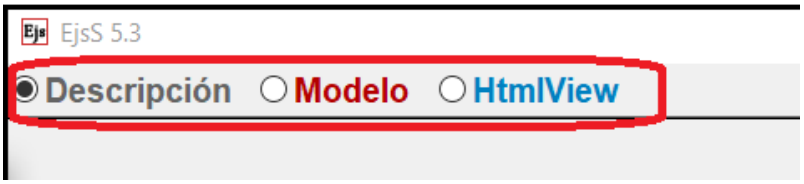


Figura 16: Diferentes apartados que se emplean en EJS. Fuente: Propia

En “Descripción” se puede añadir una descripción del proyecto desarrollado, pudiéndose incluir imágenes, tablas, formularios, etc.

El apartado “Modelo” alberga todo lo relacionado con la lógica de la simulación, ya que es la parte donde se desarrollan y se gestionan las diversas funcionalidades que se realizan para recrear el comportamiento del protocolo AODV.

Además, este apartado se divide en varios subapartados que permitirán estructurar mejor el código implementado:

Figura 17: Subapartados del apartado "Modelo". Fuente: Propia

- Variables: Parte donde se definen las variables que se utilizarán para el desarrollo de la simulación.
- Inicialización: Contiene el código que define las condiciones y aspecto inicial que presentará la simulación desarrollada.
- Evolución: Parte donde se definen todas las tareas a realizar por la simulación durante un ciclo de ejecución. Los ciclos de ejecución están siendo repetidos constantemente mientras la simulación está activa. En esta parte, se hacen llamadas a las funciones definidas, permitiendo completar la funcionalidad del simulador.
- Relaciones fijas: Contiene las partes del código que no puedan ser modificadas.
- Propio: Parte donde se definen y desarrollan las funciones que permitirán recrear en el simulador el comportamiento del protocolo AODV.
- Elementos: Parte donde quedan definidos los elementos que se quieran utilizar en el proyecto.

El último apartado es el “HTMLView”.

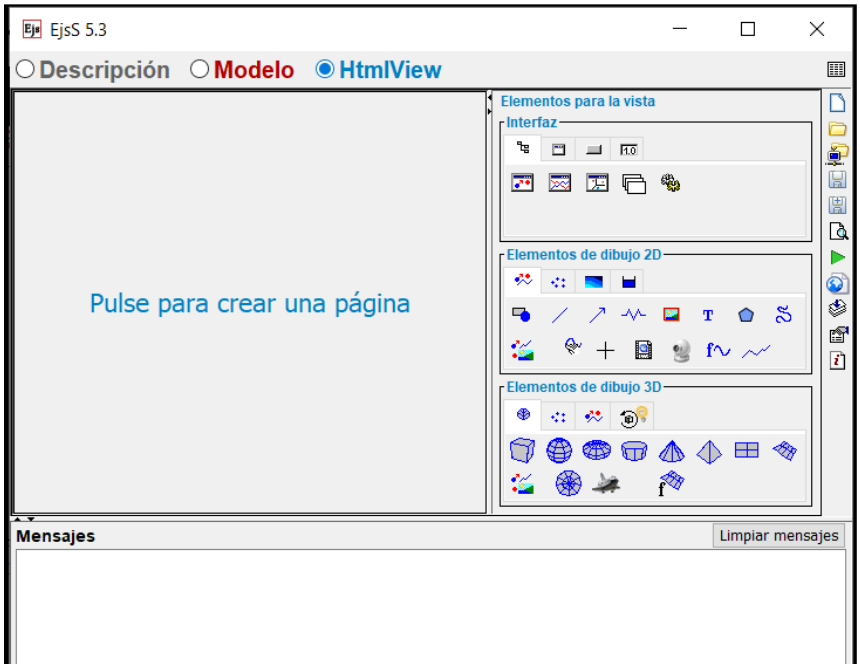


Figura 18: Pestaña HTMLView. Fuente: Propia

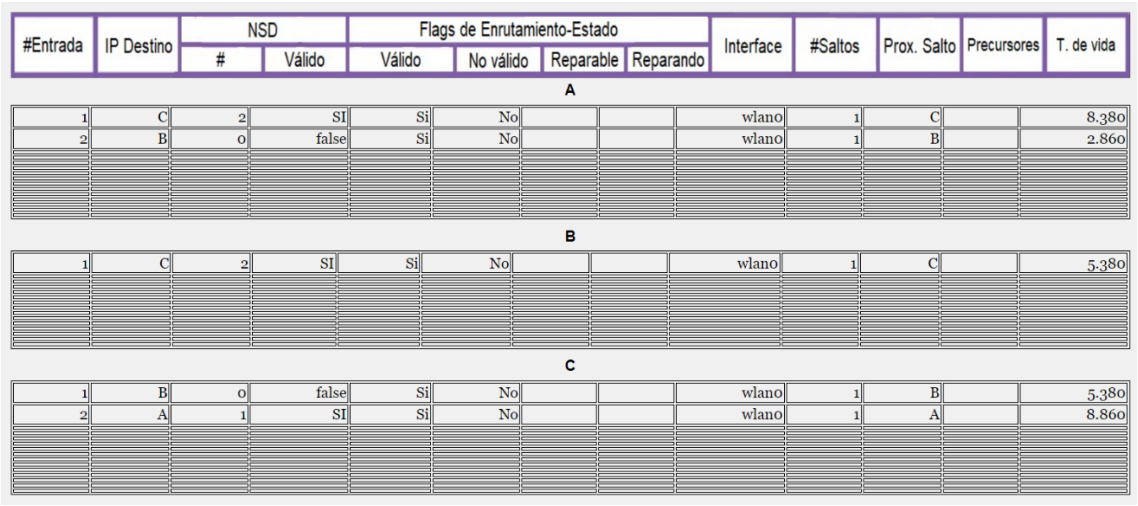
Este apartado permitirá desarrollar la interfaz gráfica que tendrá la simulación.

5.2 Componentes del protocolo AODV

Los componentes que entran en juego y son necesarios tener claro para comprender cómo funciona este protocolo son:

- Cada uno de los nodos móviles que componen la red tienen asociada una tabla de enrutamiento diferente.
- Estas tablas de enrutamiento se rellenan con rutas que se establecen gracias a los diferentes tipos de mensajes que AODV utiliza, (RREQ, RREP y RERR) los cuales se desarrollarán más adelante.
- Cada una de las rutas que se apuntan en estas tablas de enrutamiento tienen asociado un Número de Secuencia de Destino (NSD) con el fin de evitar bucles.

5.2.1 Tablas de enrutamiento



#Entrada	IP Destino	NSD		Flags de Enrutamiento-Estado				Interface	#Saltos	Prox. Salto	Precursores	T. de vida
		#	Válido	Válido	No válido	Reparable	Reparando					
A												
1	C	2	SI	Si	No			wlano	1	C		8.380
2	B	0	false	Si	No			wlano	1	B		2.860
B												
1	C	2	SI	Si	No			wlano	1	C		5.380
C												
1	B	0	false	Si	No			wlano	1	B		5.380
2	A	1	SI	Si	No			wlano	1	A		8.860

Figura 19: Tablas de enrutamiento desarrolladas en el simulador AODV. Fuente: Propia

Observando la Figura 19, se pueden apreciar los diferentes campos que tienen las entradas de las tablas de enrutamiento. Es necesario conocer el significado de cada uno de ellos:

- #Entrada: Número de la fila rellenada en una determinada tabla de enrutamiento.
- IP Destino: Dirección IP del nodo destino que alcanza una determinada ruta.
- NSD: En este campo aparece tanto el Número de Secuencia de Destino como una indicación de si este número es válido o no.
- Flags de Enrutamiento-Estado: En este campo, la parte relevante es la que indica si una ruta es válida o inválida (No confundir con NSD válido). Esto es necesario comprenderlo para entender el campo Tiempo de Vida, el cual se explicará a continuación.
- #Saltos: Indica el número de saltos necesarios para que el nodo que tiene una determinada ruta en su tabla de enrutamiento alcance el nodo destino apuntado en dicha ruta.

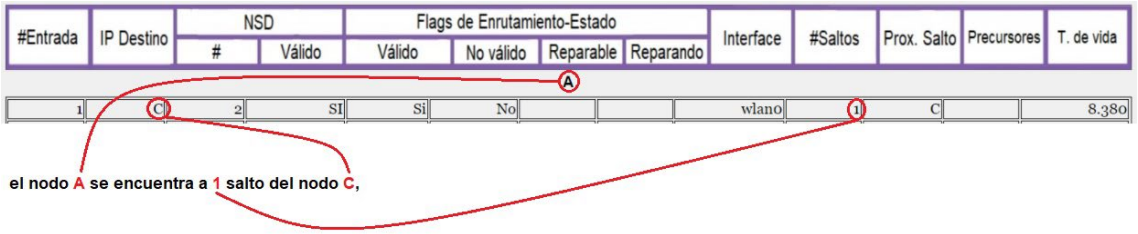


Figura 20: Significado de los campos de las tablas de enrutamiento. Fuente: Propia

- Próximo Salto: Indica cual será el siguiente nodo atravesado para alcanzar el nodo destino de una ruta. (Si el nodo destino se encuentra a un salto, el valor de Próximo Salto es el mismo que IP Destino).
- Precursores: En este campo se apuntan aquellos nodos vecinos (se encuentran a un salto) que pueden usar la ruta. Esto servirá para que, en caso de que un determinado nodo pierda o invalide una ruta, avise a estos precursores para que también eliminen esa ruta de sus tablas de enrutamiento.
- Tiempo de vida: Es un campo que cumple dos funciones. Cuando una ruta es válida, el campo Tiempo de vida expresa el tiempo que le queda a esa ruta antes de ser invalidada. Una vez que ese tiempo llega a 0, se invalida la ruta y el campo Tiempo de vida vuelve a adquirir un tiempo mayor a 0. Cuando este vuelve a alcanzar 0, esa ruta invalidada ya si será eliminada de la tabla de enrutamiento.

Una vez conocido esto, se hablará sobre los tipos de mensajes que AODV utiliza para comunicarse, los cuales se reciben por UDP a través del puerto 654.

5.2.2 *Tipos de mensajes que emplea el protocolo AODV*

- RREQ: Se trata del mensaje de petición de ruta. Este es utilizado por AODV cuando un nodo origen desea encontrar una ruta hacia un nodo destino, ya que dicho nodo origen no posee una ruta válida hacia el nodo destino deseado.
  El formato que presenta este tipo de mensajes es el siguiente:

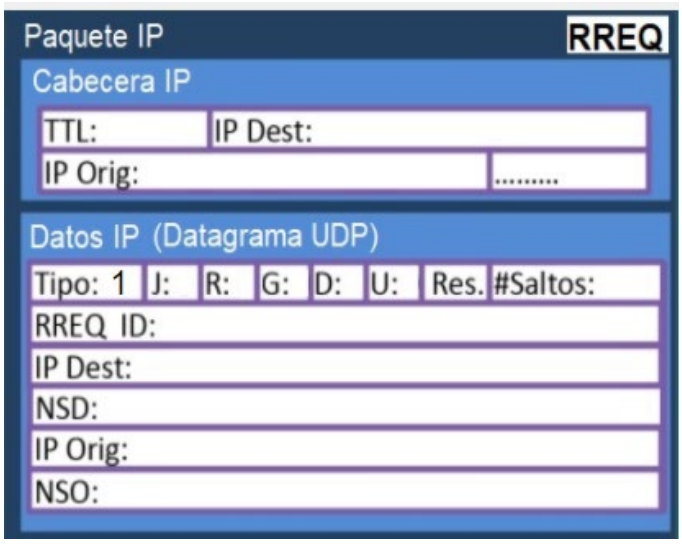


Figura 21: Formato de mensaje de petición de ruta (RREQ).
Fuente: Propia

Todos los tipos de mensajes AODV presentan la misma cabecera IP, indicando en esta su TTL (Time to live), IP destino e IP origen. En cuanto a la parte de los datos IP, si varía el formato dependiendo del mensaje AODV.

En el caso de RREQ, se trata de un mensaje Tipo 1. Los campos J, R, G, D y U están referidos a unos bits que tendrán una u otra determinada función al estar estos activados o desactivados. Los bits G, D y U serán explicados más adelante. El campo número de saltos se refiere al número de saltos dados por el mensaje RREQ desde que ha sido generado por el nodo origen. El campo RREQ ID contiene un identificador numérico (individual en cada nodo) que es incrementado en 1 cada vez que un nodo origina un RREQ. El campo IP origen e IP destino incluyen la dirección IP del nodo que desea encontrar una ruta, el cual originó el RREQ, y la dirección IP del nodo hacia dónde se quiere comunicar. En cuanto a los dos campos restantes, NSO se refiere al Número de secuencia de origen del nodo que origina un RREQ y el NSD es el número de secuencia de destino del nodo a alcanzar.

Es importante tener claro que un mensaje RREQ es originado solo por el nodo origen que quiere la ruta hacia un nodo destino. El resto de nodos reenvían el mismo mensaje RREQ que el nodo origen creó, hasta encontrar al nodo destino.

- RREP: Se trata del mensaje de respuesta de ruta. Este se emplea cuando un mensaje RREQ alcanza al nodo destino indicado en dicho mensaje o bien, alcanza a un nodo intermedio que ya posee una ruta válida hacia el nodo destino.

El formato de estos mensajes es el siguiente:

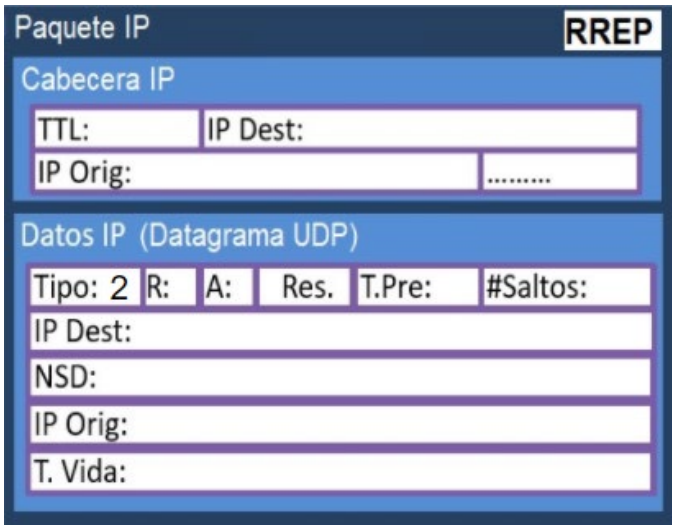


Figura 22: Formato de mensaje de respuesta de ruta (RREP).
Fuente: Propia

Es un mensaje de tipo 2 y sus campos tienen el mismo significado que en el mensaje RREQ, aunque en este aparecen otros como R, A o Tamaño de Prefijo. En este destaca el campo Tiempo de Vida, encargado de indicar a los nodos la duración que tiene una ruta activa o válida para que sea apuntada en su tabla de enrutamiento.

- RERR: Se trata de un mensaje de error de ruta. Este tipo de mensaje se emplea cuando un nodo ha invalidado una ruta hacia un destino y tiene uno o varios nodos precursores asociados a dicha ruta.

Figura 23: Momento en el que se invalida una ruta de una tabla de enrutamiento. Fuente: Propia

En cuanto al formato, este es el que presenta un mensaje RERR:

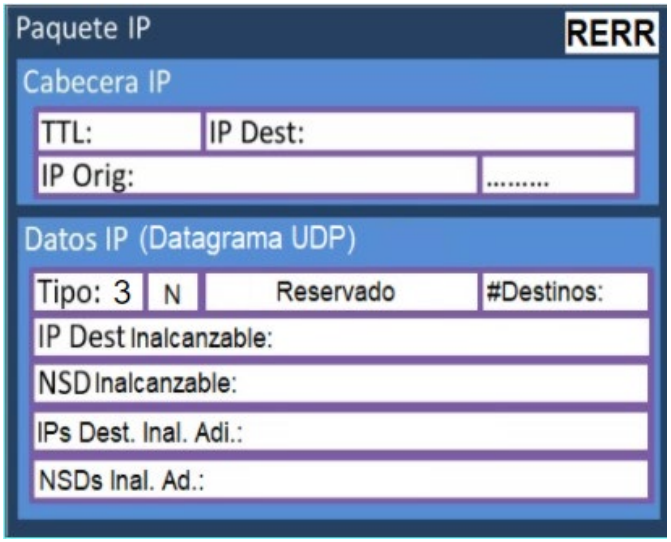


Figura 24: Formato de mensaje de error de ruta (RERR). Fuente: Propia

Este es un mensaje de tipo 3. El campo #Destinos de un mensaje RERR se refiere al número de destinos o rutas que han quedado invalidadas o ya no son alcanzables por el nodo que emite el RERR. IP destino inalcanzable se trata de la dirección IP del nodo al que ya no tiene acceso el nodo que emitió el RERR. NSD inalcanzable se trata del número de secuencia de destino que tiene asociado la dirección IP del nodo apuntado en el campo IP

destino inalcanzable. El resto de campos, se refiere a IPs y NSDs adicionales a los ya proporcionados, por si hubiese varios destinos inalcanzables.

5.2.3 ¿Cómo descubre rutas el protocolo AODV?

En el siguiente diagrama de flujo se puede ver cómo actúa AODV cuando un nodo desea obtener una ruta hacia otro nodo:

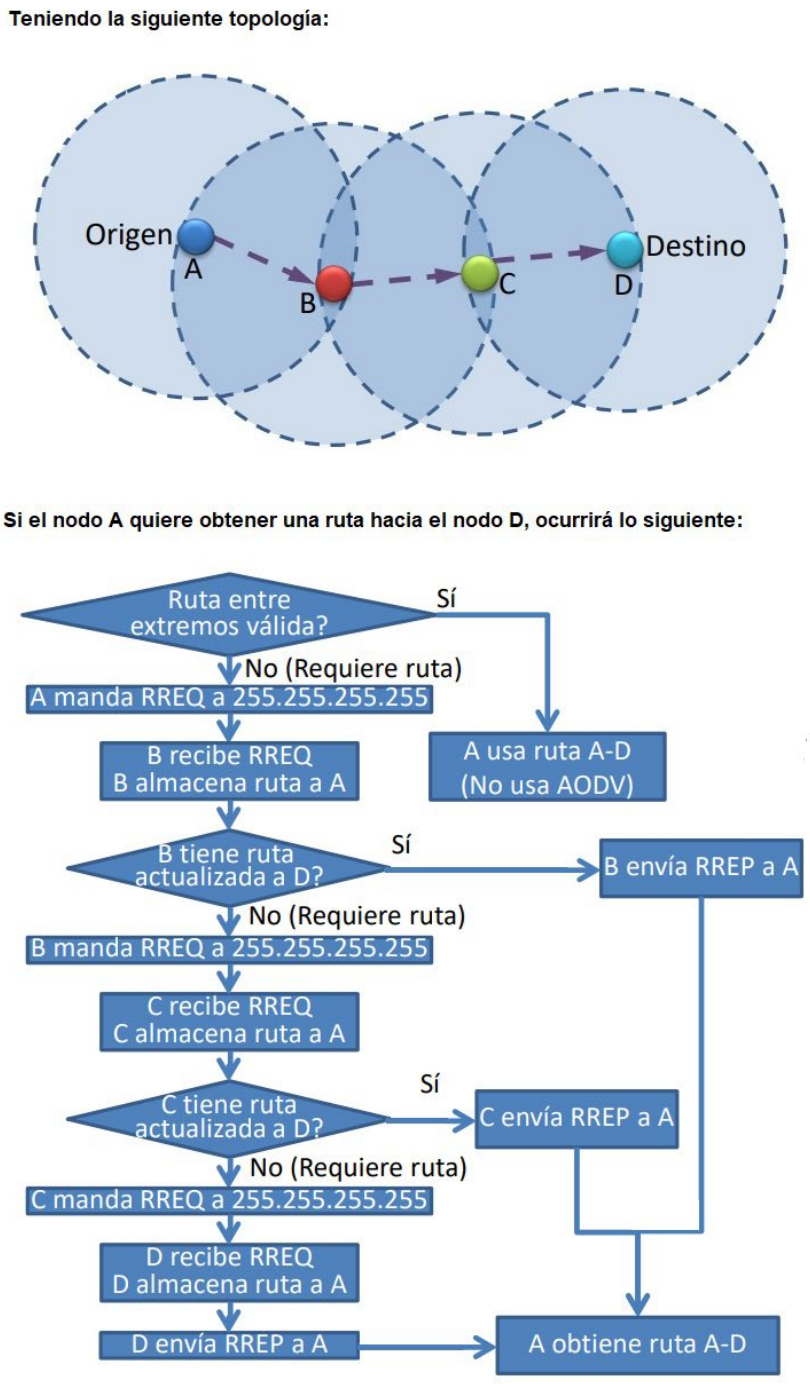


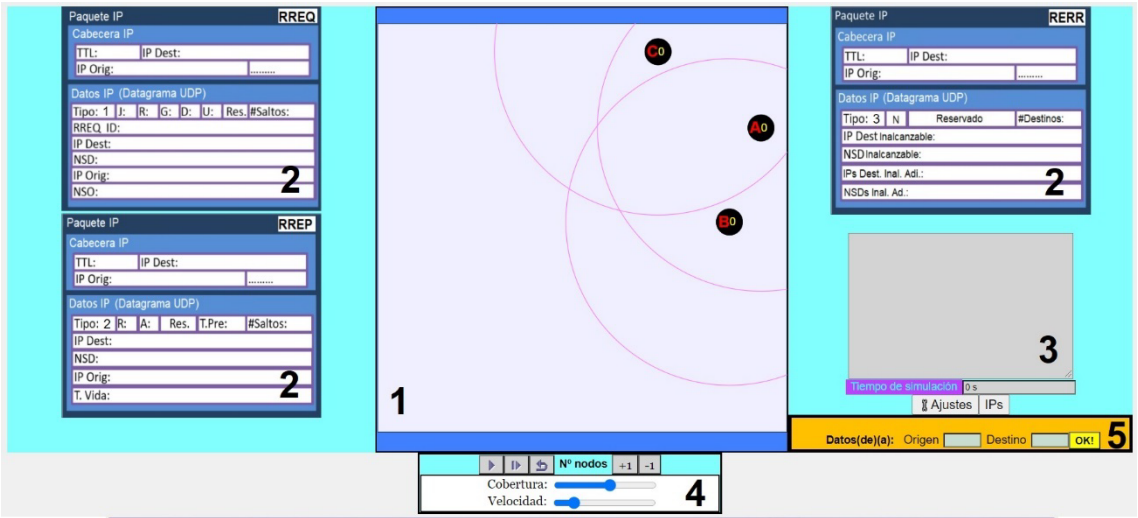
Figura 25: ¿Como descubren los nodos rutas hacia otros nodos? Fuente: https://dv.ujaen.es/goto_docencia_file_533992_download.html

5.3 Desarrollo del simulador AODV

Una vez conocida la herramienta empleada para desarrollar el simulador AODV, así como los componentes y características del protocolo, se explicará paso a paso como se ha desarrollado dicho simulador, exponiendo los diferentes procedimientos realizados y funciones creadas para recrear el comportamiento del protocolo AODV ante diferentes situaciones.

5.3.1 Descripción y construcción de la interfaz del simulador AODV

En primer lugar, es importante mostrar la interfaz propuesta para este simulador:



#Entrada	IP Destino	NSD		Flags de Enrutamiento-Estado				Interface	#Saltos	Prox. Salto	Precursores	T. de vida
		#	Válido	Válido	No válido	Reparable	Reparando					
A												
1	C	1	SI	Si	No			wlan0	1	C	B	5.610
2	B	1	SI	Si	No			wlan0	1	B	C	8.610
B												
1	A	0	false	Si	No			wlan0	1	A		2.610
2	C	1	SI	Si	No			wlan0	2	A		5.610
C												
1	A	0	false	Si	No			wlan0	1	A		2.610
2	B	1	SI	Si	No			wlan0	2	A		8.610

6

Figura 26: Interfaz del simulador AODV. Fuente: Propia

Como se puede apreciar, aparecen en diversos puntos de la interfaz, números que van del 1 al 6, estos conforman las partes más importantes del simulador. En primer lugar, se explicará qué significan cada uno de estos puntos y posteriormente, se describirá cómo se han desarrollado:

1- Se trata de un espacio donde los nodos simulados tienen total libertad de movimiento, siempre sin salirse de dicho espacio. Además, dichos nodos pueden ser arrastrados manualmente.
2- Imágenes de los diferentes formatos de paquetes IP que el protocolo AODV emite cuando se encuentra en funcionamiento. Estos son el RREQ, RREP y RERR.
3- Espacio de narración. Esta parte de la interfaz es la encargada de describir todo lo que está pasando durante la simulación del protocolo AODV.
4- Es la parte que permite a los usuarios controlar la simulación. Desde aquí se puede iniciar, pausar, reiniciar, e incluso añadir nodos nuevos a la simulación y cambiar parámetros como la velocidad o la cobertura de los nodos.
5- Espacio de comunicación. Permite la introducción de la letra del nodo origen y el nodo destino para así simular los procedimientos que AODV realiza con el fin de que estos nodos puedan establecer una comunicación.
6- Tablas de enrutamiento. Se tendrá acceso a todas las tablas de enrutamiento de todos los nodos que están siendo simulados. Aquí se podrá ver cómo evolucionan cada una de las rutas que poseen estos nodos.

Aunque no aparezca numerado, también es importante destacar los botones que aparecen sobre el espacio de comunicación.

Figura 27: Botón de "Ajuste" e "IPs".
Fuente: Propia

El menú “Ajustes” permite modificar algunos parámetros de la simulación, como mostrar más o menos información, activar/desactivar bit G/D, los cuales serán desarrollados a continuación, o cambiar el idioma del simulador. Además, pulsando sobre el botón “IPs” se podrá ver qué dirección IP tiene asociada cada una de las letras que identifican a cada nodo.

Una vez descritas las diversas partes de la interfaz gráfica que posee el simulador AODV, se describirán los pasos que se han llevado a cabo para construirla:

1- Definición de las variables y elementos HTMLView para conseguir mostrar el espacio donde los nodos se mueven, así como los propios nodos y los elementos con los que estos interactúan.

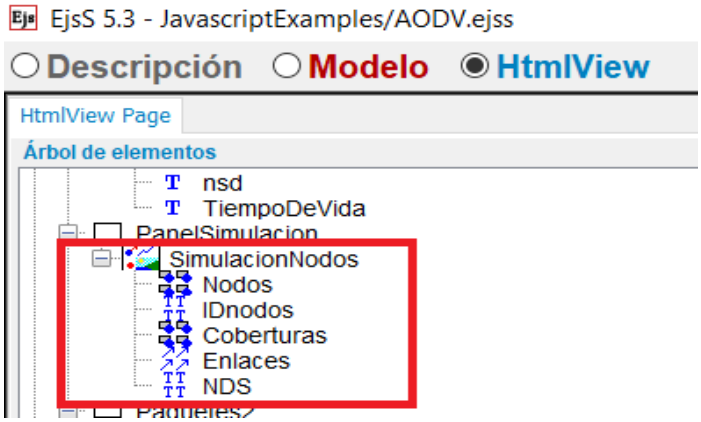


Figura 28: Definición de elementos en HTMLView para mostrar nodos en la simulación. Fuente: Propia

Como resultado, la interfaz proporciona lo siguiente:

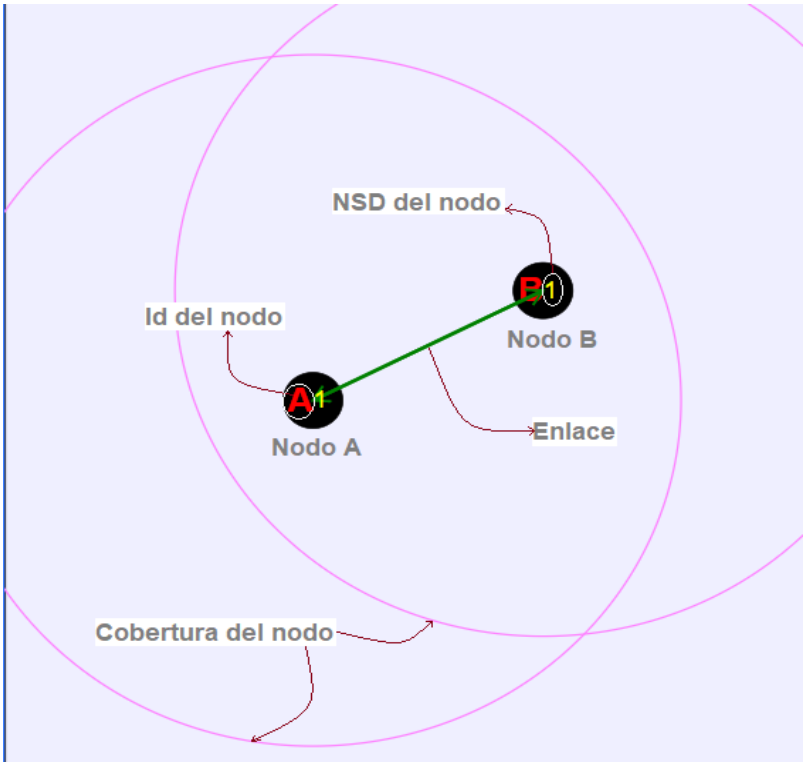


Figura 29: Componentes del espacio donde se simula el movimiento de los nodos. Fuente: Propia

2- Adición de las imágenes que representan los diferentes tipos de formatos de paquetes IP que AODV utiliza para sus diferentes tareas.

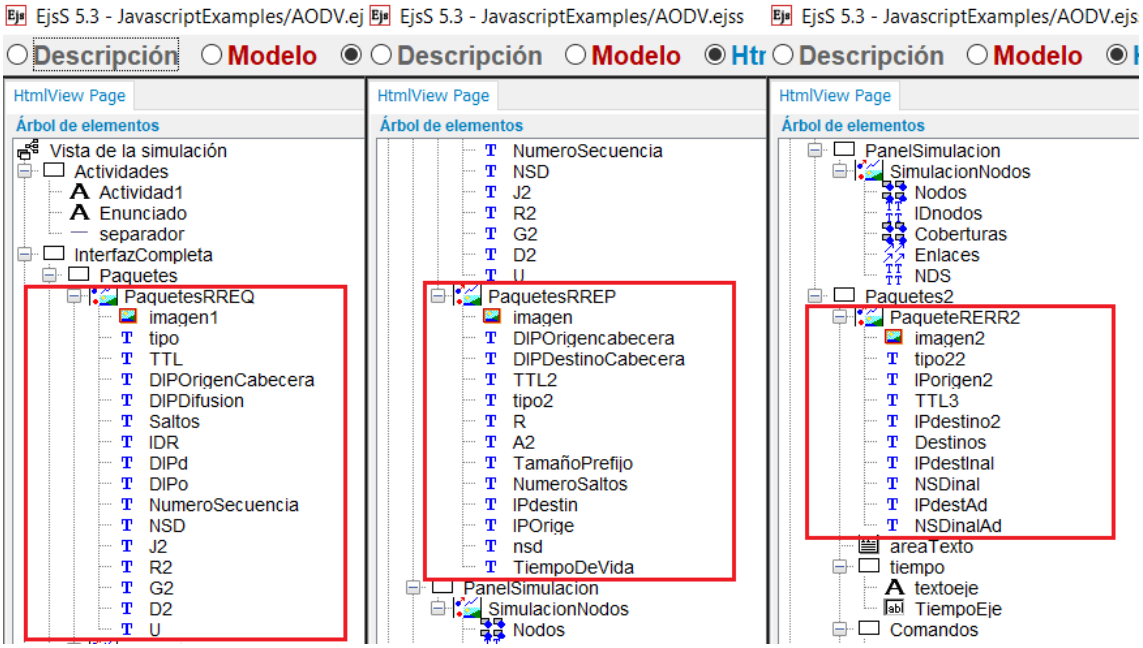


Figura 30: Definición de los diferentes elementos que permiten mostrar el formato de los mensajes empleados por AODV en la simulación. Fuente: Propia

Como se puede observar, existe para cada uno de los tipos de mensajes (RREQ, RREP y RERR) un elemento Imagen y varios elementos de Texto. La imagen se trata solamente del Paquete IP en cuestión, sin campos rellenos, mientras que los elementos de texto permiten mostrar sobre dicha imagen la información que contendría un determinado paquete IP.

Para cargar estas imágenes, únicamente se debe incluir la URL de cada imagen en el elemento Imagen, tal y como aparece a continuación

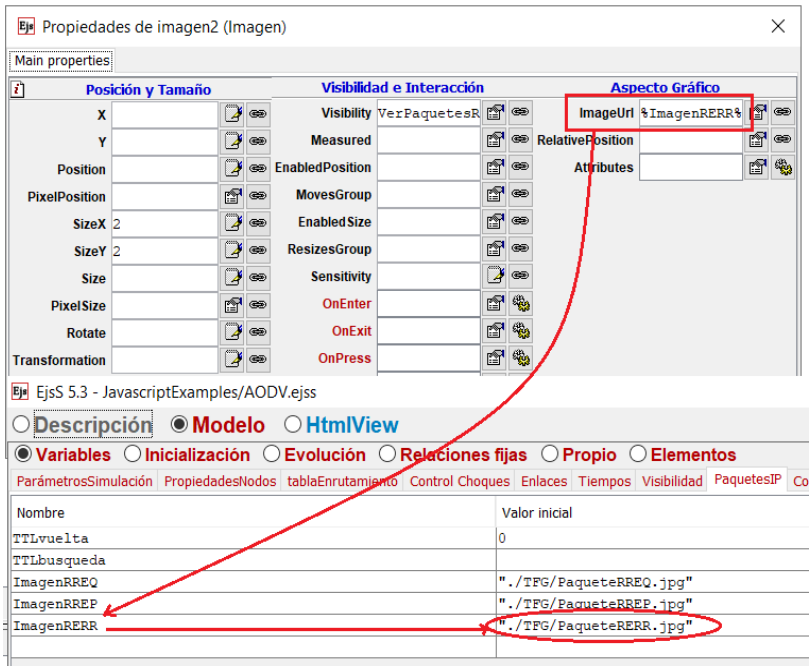


Figura 31: ¿Como se añaden imágenes a la simulación? Fuente: Propia

Gracias a esto, se consigue ver la siguiente parte de la interfaz de simulación:

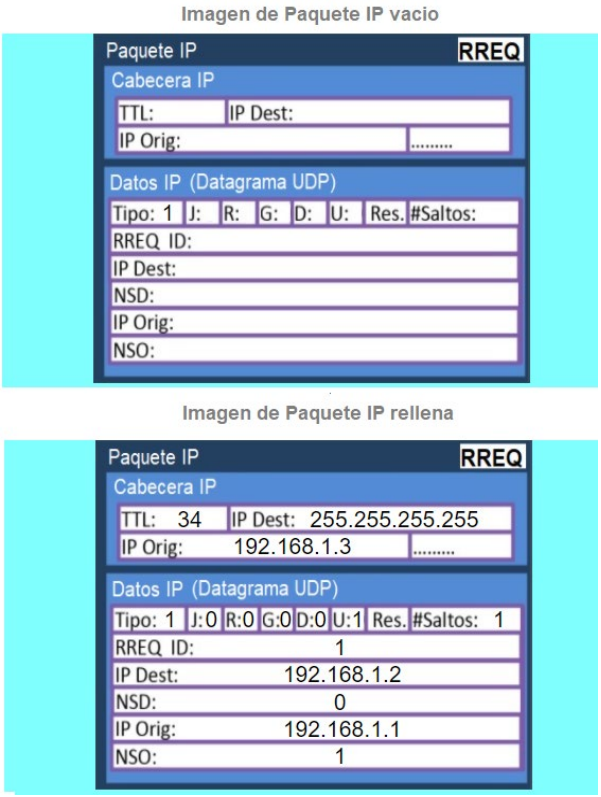


Figura 32: Estados que pueden presentar las imágenes que representan el formato de los mensajes empleados por AODV. Fuente: Propia

3- Inserción del espacio de narración a la interfaz mediante la definición el siguiente elemento en la pestaña HTMLView.

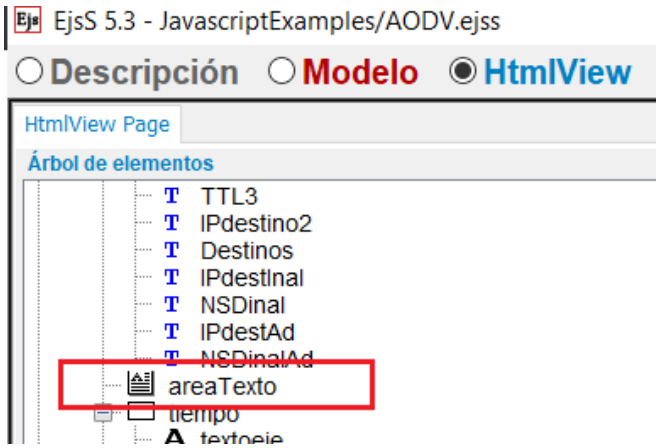


Figura 33: Definición del espacio de narración. Fuente: Propia

Lo que se obtiene es una simple área de texto capaz de mostrar durante la ejecución de la simulación que está realizando el simulador en cada momento. La variable String empleada para rellenar este campo de texto se llama *Accion* y esta varía durante toda la simulación para describir las diversas tareas que AODV puede realizar.

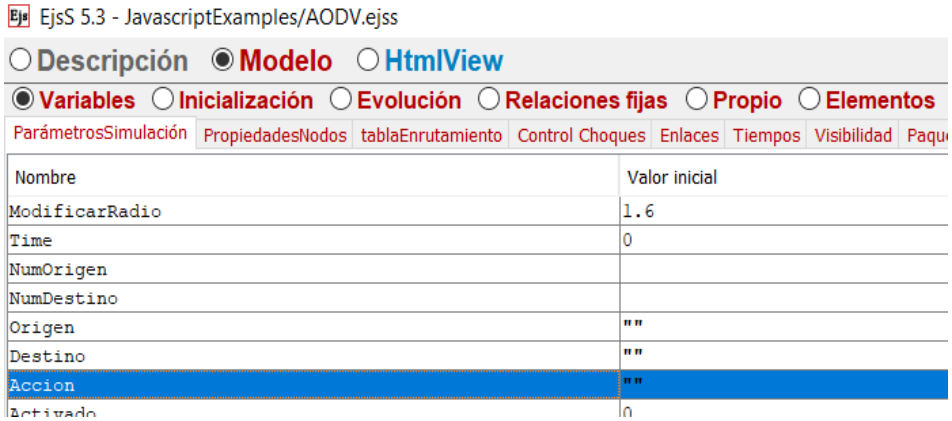


Figura 34: Definición de la variable empleada para describir los procesos desarrollados durante la simulación. Fuente: Propia

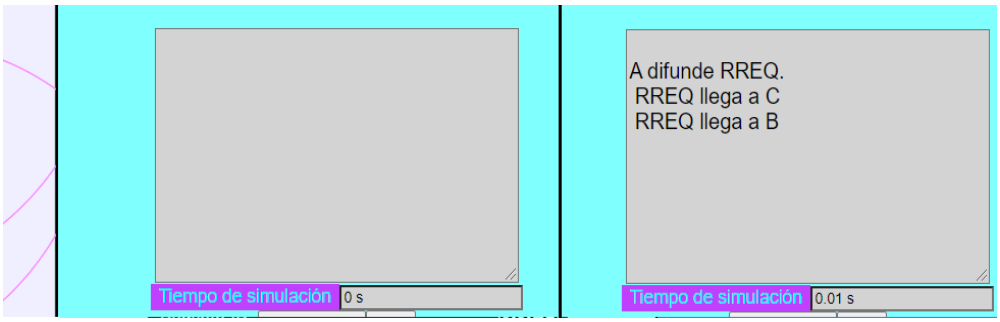


Figura 35: Ejemplo del funcionamiento del espacio de narración. Fuente: Propia

4- Desarrollo de un panel de control para la simulación definiendo los siguientes elementos.

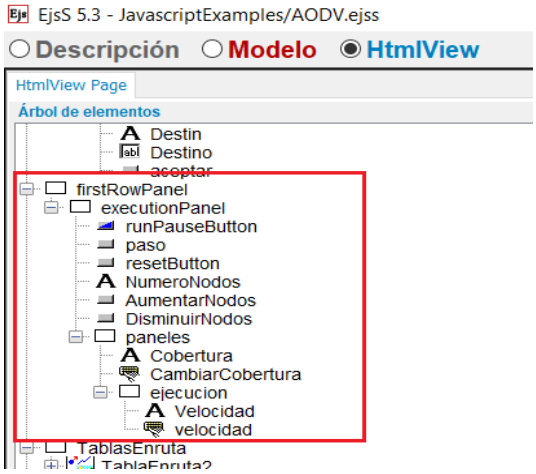


Figura 36: Definición del panel de control del simulador. Fuente: Propia

Esto permite visualizar el siguiente panel en la interfaz:

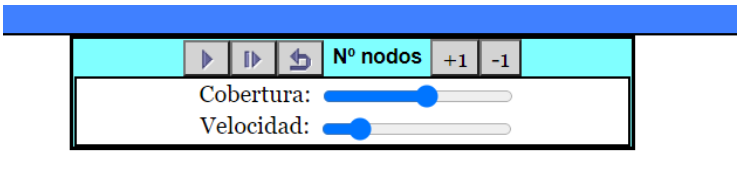


Figura 37: Como se muestra el panel de control en la interfaz del simulador. Fuente: Propia

A través del mismo, el usuario podrá iniciar, pausar y reiniciar la simulación. Además, permitirá añadir/borrar nodos, así como modificar la cobertura y velocidad de movilidad de los mismos.

5- Desarrollo de un panel que permite al usuario introducir un nodo origen y destino para así ver los pasos que AODV realiza con el fin de establecer rutas entre ellos. La parte gráfica de se consigue definiendo los siguientes elementos.

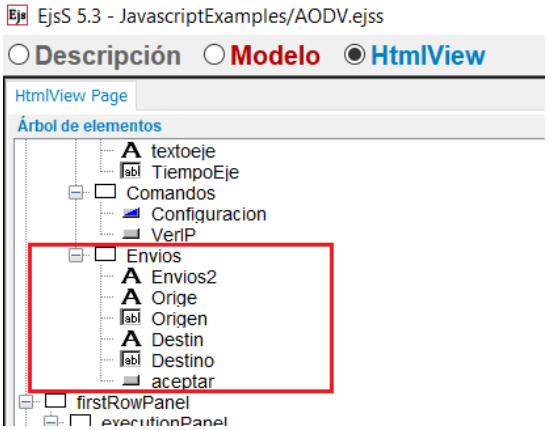


Figura 38: Definición del espacio de comunicación. Fuente: Propia

Sencillamente, se han definido dos elementos Input que nos permitirán introducir las letras de los nodos, así como un botón que realizará una tarea en base a las letras introducidas.

La interfaz mostraría lo siguiente:

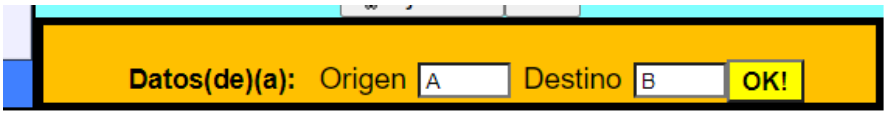


Figura 39: Aspecto del espacio de comunicación en la interfaz del simulador. Fuente: Propia

6- Definición de las diferentes tablas de enrutamiento. Como cada nodo tiene una tabla de encaminamiento propia, se han tenido que definir 10 tablas diferentes, ya que el límite de nodos simulables en este simulador es de 10. (Desde la A, hasta la J).

Para mostrar estas tablas en la interfaz, es necesario definir los siguientes elementos:

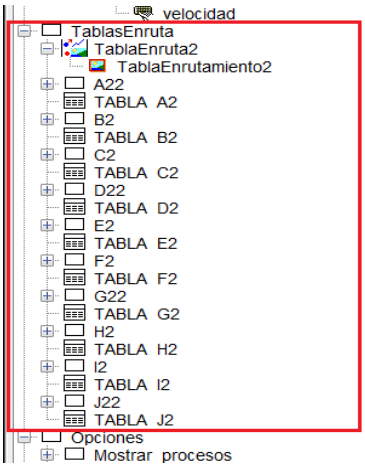


Figura 40: Definición de los elementos que permiten visualizar las tablas de enrutamiento de cada nodo. Fuente: Propia

Aunque gráficamente, solo se puedan ver las tablas de encaminamiento de aquellos nodos que están siendo simulados, el resto de tablas también se encuentran en la interfaz, lo que ocurre es que permanecen ocultas hasta que su nodo es añadido a la simulación.

Aquí también se cuenta con una imagen situada sobre todas las tablas que permite conocer cuál es cada campo relleno en estas tablas.

Definido todo esto, la interfaz nos muestra lo siguiente:

#Entrada	IP Destino	NSD		Flags de Enrutamiento-Estado				Interface	#Saltos	Prox. Salto	Precursores	T. de vida
		#	Válido	Válido	No válido	Reparable	Reparando					
A												
1	C	0	false	Si	No			wlano	1	C		3.000
B												
1	C	0	false	Si	No			wlano	1	C		3.000
2	A	1	SI	Si	No			wlano	2	C		3.000
C												
1	A	1	SI	Si	No			wlano	1	A		3.000

Figura 41: Aspecto de las tablas de enrutamiento mostradas en el simulador. Fuente: Propia

En cuanto al botón “Ajustes”, primero es necesario definir los siguientes elementos en la pestaña HTMLView:

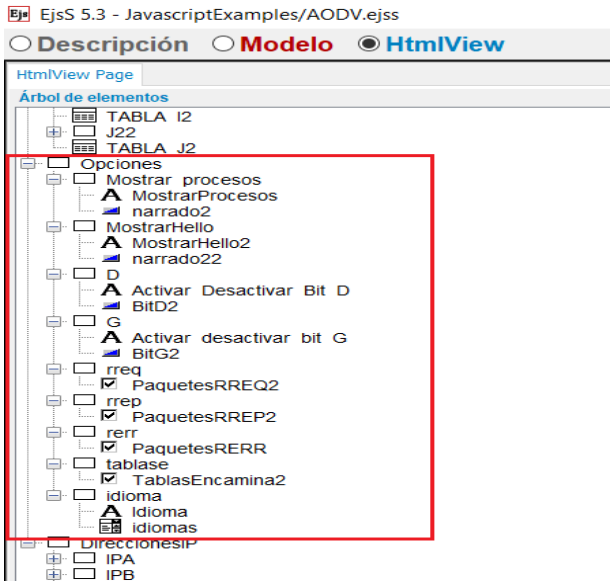


Figura 42: Definición del menú de ajustes. Fuente: Propia

Gracias a esto, una vez que el usuario pulse sobre dicho botón, obtendrá el siguiente menú emergente en la interfaz:

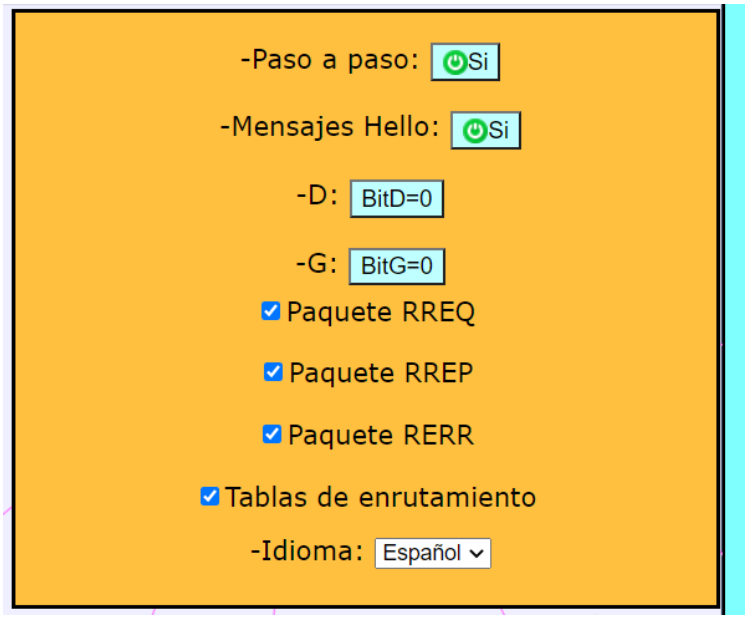


Figura 43: Aspecto que presenta el menú de ajustes. Fuente: Propia

Este menú permite modificar diferentes aspectos de la simulación, tales como:

- Decidir si se quiere o no que la simulación se pause con cada una de las tareas que realiza el protocolo AODV para que sean explicadas.
- Decidir si se quiere que la simulación se pause con cada mensaje HELLO que los nodos emiten.
- Activar/Desactivar bit G/D
- Mostrar/Ocultar diferentes partes de la interfaz, como las tablas de enrutamiento o las imágenes de paquetes IP utilizadas para simular los tipos de mensajes que AODV emite.
- Modificar el idioma de la simulación. Se encuentra disponible en inglés y español.

Otro aspecto importante a conocer de la interfaz, es que los nodos pueden presentar tres colores diferentes durante el desarrollo de la simulación:

Figura 44: Colores que los nodos pueden tener durante la simulación. Fuente: Propia

Sabiendo como se ha construido la interfaz gráfica de nuestro simulador, se explicará cómo se han desarrollado las diferentes funcionalidades que es capaz de desempeñar para recrear el comportamiento del protocolo AODV.

5.3.2 Inicialización de la simulación

Cuando un usuario acaba de acceder al simulador, se inicializan una serie de valores necesarios para comenzar la ejecución de la simulación.

Esta inicialización consta de tres subapartados dentro del apartado “Inicialización” de la herramienta EJS, como se puede apreciar en la figura inferior.

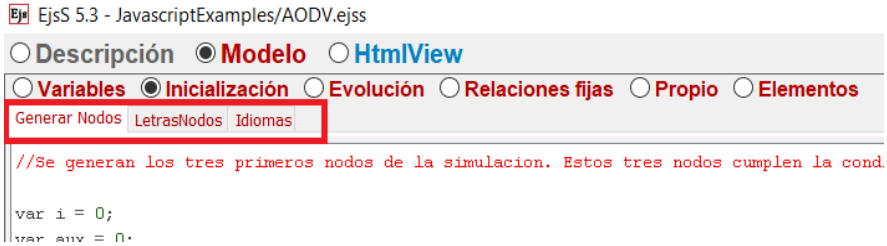


Figura 45: Apartados desarrollados para la inicialización de la simulación. Fuente: Propia

- Generar Nodos:

Este subapartado es el encargado de asignar las posiciones y velocidades por defecto que cada uno de los nodos va a tener al principio de la simulación en el espacio donde estos se mueven. Al comienzo de la simulación aparecen tres nodos, ya que así está definido en la variable *NumeroNodos*.

La condición que establece este subapartado para asignar las posiciones de los nodos, es que ningún nodo aparezca solapado sobre otro, ni fuera del plano.

Para situar un nodo, previamente comprueba si la posición para dicho nodo ya está ocupada. Si no lo está, se establece esa posición, una velocidad aleatoria y un color de nodo (En este caso es el negro). Para saber si la posición de un nodo está solapada sobre otro nodo, se emplea la función *distancia.*

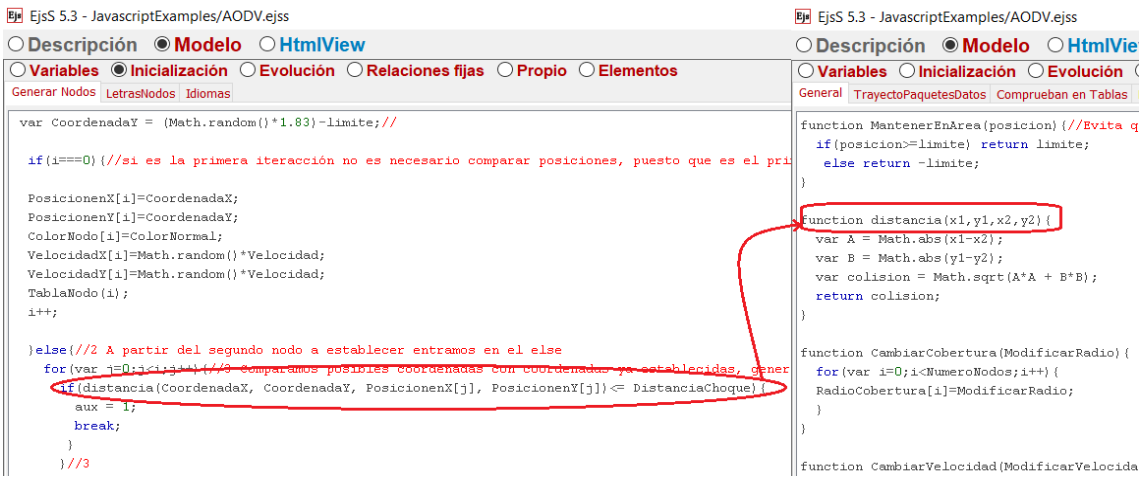


Figura 46: Definición de la función distancia, encargada de calcular longitudes entre nodos. Fuente: Propia

- LetrasNodos:

  Este subapartado se encarga de asignar a cada nodo una letra y una dirección IP.

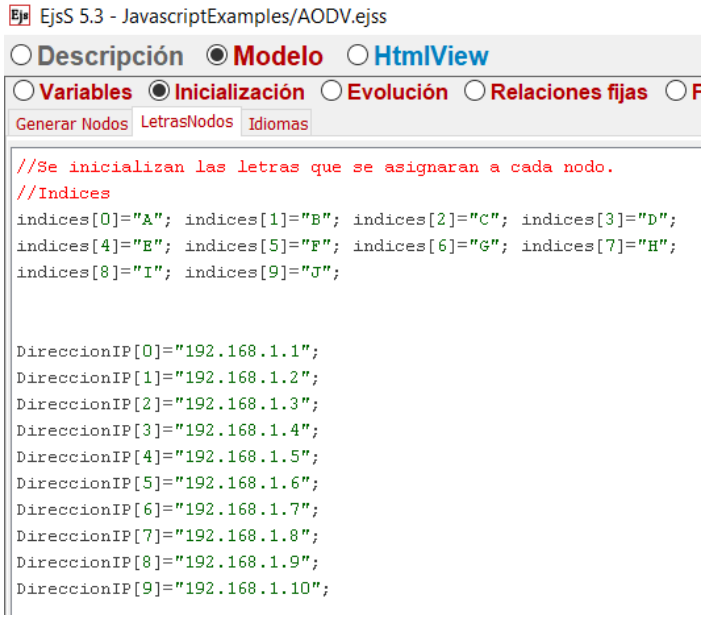


Figura 47: Definición de las variables que permiten diferenciar a los nodos. Fuente: Propia

- Idiomas:

  Establece las dos opciones de idiomas disponibles que aparecen en el menú de ajustes de la simulación

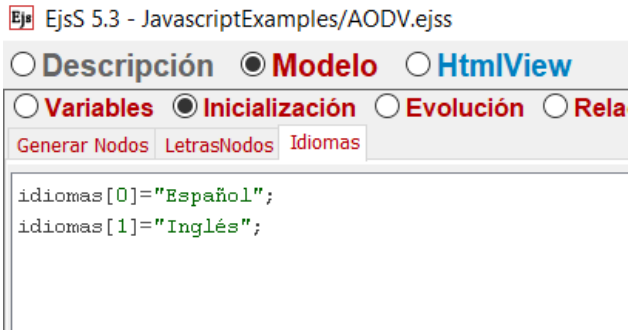


Figura 48: Definición de los idiomas seleccionables en la simulación. Fuente: Propia

Una vez es inicializada la simulación, veremos cómo se desarrolla el resto del software para hacer que se comporte como el protocolo AODV.

5.3.3 Implementación de las funcionalidades del simulador

En el subapartado “Ciclo”, el cual se encuentra dentro del apartado “Evolución” de la herramienta EJS, aparece definido que ocurre en cada ciclo de la simulación dependiendo de diversas acciones y condiciones, permitiendo recrear AODV en todas las posibles situaciones.

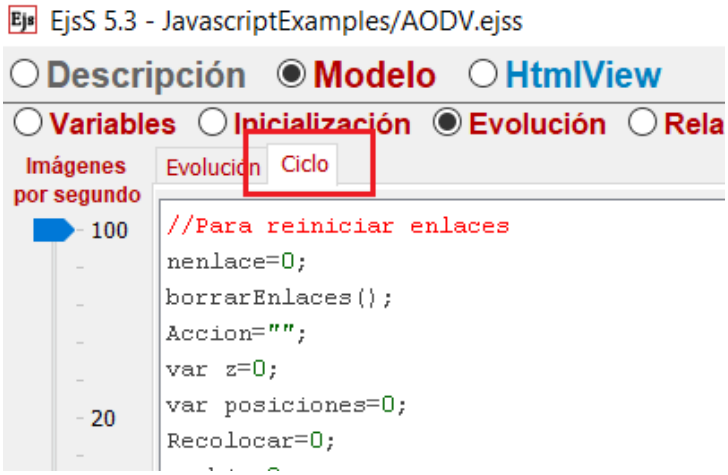


Figura 49: Implementación del apartado Ciclo, encargado de definir que ocurre en cada ciclo que se desarrolla en la simulación.
Fuente: Propia

5.3.3.1 Gestión de la movilidad de los nodos

Cuando se pulse el botón “PLAY” cada uno de los nodos se desplazarán independientemente por el plano, además se controla la colisión entre nodos, haciendo que estos reboten cuando se aproximan demasiado entre ellos. También se mantienen los nodos dentro del plano, haciendo que estos reboten cuando llegan a los bordes.

Esto se consigue desarrollando lo siguiente:

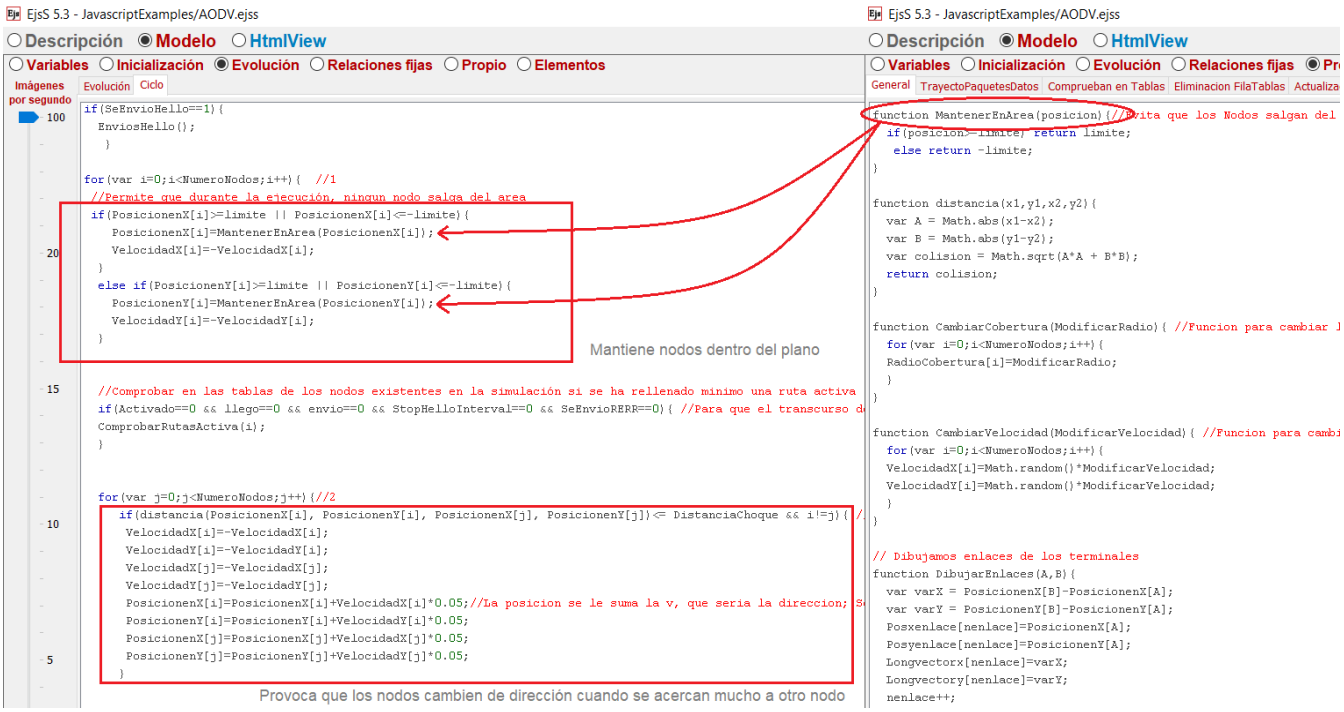


Figura 50: Implementación de métodos que permiten controlar las trayectorias de los nodos. Fuente: Propia

5.3.3.2 Envío de mensajes desde un nodo origen hacia un nodo destino

Cuando se quiere observar que pasos realiza AODV si un nodo origen desean enviar datos a un nodo destino, se introduce la letra del nodo origen y la letra del nodo destino en el espacio de comunicación, como se ve a continuación:

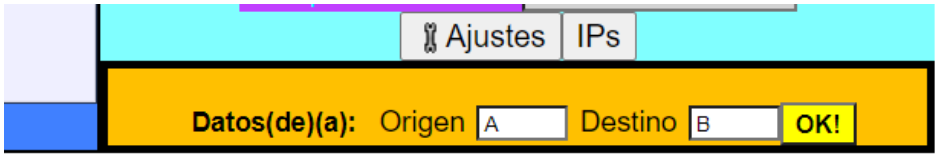


Figura 51: Utilización del espacio de comunicación. Fuente: Propia

Una vez se introducen dichas variables y se pulsa sobre el botón “OK!”, se llamará a la función *On*.

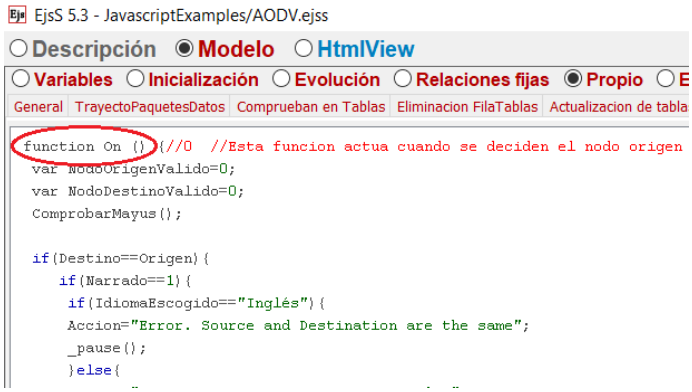


Figura 52: Definición de la función On. Fuente: Propia

Esta función se encarga, en primer lugar, de comprobar si los valores introducidos son válidos (Es válida la introducción de letras desde la A, hasta la J, tanto en minúsculas como en mayúsculas y no se puede establecer la misma letra como nodo origen y destino). Si dichos valores son correctos, se entra en la función *VolverAEmpezar* donde se establecen una serie de variables como por defecto para comenzar una comunicación entre dos nodos.

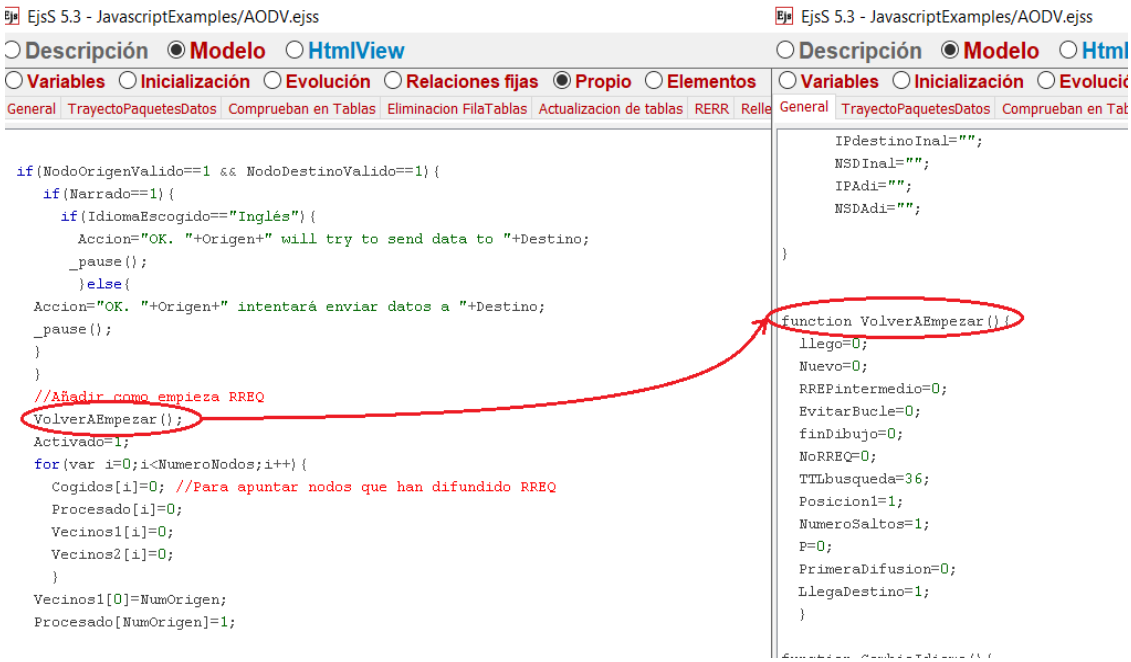


Figura 53: Definición de la función VolverAEmpezar. Fuente: Propia

Seguido a esta función, se puede ver a la variable *Activado*. Cuando esta se establece a 1, permite entrar en la fase de descubrimiento de ruta del protocolo AODV dentro del apartado “Evolución”.

Además, para entrar a esta fase, es necesario conocer las siguientes cuatro variables que se van a utilizar:

```
Accion="OK. "+Origen+" intentará enviar datos a "+Destino;
_pause();
}
}
//Añadir como empieza RREQ
VolverAEmpezar();
Activado=1;
for(var i=0;i<NumeroNodos;i++){
  Cogidos[i]=0; //Para apuntar nodos que han difundido RREQ
  Procesado[i]=0;
  Vecinos1[i]=0;
  Vecinos2[i]=0;
  }
Vecinos1[0]=NumOrigen;
Procesado[NumOrigen]=1;


}else if(NodoOrigenValido!=1 || NodoDestinoValido!=1){ //Si no son nodos validos
  if(Narrado==1){
    if(IdiomaEscogido=="Inglés"){
```

Figura 54: Definición de los vectores Cogidos, Procesado, Vecinos1, Vecinos2. Fuente: Propia

- *Cogidos*:

Se trata de un vector cuya función es la de apuntar que nodos han difundido un mensaje RREQ en una misma fase de descubrimiento de ruta. La teoría de AODV indica que, cuando un nodo difunde un RREQ en una fase de descubrimiento de ruta, ya no puede volver a difundir un RREQ durante esa misma fase. Para conseguir que esto se cumpla, cuando un nodo difunda un RREQ, se establece el valor 1 en su posición dentro del vector *Cogidos*, evitando con esto que el nodo vuelva a difundir un RREQ en esa misma fase. (Los nodos que pueden difundir RREQ en una fase de descubrimiento de ruta tendrán el valor 0 dentro de este vector).

- *Procesado:*

Se utiliza para controlar que cuando un nodo haya procesado (recibido) un mensaje RREQ, no vuelva a procesar otro de la misma difusión (RREQ ID y NSD iguales que el que recibió anteriormente). Funciona de manera similar al vector *Cogidos*, estableciendo un 1 en la posición del nodo que acaba de recibir un RREQ.

- *Vecinos1:*

Es un vector que registra todos los nodos que pertenecen a un mismo salto respecto a un nodo escogido como origen durante la fase de descubrimiento de rutas.

- *Vecinos2*:

Es un vector que almacena los nodos vecinos a los que *vecinos1* envía mensajes RREQ durante la fase de descubrimiento, es decir, *Vecinos2* contendría los nodos del siguiente salto a *Vecinos1*

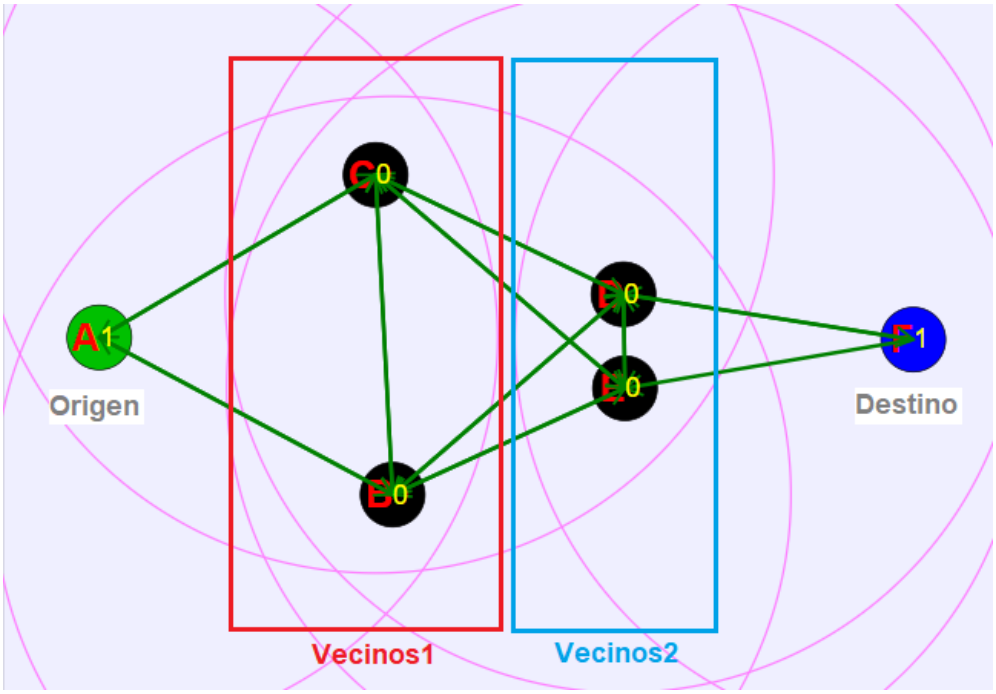


Figura 55: Representación de los nodos almacenados por los vectores Vecinos1 y Vecinos2 en el proceso de descubrimiento de ruta. Fuente: Propia

El funcionamiento de todos estos vectores se comprenderá mejor durante la explicación de la función dedicada a la difusión de mensajes RREQ.

5.3.3.2.1 Difusión de mensajes RREQ

Como se ha descrito anteriormente, al ser *Activado*=1, la simulación llega a la siguiente zona del código:

Figura 56: Parte del código encargada de la comprobación de rutas y difusión de mensajes RREQ. Fuente: Propia

Esta parte se encarga de analizar si los nodos que están siendo simulados poseen o no rutas válidas entre el nodo origen y destino indicado previamente.

En el caso de que la simulación acabe de ser ejecutada, las tablas de enrutamiento de todos los nodos se encontrarán vacías, presentando este aspecto:

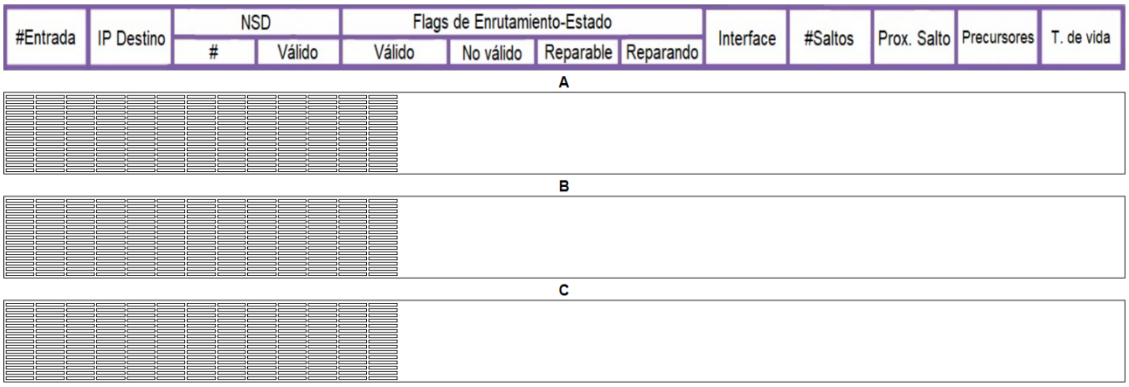


Figura 57: Aspecto que presentan las tablas de enrutamiento cuando todas están vacías. Fuente: Propia

Debido a que no existen aún rutas válidas entre los nodos origen y destino, deben establecerse. Para ello, se realiza lo siguiente:

En primer lugar, cuando el nodo origen genera un RREQ, debe incrementar en 1 su NSD propio además de incrementar en 1 el IDRREQ que irá incluido en el mensaje.

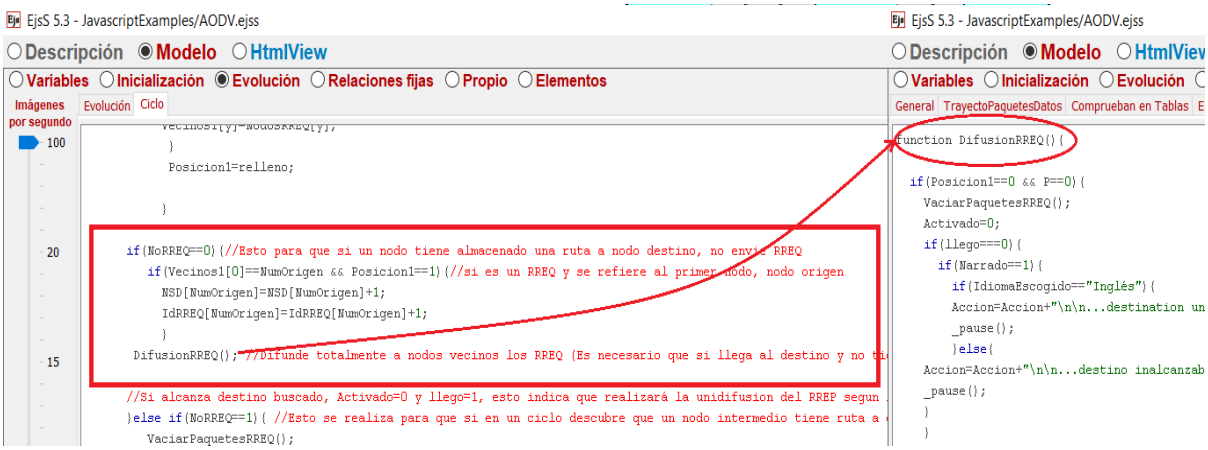


Figura 58: Función encargada de la difusión de mensajes RREQ. Fuente: Propia

Tras esto, se llama a la función *DifusionRREQ*.

Esta función se encarga de:

- Comprobar si el nodo que tiene que difundir el RREQ en ese ciclo ya lo difundió o no.
- Nos muestra sobre la imagen de Paquete IP RREQ como se rellenan los campos del mensaje
- Llama a la función *tablitas*, encargada de rellenar las tablas de enrutamiento.

Es importante saber que en un ciclo de ejecución solo se simula la difusión del mensaje RREQ de un solo nodo. Esto nos permite ver paso a paso (ciclo a ciclo) cómo se van difundiendo los mensajes.

Para comprobar si un nodo ya ha difundido o no, como dijimos anteriormente, se emplea el vector *Cogidos*, observando si para ese nodo su valor es 1 o 0.

Si este no ha difundido, el simulador nos muestra cómo sería el formato del mensaje RREQ que dicho nodo difunde.

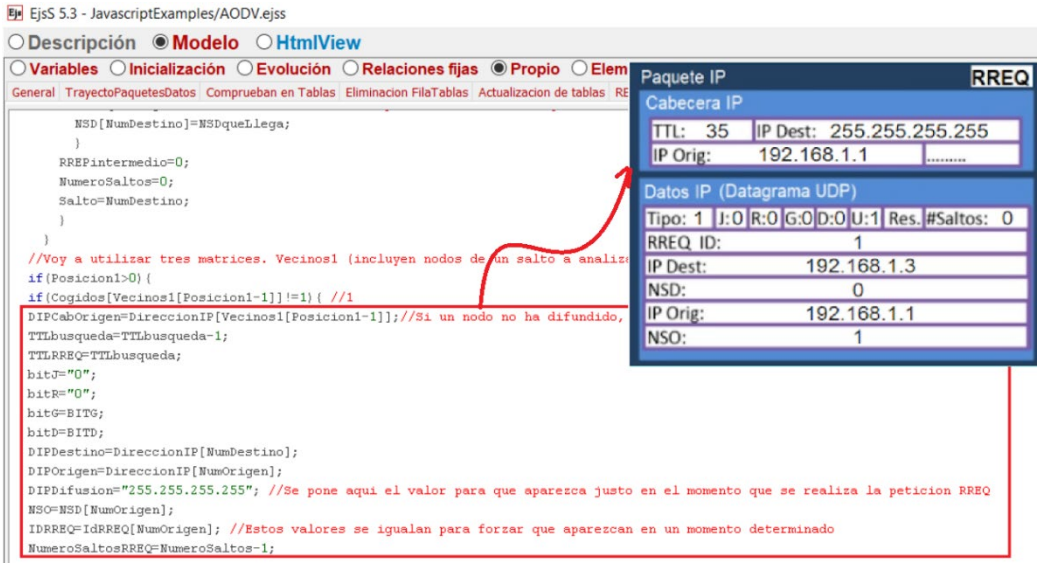


Figura 59: Ejemplo del formato de mensaje RREQ que el protocolo AODV emplea para solicitar una ruta. Fuente: Propia

Al mismo tiempo, se comprueba hacía que nodos vecinos podrá llegar el mensaje que se emite en base a la cobertura establecida en dichos nodos, empleando la función *distancia* descrita anteriormente. Se mostrará esto en el espacio de narración.

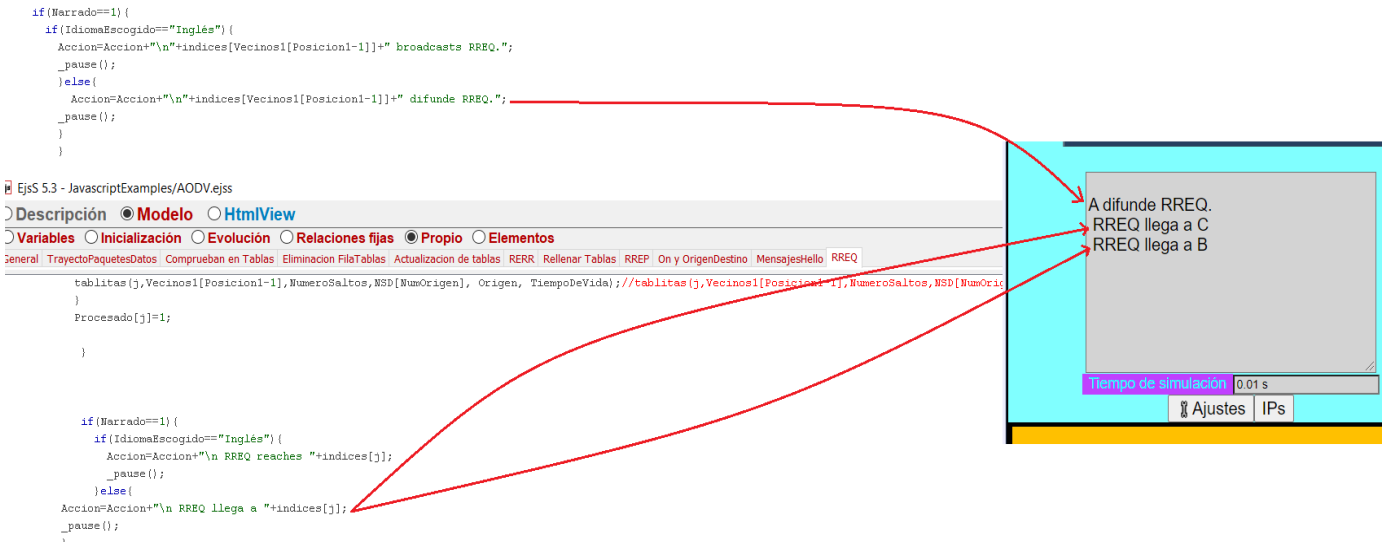


Figura 60: Mensaje mostrado en el espacio de narración cuando se simula el envío de un mensaje RREQ. Fuente: Propia

En este punto, es importante comentar el significado del campo Bit U:

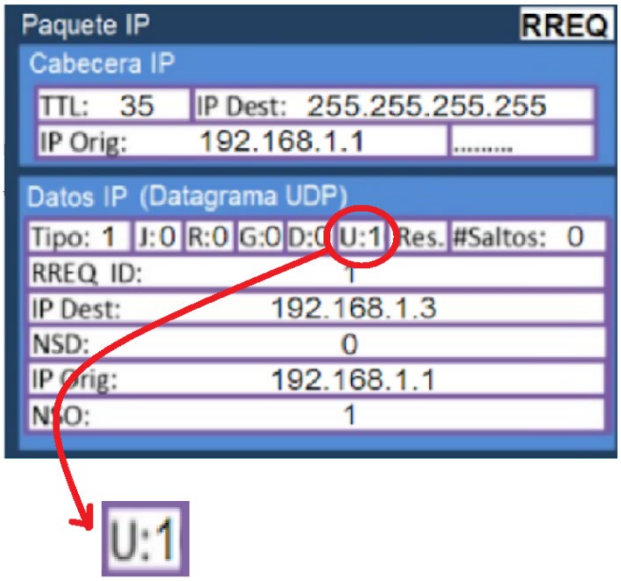


Figura 61: Localización del Bit U dentro de un mensaje RREQ. Fuente: Propia

Este campo aparece en un mensaje RREQ con el valor 1 si el nodo que generó o reenvió dicho mensaje RREQ desconoce el número de secuencia del nodo destino al que se quiere conseguir una ruta. Cuando el valor es 0, significa que si se conoce el NSD de dicho nodo.

En esta fase de descubrimiento de rutas, los vectores explicados anteriormente, *Vecinos1* y *Vecinos2* juegan un rol muy importante: *Vecinos1* almacena aquellos nodos de un mismo salto con respecto al nodo origen a los que les toca difundir. Este vector se va recorriendo desde el final, hasta llegar al principio del mismo, haciendo que todos los nodos que tiene registrados difundan su RREQ. Durante esta difusión, los nodos que reciben estos mensajes se van almacenando en *Vecinos2*. Cuando el último nodo registrado en *Vecinos1* termina de difundir su mensaje RREQ, el vector *Vecinos2* se vuelca sobre *Vecinos1*, lo que permite que ahora difundan los nodos del siguiente salto.

Figura 62: Como evolucionan Vecinos1 y Vecinos2 hasta completar la difusión de mensajes RREQ. Fuente: Propia

Esto permite simular la inundación de paquetes RREQ por la red. (Siempre limitado por el TTL máximo establecido = 35 saltos).

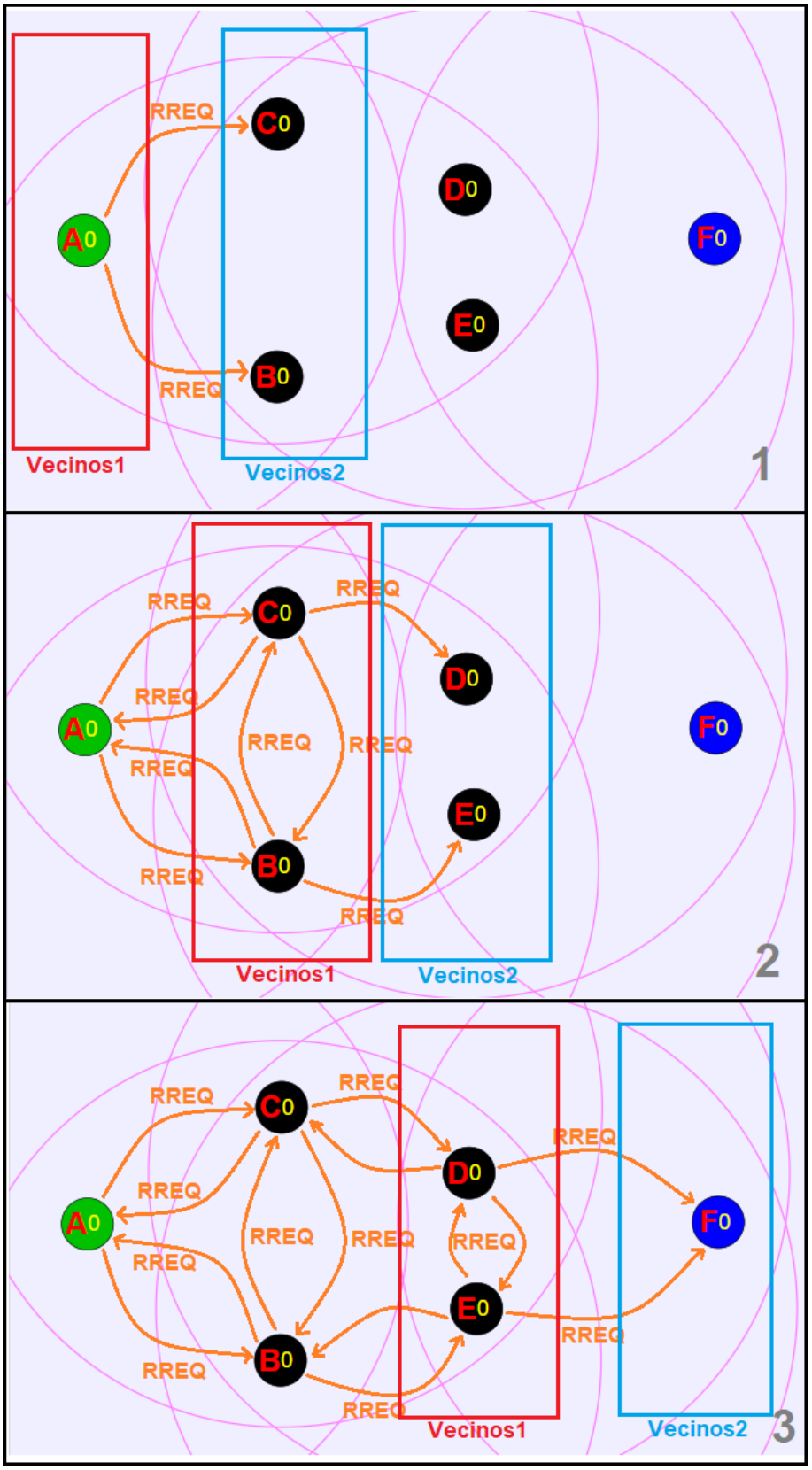


Figura 63: Representación gráfica de la evolución de Vecinos1 y Vecinos2 hasta que se completa la difusión de mensajes RREQ por toda la red. Fuente: Propia

Gráficamente, cada vez que un nodo ha establecido una ruta activa hacia un nodo vecino, aparece un enlace en color verde.

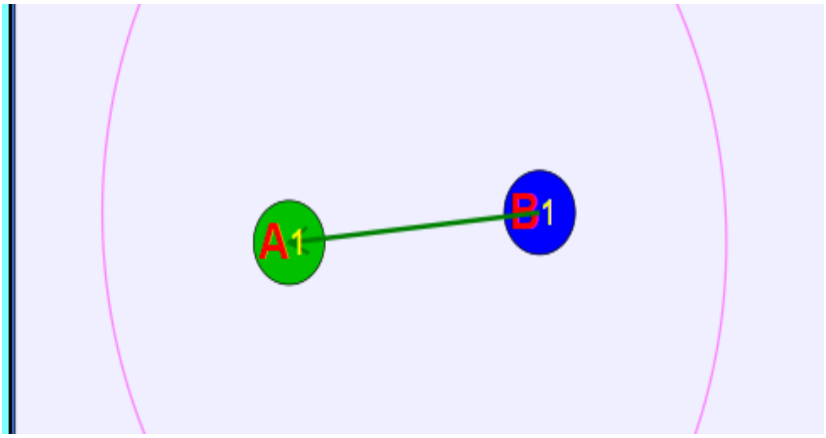


Figura 64: Representación de enlace establecido entre dos nodos con una ruta entre ellos. Fuente: Propia

Para lograr esto, en cada ciclo se comprueba nodo a nodo si se está en cobertura con otro nodo. Para que el enlace sea dibujado, un nodo debe tener una ruta activa y de un salto hacia otro nodo. Si se cumplen estas condiciones, se dibuja el enlace.

```
 VelocidadX[j]=-VelocidadX[j];
 VelocidadY[j]=-VelocidadY[j];
 PosicionenX[i]=PosicionenX[i]+VelocidadX[i]*0.05;//La posicion se le suma la v, que seria la direccion; Se multiplica por 0.05 para estabilizar cuando cambian de d
 PosicionenY[i]=PosicionenY[i]+VelocidadY[i]*0.05;
 PosicionenX[j]=PosicionenX[j]+VelocidadX[j]*0.05;
 PosicionenY[j]=PosicionenY[j]+VelocidadY[j]*0.05;
}

if((distancia(PosicionenX[i], PosicionenY[i], PosicionenX[j], PosicionenY[j]))<=RadioCobertura[i]/2  && i!=j){

ComprobarDibujarEnlace(i,j);//entra en el nodo i, recorre su tabla hasta que encuentra destino j, si se cumple que en su fila la ruta es valida y esta a un salto ac
if(Dibujar==1){
DibujarEnlaces(i,j);
Dibujar=0;
}
}

}//2
```

Figura 65: Método que controla la representación de enlaces entre nodos. Fuente: Propia

En cuanto a las tablas de enrutamiento, para conseguir que estas se rellenen, se emplea la función *tablitas*

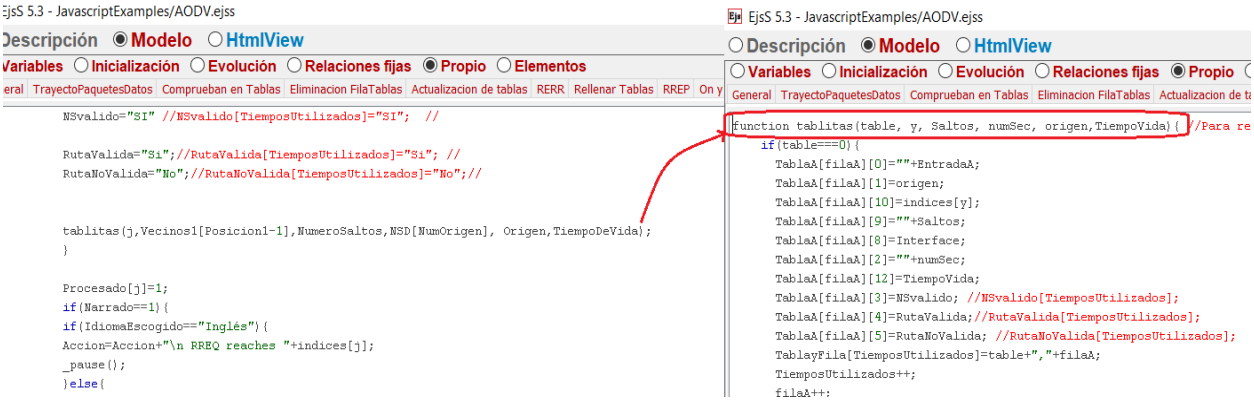


Figura 66: Definición de la función tablitas. Fuente: Propia

Esta permite introducir rutas en las diferentes tablas de encaminamiento de los diferentes nodos.

Una vez ha finalizado la difusión de mensajes RREQ, la variable Activado se establece a 0 para que no entre de nuevo en la función *DifusionRREQ*. Además, se comprueba si con esa difusión de mensajes RREQ se alcanzó el nodo destino o no. Si no lo alcanzó, el espacio de narración indica que el destino es inalcanzable.

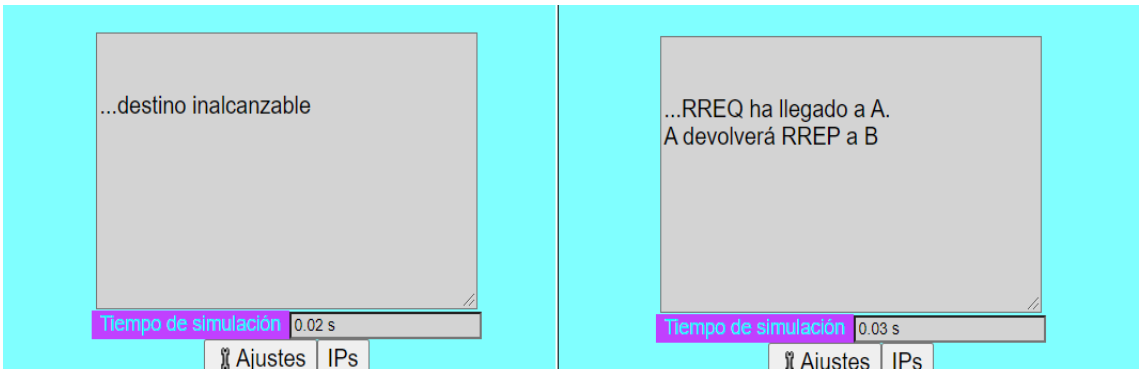


Figura 67: Mensaje mostrado en el espacio de narración indicando si el RREQ alcanzó el nodo destino o no. Fuente: Propia

Si lo alcanzó, se incrementa en 1 el número de secuencia del nodo destino y se compara con el NSD que llega en el paquete RREQ. El nodo destino establecerá como propio el NSD más alto tras comparar estos dos.

```
    Accion=Accion+"\n\n...RREQ ha llegado a "+Destino+".\n"+Destino+" devolverá RREP a "+Origen;
    _pause();
    }
    }
    NSD[NumDestino]=NSD[NumDestino]+1;
    if(NSDqueLlega>NSD[NumDestino]){//Esto es para la condicion de que si un nodo que genera un
      NSD[NumDestino]=NSDqueLlega;
      }
    RREPintermedio=0;
    NumeroSaltos=0;
    Salto=NumDestino;
    }
}
```

Figura 68: Método que establece el NSD de un nodo destino. Fuente: Propia

Una vez realizado esto, entra en juego la variable *llego* para continuar con la siguiente fase de descubrimiento de rutas

5.3.3.2.2 Unidifusión de mensajes RREP

Cuando la variable *llego* se establece a 1 comienza el turno de la emisión de mensajes RREP.

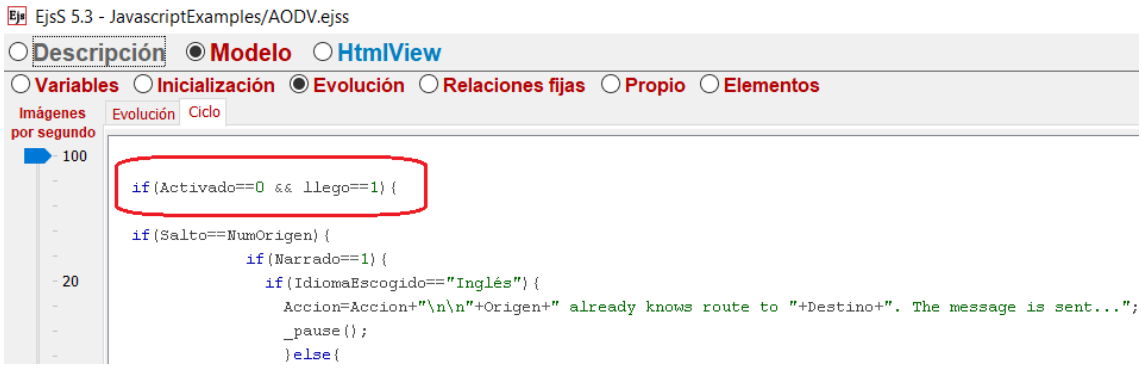


Figura 69: Parte del código encargada de la unidifusión de mensajes RREP. Fuente: Propia

La tarea que se realiza a continuación se entenderá mejor viendo la siguiente imagen:

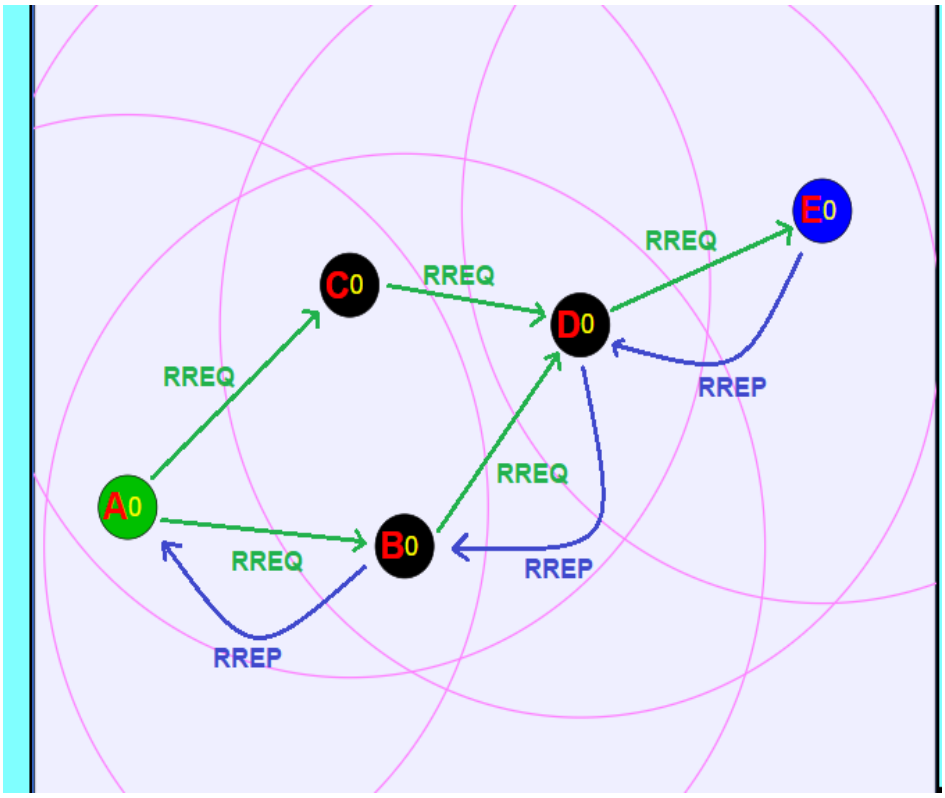


Figura 70: Tarea que se desarrollara en esta parte del código. Fuente: Propia

Las flechas verdes representan la fase anterior, donde se han difundido mensajes RREQ desde el origen A, hasta alcanzar el destino E. La tarea que se realiza ahora es la de recorrer los nodos desde el destino hasta el origen, escogiendo el mejor camino, como el que siguen las flechas azules. Para esto, se desarrolla lo siguiente:

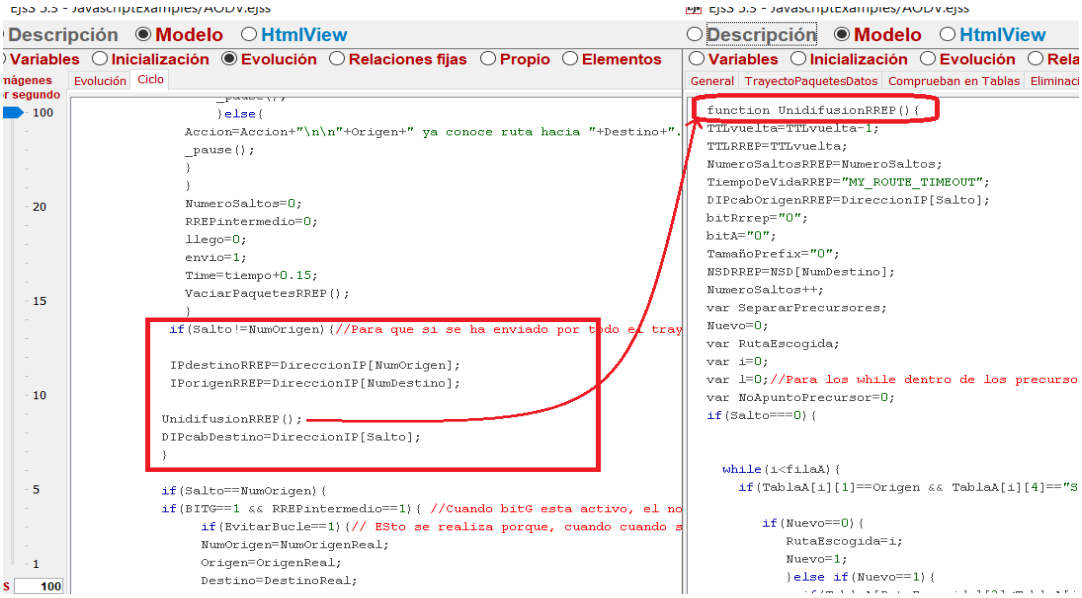


Figura 71: Definición de la función UnidifusionRREP. Fuente: Propia

La idea es ir recorriendo en cada ciclo un nodo, partiendo desde el nodo destino, hasta que en un determinado ciclo alcance al nodo origen. Se utiliza la función *UnidifusionRREP* para esto.

Cuando se analiza un determinado nodo en esta función, observa las rutas ya existentes en su tabla de enrutamiento, las cuales se han registrado durante la difusión de los mensajes RREQ.

E

1	D	0	false	Si	No			wlano	1	D	3.000
2	A	1	SI	Si	No			wlano	3	D	3.000

Figura 72: Tabla de enrutamiento rellena gracias a los mensajes RREQ enviados durante la fase anterior. Fuente: Propia

La función *UnidifusionRREP* escoge la mejor ruta para volver al nodo origen.

Figura 73: Método que permite escoger la mejor ruta para que un nodo envíe el mensaje RREP. Fuente: Propia

Una vez se escoge al siguiente nodo por el que seguirá el camino hacia el origen, se apunta en la tabla de enrutamiento de ese nodo la ruta hacia el destino.

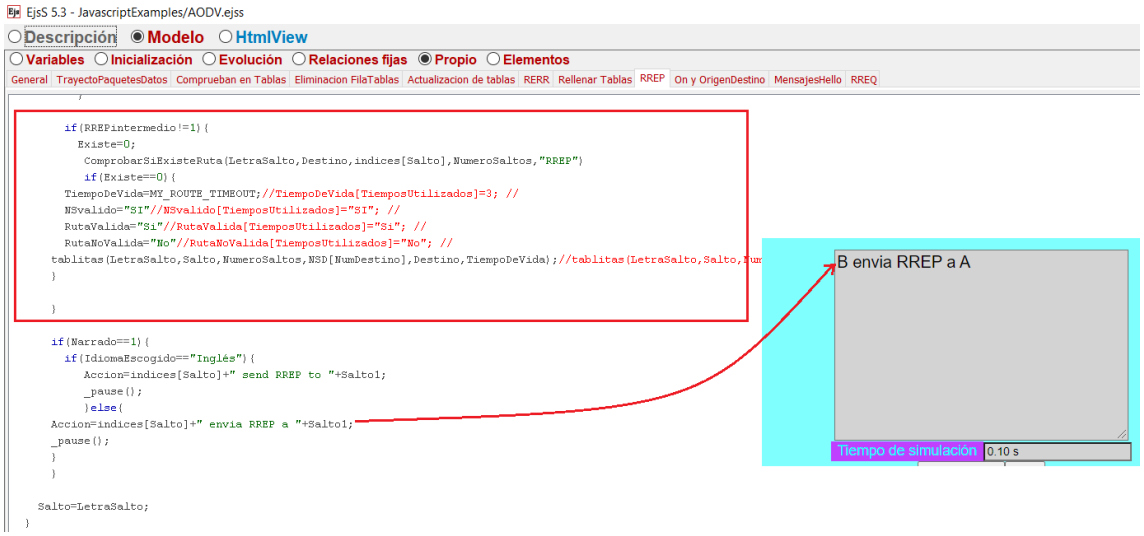


Figura 74: Implementación del método que apunta la ruta hacia el nodo destino en una tabla de enrutamiento y como se muestra esta acción en el espacio de narración. Fuente: Propia

Cuando termina la fase de unidifusión de mensajes RREP, ya que el mensaje RREP llega al nodo origen, el espacio de narración indica que el nodo origen ya ha descubierto una ruta hacia el destino

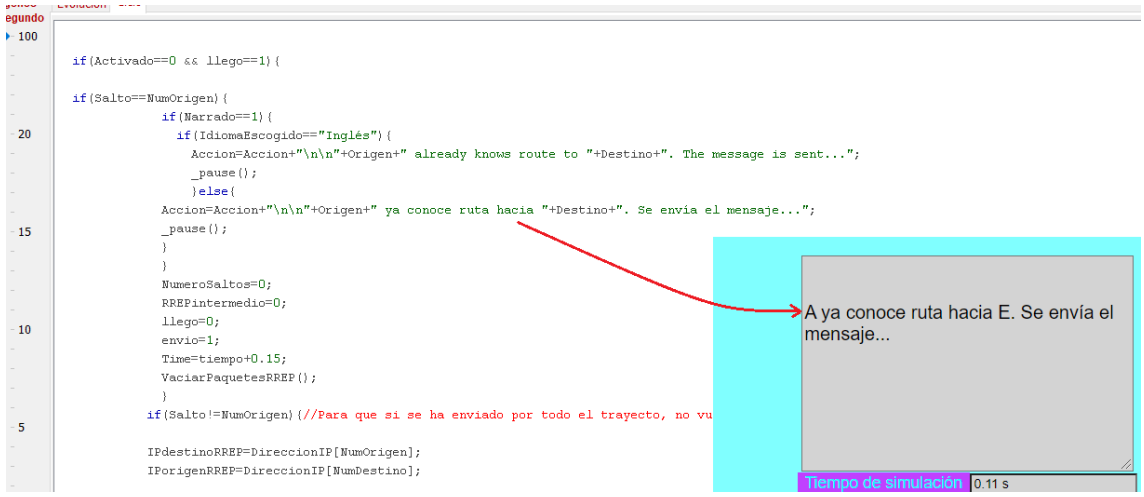


Figura 75: Desarrollo del método que indica el final de la fase de descubrimiento de ruta en el espacio de narración. Fuente: Propia

En este punto, la variable *llego* se establece a 0 para que no vuelva a entrar en la fase de envíos RREP y utilizamos la variable *envio*.

Cuando esta variable es igual a 1, el software entrará en una fase donde simulará el trayecto que describirán los datos que el origen quiere enviar hacia el destino.

5.3.3.2.3 Envío del mensaje desde origen a destino cuando las rutas se han establecido

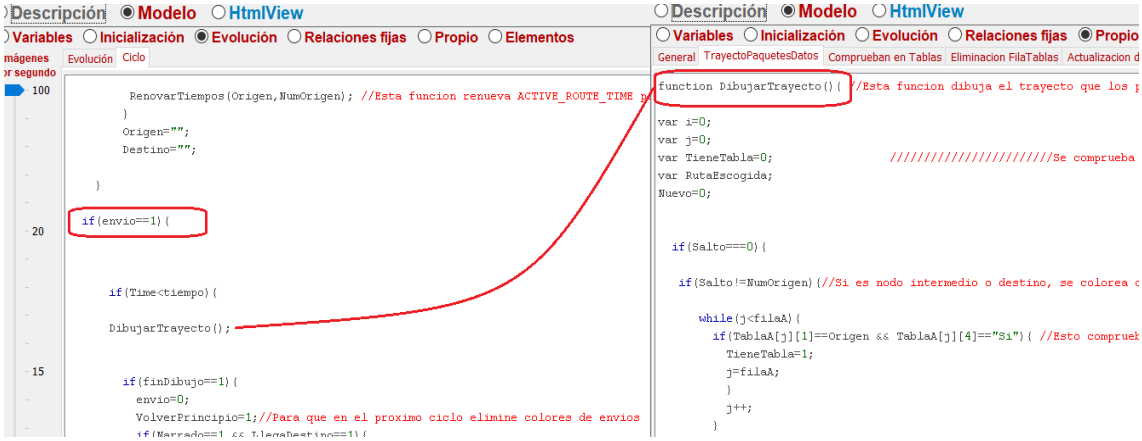


Figura 76: Definición de la función encargada de dibujar el trayecto que siguen los datos desde origen a destino tras establecerse las rutas. Fuente: Propia

Dentro de esta fase, se llamará a la función *DibujarTrayecto*. Esta función se encarga de colorear uno por uno los nodos que conforman el trayecto que toman los datos que se envían desde el origen hacia el destino. La siguiente imagen muestra el proceso gráfico de lo que se consigue con dicha función.

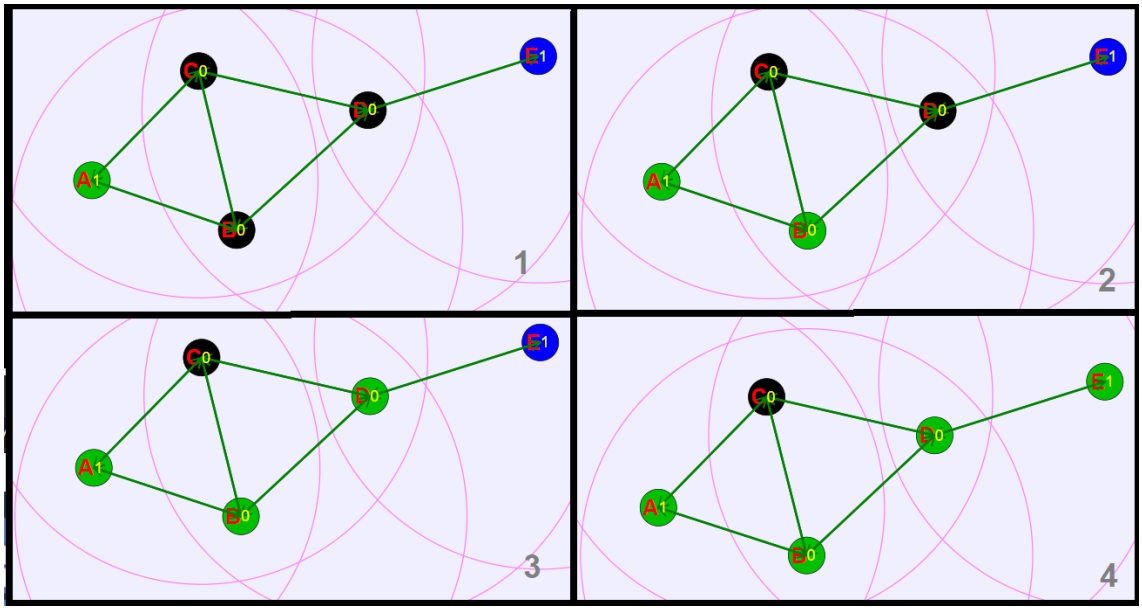


Figura 77: Representación gráfica de la trayectoria descrita por los datos enviados desde el nodo A, hasta el nodo E. Fuente: Propia

Cuando el mensaje alcanza el destino, el espacio de narración lo indica.

```
if(Time<tiempo){

DibujarTrayecto();


  if(finDibujo==1){
    envio=0;
    VolverPrincipio=1;//Para que en el proximo ciclo elimine colores de envios
    if(Narrado==1 && LlegaDestino==1){
      if(IdiomaEscogido=="Inglés"){
        Accion=Accion+"\n\nTHE MESSAGE HAS REACHED ITS DESTINATION";
        _pause();
        }else{
   Accion=Accion+"\n\nEL MENSAJE HA LLEGADO A SU DESTINO";
    _pause();
    }
    }

  }
 }
}
```

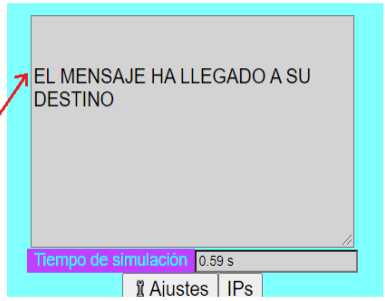


Figura 78: Momento en el que el espacio de narración indica que los datos han llegado al nodo destino. Fuente: Propia

Tras el uso de las rutas entre nodo origen y destino para enviar datos, la teoría del protocolo AODV indica que dichas rutas empleadas deben incrementar su tiempo de vida. Para que el simulador realice esto, tras finalizar la tarea de dibujar el envío de datos, la variable *envio* se establece a 0 y entra en juego una nueva variable nombrada como *VolverPrincipio.*

5.3.3.2.4 Incremento del tiempo de vida de las rutas empleadas para la comunicación

En este punto, se establece que *VolverPrincipio* = 1, lo que provoca la entrada del programa en el siguiente apartado.

Figura 79: Definición del método utilizado para renovar los tiempos de las rutas utilizadas para el envío de datos entre nodos. Fuente: Propia

Aquí se utiliza la función *RenovarTiempos,* la cual se encarga de analizar las rutas que fueron utilizadas para el envío de los datos desde el origen hacia el destino y, en ese mismo ciclo, incrementa el tiempo de todas esas rutas.

El desarrollo de las funcionalidades descritas hasta ahora suponen la parte más básica del simulador de protocolo AODV, ya que se ha explicado cómo se difunden los mensajes RREQ y RREP, así como simular el envío de datos entre un nodo origen y destino.

A continuación, se describirá cómo se han desarrollado los aspectos más complejos del simulador AODV.

5.3.3.3 Evolución de las tablas de enrutamiento

Una vez que se establecen rutas en las tablas de enrutamiento, con el paso de los ciclos de ejecución, el tiempo de vida de dichas rutas se va modificando.

En cada ciclo de ejecución, los tiempos de vida de cada una de las rutas existentes se irán decrementando. Esto se consigue gracias a la función *CambioTiempo.*

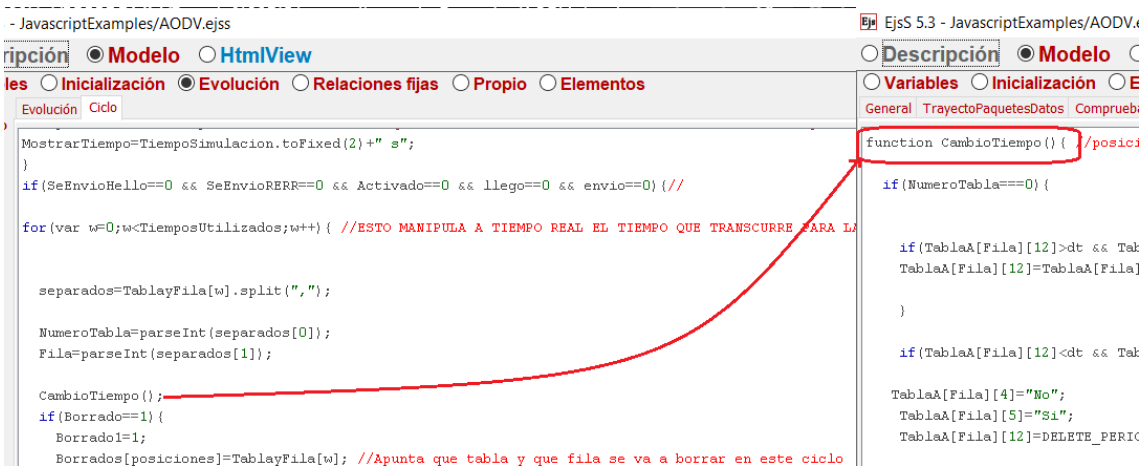


Figura 80: Definición de la función encargada de actualizar el campo Tiempo de Vida de cada ruta registrada en las tablas de enrutamiento. Fuente: Propia

CambioTiempo, en primer lugar, comprueba si una ruta es válida o no. Si la ruta es válida, el tiempo existente es decrementado.

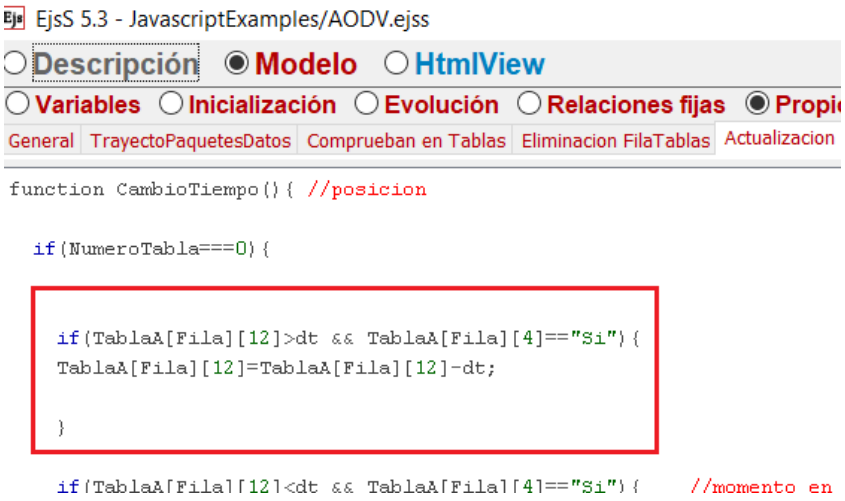


Figura 81: Método que decrementa el tiempo de vida de una ruta válida. Fuente: Propia

Cuando el tiempo de vida de una ruta válida llega a 0, se establece como ruta no válida y se restablece su tiempo de vida, siendo este mayor a 0.

```
if(TablaA[Fila][12]>dt && TablaA[Fila][4]=="Si"){
TablaA[Fila][12]=TablaA[Fila][12]-dt;

}
if(TablaA[Fila][12]<dt && TablaA[Fila][4]=="Si"){    //momento en el que se invalida ruta

TablaA[Fila][4]="No";
 TablaA[Fila][5]="Si";
 TablaA[Fila][12]=DELETE_PERIOD;

if(TablaA[Fila][9]==1){ //ESto lo hace porque cuando se elimina una ruta que esta a mas de un salt
  Salto=0;
  Salto1=TablaA[Fila][10];
```

Figura 82: Método encargado de invalidar una ruta válida cuando su tiempo de vida expira. Fuente: Propia

Cuando la ruta no es válida, en cada ciclo se decrementa su tiempo de vida, al igual que cuando la ruta era válida. Cuando el tiempo de vida de la ruta inactiva llega a 0, se elimina la ruta utilizando la función *BorrarTablas*.

```
if(TablaA[Fila][12]>dt && TablaA[Fila][4]=="No"){
TablaA[Fila][12]=TablaA[Fila][12]-dt;

}

if(TablaA[Fila][12]<dt && TablaA[Fila][4]=="No"){
 BorrarTablas();

}

}
```

Figura 83: Método encargado de eliminar una ruta de una determinada tabla de enrutamiento. Fuente: Propia

Para que una ruta sea eliminada de las tablas de encaminamiento, se debe conocer lo siguiente:

Cada una de las rutas existentes en todas las tablas de enrutamiento, se encuentran apuntadas en un vector conocido como *TablayFila*.

```
TablayFila[TiemposUtilizados]=table+","+filaA;
```

Figura 84: Vector empleado para registrar cada una de las rutas definidas. Fuente: Propia

Este vector almacena, en cada una de sus posiciones, qué fila de qué tabla posee una ruta apuntada. Esto se consigue incluyendo en cada posición del vector *TablayFila*, la tabla en cuestión junto con la fila.

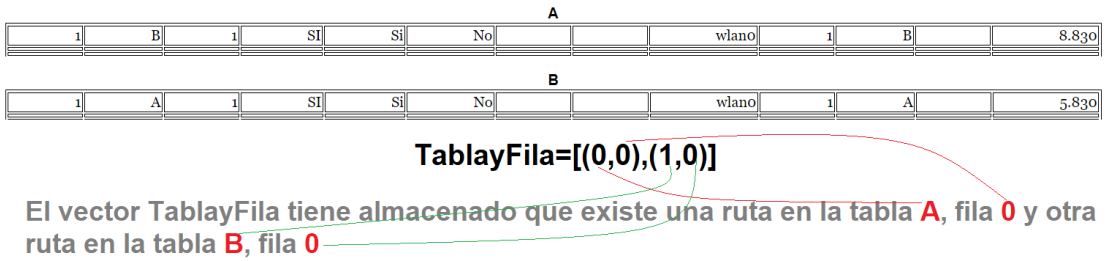


Figura 85: Explicación sobre el contenido del vector TablayFila. Fuente: Propia

Sabiendo esto, para que una ruta sea borrada, se realizan los siguientes pasos:

Suponiendo que la tabla A está rellena de la siguiente forma:

A

1	B	0	false	Si	No			wlano	1	B		1.690
2	C	0	false	Si	No			wlano	1	C		1.690
3	D	0	false	No	Si			wlano	1	D		0.090
4	E	1	SI	Si	No			wlano	1	E		9.690

TablayFila=[(0,0),(0,1),(0,2),(0,3)]

Figura 86: Estructura de la tabla de enrutamiento del nodo A rellena. Fuente: Propia

Como se puede apreciar, la entrada nº3 se encuentra inválida y casi ha expirado su tiempo de vida. Cuando esta llegue a 0, se llama a la función *BorrarTablas*.

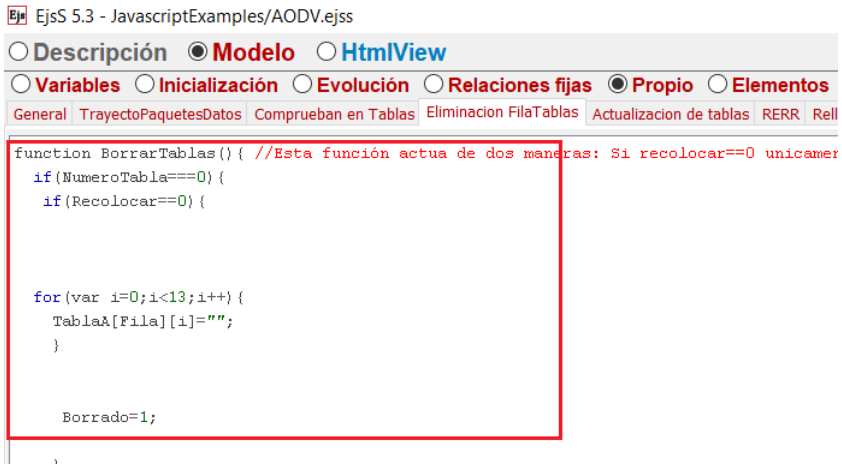


Figura 87: Parte de la función BorrarTablas que permite la eliminación visual de una ruta. Fuente: Propia

Esta parte, se encargará de eliminar "gráficamente" la ruta de la tabla de encaminamiento del nodo A.

A

1	B	0	false	Si	No			wlano	1	B		1.690
2	C	0	false	Si	No			wlano	1	C		1.690
4	E	1	SI	Si	No			wlano	1	E		9.690

TablayFila=[(0,0),(0,1),(0,2),(0,3)]

Figura 88: Resultado de eliminar gráficamente la entrada nº3 de la tabla de enrutamiento del nodo A. Fuente: Propia

A continuación, echando un vistazo en el vector *TablayFila*, se aprecia que aún sigue existiendo la ruta borrada gráficamente, ya que es la (0,2). Para eliminarla, se emplea la función *Reestructurar*.

```
posiciones=0;


if(Borrado1==1){


while(z<TiemposUtilizados){ //sI ALGUNA FILA SE ELIMINA, SE

  if(Borrados[posiciones]==TablayFila[z]){

      Reestructurar(z);
      TiemposUtilizados--;
```

```
function Reestructurar(posicion){ //Esta funcion, cuando se elimina una fila de una tabla, detec

var diferencia = TiemposUtilizados -(posicion+1); //Para saber cuantos movimientos hay que hacer

for(var j=0;j<diferencia;j++){
    TablayFila[posicion]=TablayFila[posicion+1];
    posicion++;
    }
}
```

Figura 89: Definición de la función Reestructurar. Fuente: Propia

Tras esto, el resultado sería el siguiente:

							A					
1	B	0	false	Si	No			wlano	1	B		1.690
2	C	0	false	Si	No			wlano	1	C		1.690
4	E	1	SI	Si	No			wlano	1	E		9.690

TablayFila=[(0,0),(0,1),(0,3)]

Figura 90: Eliminación de la ruta dentro del vector TablayFila. Fuente: Propia

La siguiente tarea es la de apilar todas las rutas en la parte superior de la tabla. Como se puede apreciar, existe un hueco entre la entrada nº2 y la nº4. La idea es eliminar esos huecos y unir todas las rutas arriba. Para esto, se vuelve a entrar de nuevo en la función *BorrarTablas*, la cual también desempeña la tarea de reorganizar las rutas.

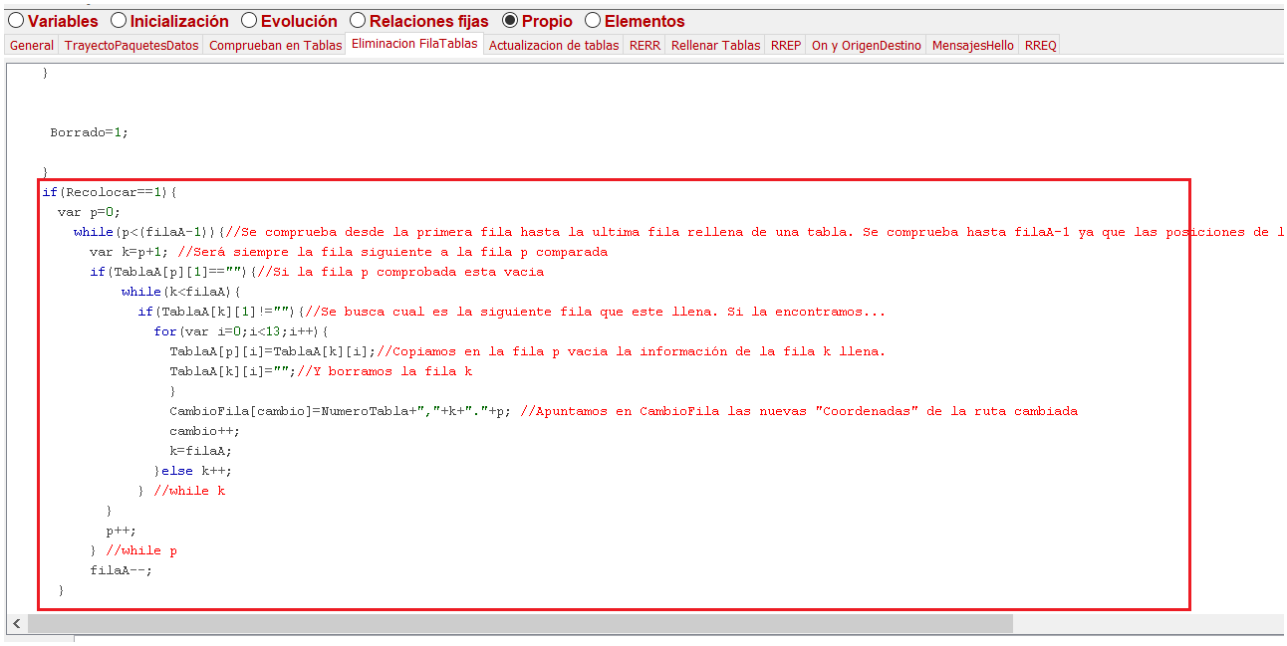


Figura 91: Parte de la función BorrarTablas encargada de apilar gráficamente las rutas existentes en la tabla de enrutamiento. Fuente: Propia

El resultado que la función nos ofrece es el siguiente:

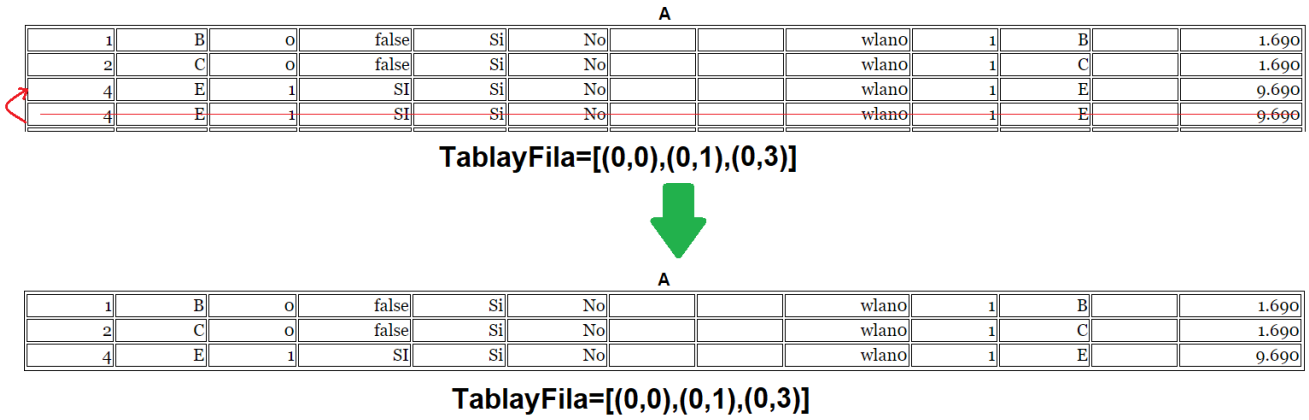


Figura 92: Resultado ofrecido por la parte de la función BorrarTablas descrita anteriormente. Fuente: Propia

Ya casi esta toda la ruta eliminada, tanto gráficamente como del vector *TablayFila*, el cual es necesario para regular todos los tiempos de las rutas. Si se pone atención, la ruta nº4 ha cambiado de posición al ocupar el hueco que la ruta nº3 dejó, por lo que ahora estaría ocupando la posición (0,2) y no la (0,3) como se encuentra registrado en el vector *TablayFila*. Para terminar completamente con la eliminación de la ruta, se debe establecer la posición correcta dentro de *TablayFila* que ahora está ocupando la ruta nº4 en la tabla de enrutamiento del nodo A. Para ello, gracias a un vector conocido como *CambioFila*, utilizado para indicar que rutas han cambiado su fila para ocupar los huecos de las rutas

eliminadas, se podrá establecer las filas actuales que ocupan las rutas que se han desplazado.

```
  NumeroTabla=parseInt(separados[0]);
  BorrarTablas(); //Comprueba los huecos que han aparecido con el borrado de este ciclo y recoloca, apilanc
  Borrados[j]="0";

  }
for(var i=0;i<cambio;i++){ //Esta parte actualiza la nueva posicion que ocupa una fila en una determinada t
  separados=CambioFila[i].split(".");
  var FilaAnterior=(separados[0]);
  var FilaNueva=(separados[1]);
  for(var j=0;j<TiemposUtilizados;j++){
    if(TablayFila[j]==FilaAnterior){
      var sep=TablayFila[j].split(",");
      var tabla=(sep[0]);
      TablayFila[j]=tabla+","+FilaNueva;
      }
    }
    CambioFila[i]="0";
  }
Recolocar=0;
}
```

Figura 93: Método encargado de actualizar las coordenadas de las tablas de enrutamiento registradas en el vector TablayFila. Fuente: Propia

Esto da como resultado:

							A					
1	B	0	false	Si	No			wlano	1	B		1.690
2	C	0	false	Si	No			wlano	1	C		1.690
4	E	1	SI	Si	No			wlano	1	E		9.690

TablayFila=[(0,0),(0,1),(0,2)]

Figura 94: Resultado final del borrado de una ruta de una tabla de encaminamiento. Fuente: Propia

Este es el procedimiento completo para eliminar rutas de las tablas de enrutamiento.

5.3.3.4 Registro de nodos precursores en las tablas de enrutamiento y emisión de mensajes RERR

La tarea de apuntar los nodos precursores dentro de las tablas de enrutamiento se produce únicamente durante la unidifusión de mensajes RREP, por lo que para ello se emplea la función *UnidifusionRREP*.

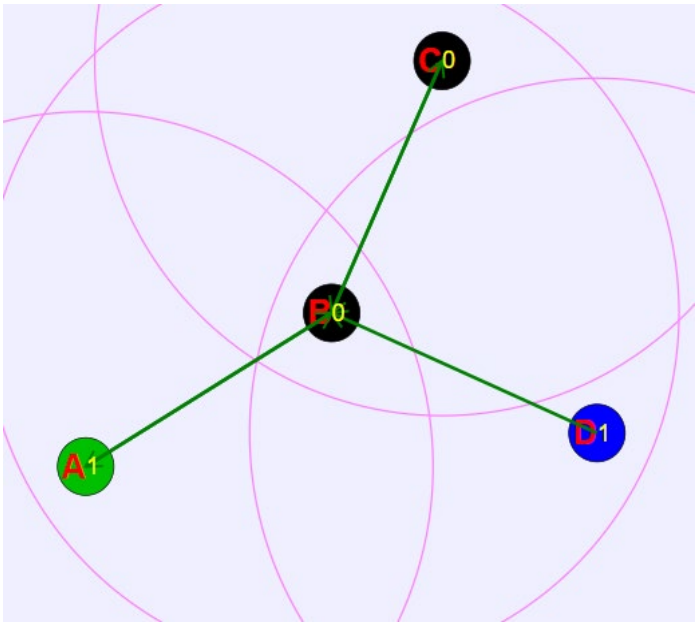


Figura 95: Ejemplo de topología utilizada para describir el procedimiento de apuntar nodos precursores en las tablas de enrutamiento. Fuente: Propia

Apreciando la Figura 95, se supone que los nodos B, D y C tienen una ruta hacia A. El nodo B, al encontrarse entre la ruta que C y D tienen hacia A, apuntará como precursores en su tabla de enrutamiento a D y C

	Ruta hacia nodo A					B					Nodos precursores
1	A	2	SI	Si	No			wlan0	1	A	D,C	8.700

Figura 96: Ruta con destino al nodo A registrada en la tabla de enrutamiento del nodo B. Fuente: Propia

Como se ha comentado anteriormente, la adición de estos nodos precursores a una ruta se realiza dentro de la función *UnidifusionRREP.*

Figura 97: Parte de la función UnidifusionRREP encargada de analizar y apuntar los nodos precursores asociados a un determinado nodo destino. Fuente: Propia

Gracias a esta tarea, los nodos pueden añadir nodos precursores a sus rutas. La principal finalidad de esto es que, cuando un nodo pierda un enlace con otro nodo, avise a sus nodos precursores mediante la emisión de mensajes RERR.

En este punto, se supondrá que el nodo B pierde su enlace hacía A.

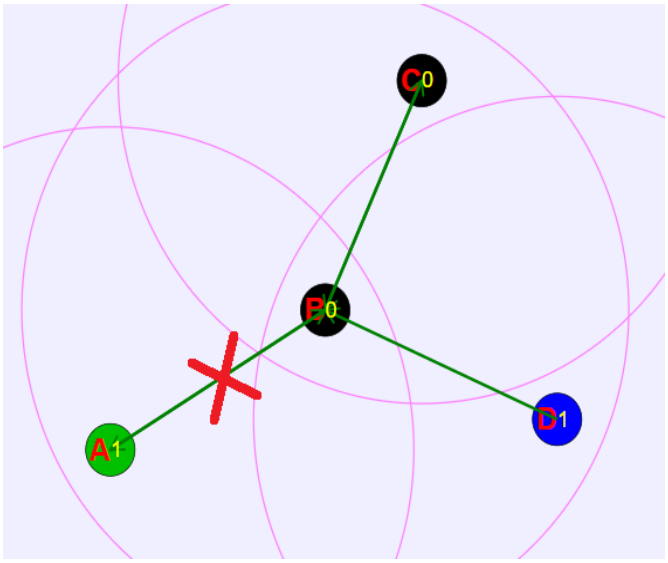


Figura 98: Representación gráfica de la rotura del enlace A-B. Fuente: Propia

Si se observa la tabla de enrutamiento de B, la ruta que tiene hacia A, lleva apuntada como precursores a D y C

B

1	A	2	SI	Si	No			wlan0	1	A	D,C	8.700
2	C	0	false	Si	No			wlan0	1	C	A	5.700
3	D	1	SI	Si	No			wlan0	1	D	A	8.700

Figura 99: Ejemplo de la tabla de enrutamiento del nodo B rellena. Fuente: Propia

Por lo que, al perderse el enlace con el nodo A, el nodo B debe emitir un mensaje RERR hacia D y C avisando de que la ruta hacia A ya no está disponible.

Para ello, se llama a la función *EnviosRERR*.

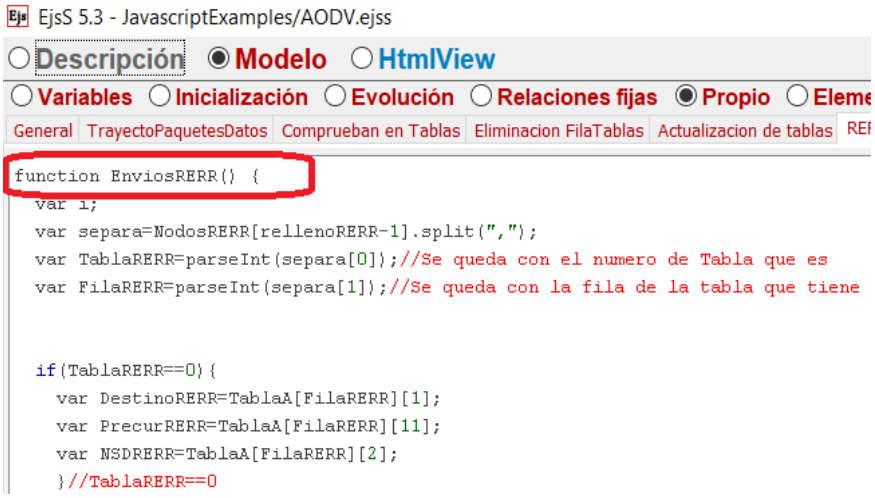


Figura 100: Definición de la función EnviosRERR. Fuente: Propia

Esta función se encarga de analizar qué nodo es el que ha perdido la ruta, que ruta ha perdido y hacia qué nodo o nodos irá destinado el mensaje RERR que este emitirá. Además, dentro de esta función, se establecen los valores que muestran cómo sería el formato de un mensaje RERR.

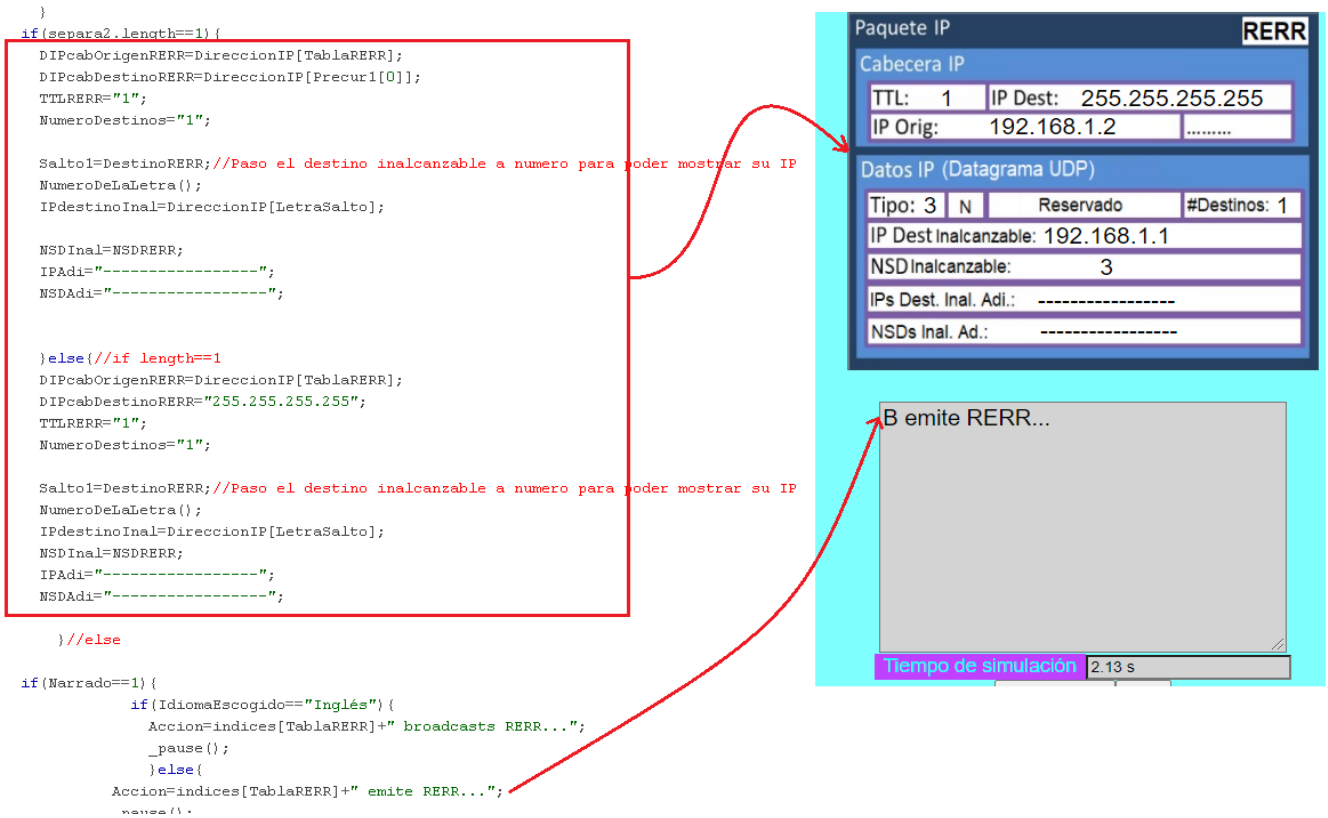


Figura 101: Parte de la función EnviosRERR encargada de mostrar en la interfaz gráfica del simulador el formato del mensaje RERR y la descripción indicada en el espacio de narración. Fuente: Propia

Una vez que la función decreta el nodo o nodos al que irá destinado el RERR, se utiliza la función *ExtenderRERR*.

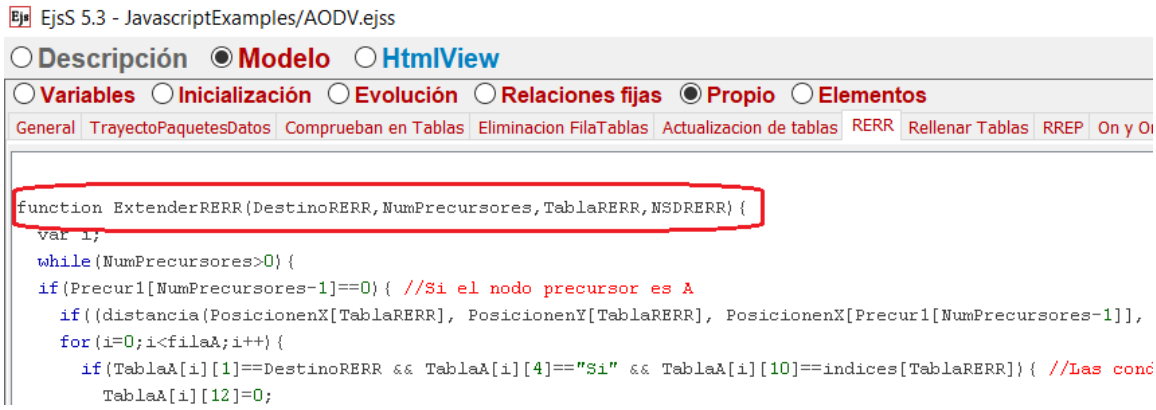


Figura 102: Definición de la función ExtenderRERR. Fuente: Propia

Esta se encargará de anular las rutas afectadas en las tablas de enrutamiento de los nodos que reciben el RERR.

5.3.3.5 Emisión de mensajes HELLO

Los mensajes HELLO son utilizados por los nodos para alertar de su presencia y son enviados a los nodos vecinos. La única condición que existe para que un nodo emita un mensaje HELLO es que este posea al menos, una ruta activa.

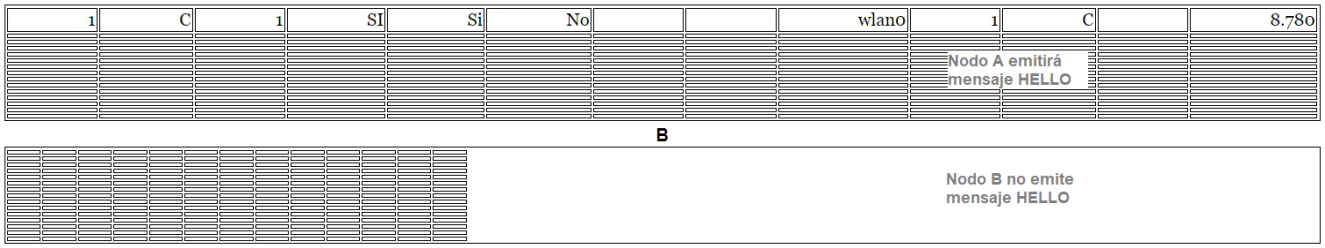


Figura 103: Condición para que un nodo emita un mensaje HELLO. Fuente: Propia

Para que el simulador conozca qué nodos tienen al menos una ruta activa, se emplea el vector *RutaActiva*. Cada una de las posiciones de este vector representará cada una de las tablas de enrutamiento de la simulación. La posición de este vector que tiene el valor 0, significa que no tiene ninguna ruta activa, mientras que, si el valor es 1, significa que al menos tiene una ruta activa.

```
if(RutaActiva[j]==0){//esto compruek
        RutaActiva[j]=1;
        }
```

Figura 104: Método que establece que un nodo tiene (al menos) una ruta activa. Fuente: Propia

Cuando ocurre esto, el nodo debe emitir un mensaje HELLO cada vez que un contador llegue a 0. Cada nodo tendrá su propio contador y se establece gracias al vector *ContadorHello.*

```
CuantosHello++;
}
NoHello[Nodo]=0;
ContadorHello[Nodo]=HELLO_INTERVAL;
```

Figura 105: Definición de los contadores que cada nodo utilizarán para emitir un mensaje HELLO. Fuente: Propia

Una vez que este contador finaliza, el nodo emite un mensaje HELLO a través de la función *EnviosHello*.

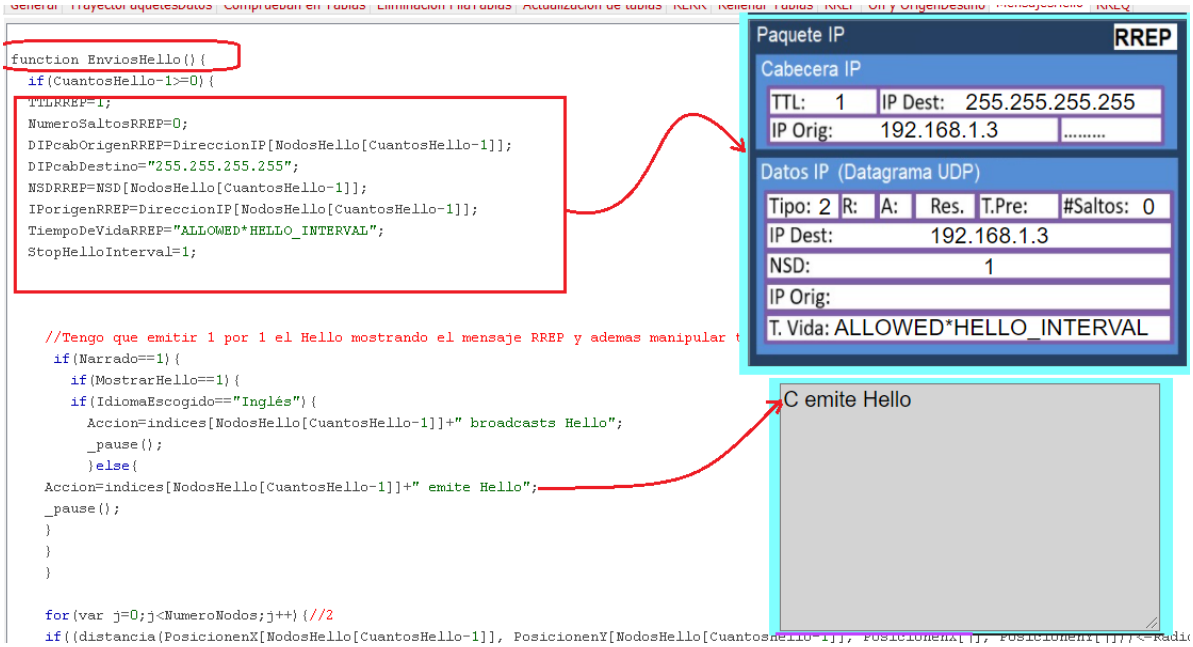


Figura 106: Definición de la función EnviosHello. Fuente: Propia

Una vez ha emitido el mensaje HELLO, el contador se reinicia y, mientras siga teniendo al menos una ruta activa, un nodo emitirá un mensaje HELLO cada vez que su contador propio finalice.

5.3.3.6 Gestión de rutas activas en las tablas de enrutamiento de los nodos simulados

Cuando los nodos que están siendo simulados ya poseen rutas válidas hacía el resto de nodos, entran en juego otra serie de funciones necesarias para el correcto funcionamiento del simulador AODV. Estas funciones responderán a las siguientes preguntas:

- ¿Qué ocurre cuando se tiene que escribir en una tabla de enrutamiento una ruta que ya existe?

Para esta cuestión se ha creado la función *ComprobarSiExisteRuta*.

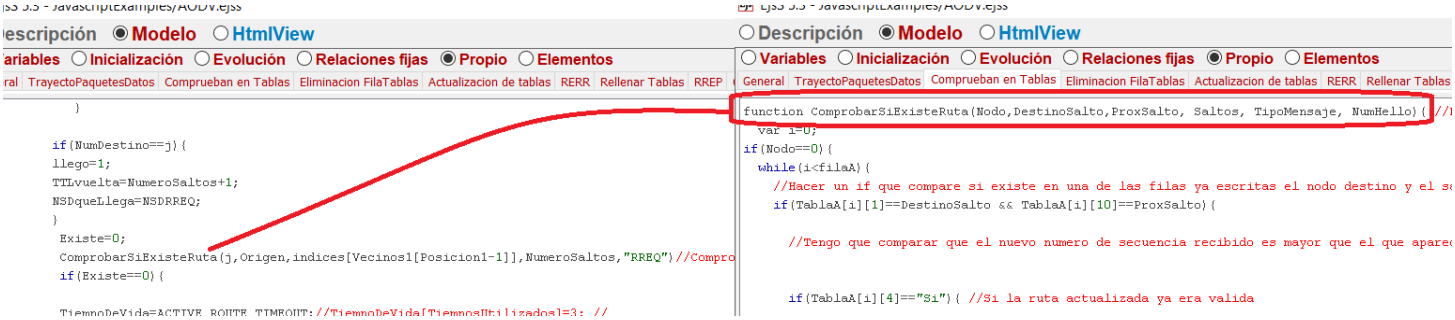


Figura 107: Definición de la función ComprobarSiExisteRuta. Fuente: Propia

Esta función es utilizada en cada una de las funciones que introducen nuevas rutas a las tablas de enrutamiento. *DifusionRREQ*, *UnidifusionRREP*, *EnviosHello* son algunas de estas funciones. *ComprobarSiExisteRuta* tiene la finalidad de comprobar si la nueva ruta a introducir en una tabla de enrutamiento ya existe en esta y en qué condiciones se encuentra (válida o no válida). Si la nueva ruta a introducir ya existe en una tabla de enrutamiento, pero esta aparece como no válida, la función *ComprobarSiExisteRuta* actualizará la ruta existente y la hará válida de nuevo. En cambio, si la nueva ruta a introducir ya existe y, además aparece como válida en la tabla, *ComprobarSiExisteRuta* aumentará el tiempo de vida de dicha ruta. En resumen, esta función tiene la tarea de evitar que una misma ruta aparezca más de una vez en una tabla de enrutamiento.

Como nos indica la teoría del protocolo AODV, si un nodo origen desea conocer una ruta hacia un nodo destino, este origina un mensaje RREQ. Si durante el trayecto del mensaje RREQ, este es recibido por un nodo intermedio que posee una ruta activa hacia el destino, este nodo intermedio podrá responder al RREQ con un RREP hacia el nodo origen, proporcionandole la ruta para comunicarse con el nodo destino.

Figura 108: Representación de que ocurre cuando un nodo intermedio posee una ruta activa hacia un nodo destino. Fuente: Propia

Además, los nodos intermedios que poseen rutas al destino, no reenvían mensajes RREQ (salvo excepción que se verá a continuación). Sabiendo esto…

- ¿Cómo sabe el simulador si un nodo intermedio posee una ruta activa hacia el nodo destino buscado?

Esta tarea está a cargo de la función *ComprobarDestino*.

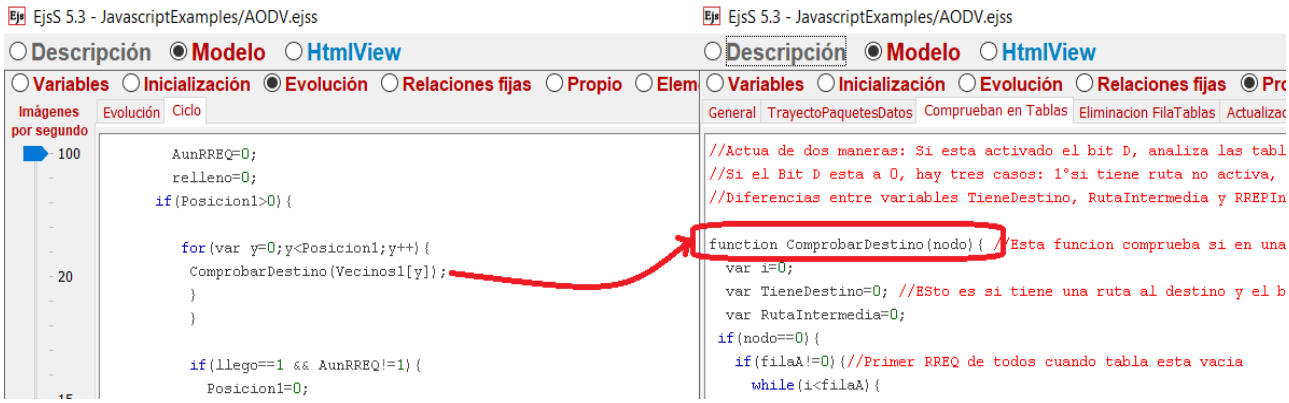


Figura 109: Definición de la función ComprobarDestino. Fuente: Propia

Esta función revisa el vector *Vecinos1*, del cual se habló anteriormente, para comprobar si alguno de los nodos almacenados en dicho vector, posee una ruta activa en su tabla de enrutamiento hacia el nodo destino.

Ese sería el resumen de lo que esta función realiza, aunque se entrará más en detalle después de comentar unos aspectos necesarios a conocer:

Dentro del simulador AODV, los nodos pueden llegar a formar cualquier tipo de topología de red. Con algunas topologías, la tarea de que un nodo origen pueda obtener una ruta hacia un nodo destino gracias a un nodo intermedio, puede resultar sencilla.

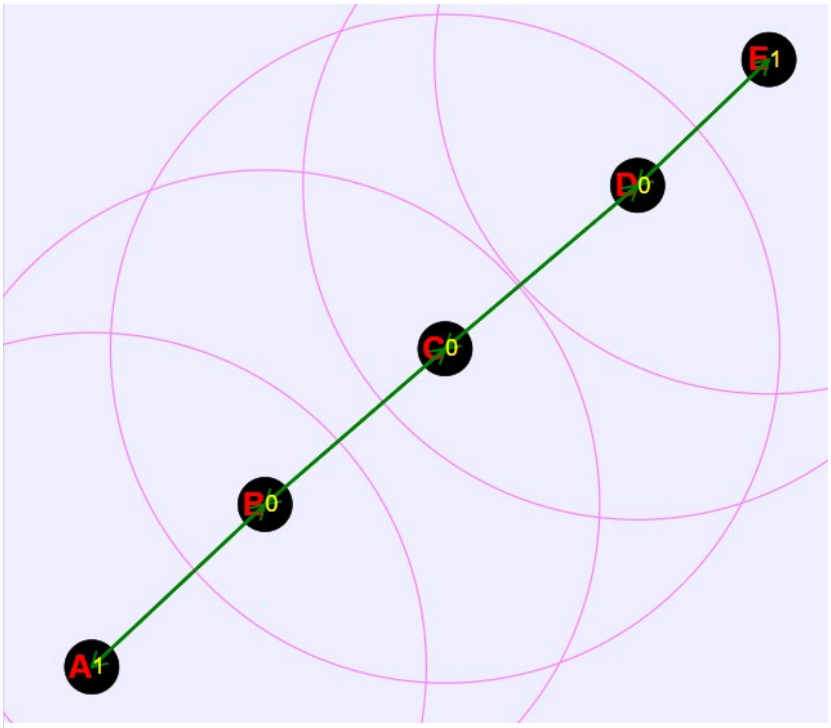


Figura 110: Ejemplo de topología sencilla. Fuente: Propia

Sin embargo, también pueden formarse topologías que dificultan esta tarea.

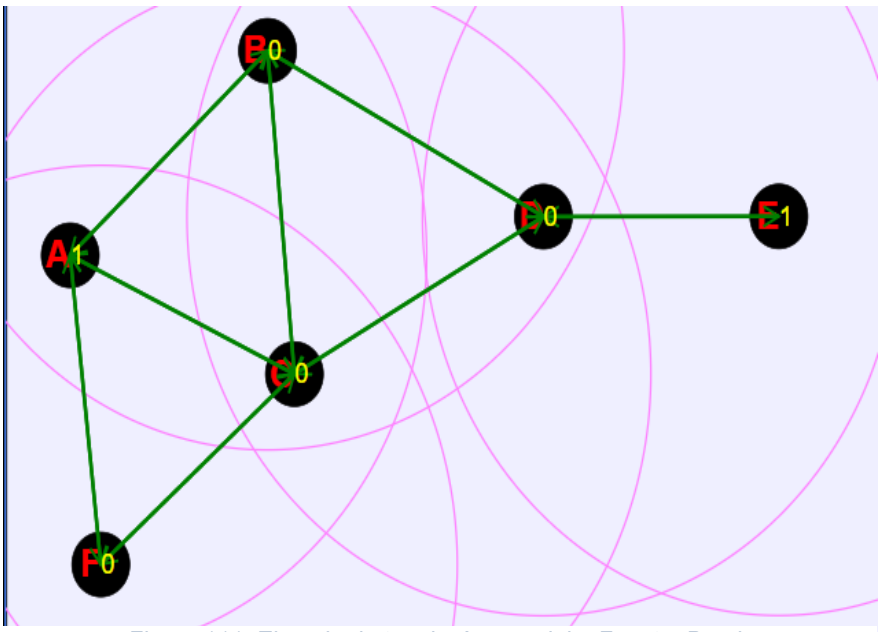


Figura 111: Ejemplo de topología compleja. Fuente: Propia

Por ello, para el desarrollo de la simulación, se ha ideado un algoritmo que permite ofrecer al nodo origen una única mejor ruta posible hacia un nodo destino a través de un nodo intermedio, capaz de evitar que la red se colapse por el envío y recepción de varios RREP simultáneos.

Este algoritmo prioriza, en el siguiente orden, las tareas a realizar para ofrecer al nodo origen, la mejor ruta hacia el nodo destino:

1) Mientras existan nodos que tengan que difundir RREQ, se difundirán RREQ. Lo que viene a decir que, aunque un mensaje RREQ haya alcanzado un nodo intermedio con una ruta activa hacia el destino, o incluso, el mensaje RREQ haya alcanzado el propio destino, si hay nodos que tienen que difundir RREQ, lo difundirán.

2) Cuando todos los nodos terminan de difundir el RREQ, se comprobará si alguno de los mensajes difundidos llegó al nodo destino. Si llegó hasta el destino, aunque el mensaje también llegase a un nodo intermedio con ruta válida hacia el destino, el RREP lo emitirá sólo el nodo destino, ya que este proporcionará la información más actualizada.

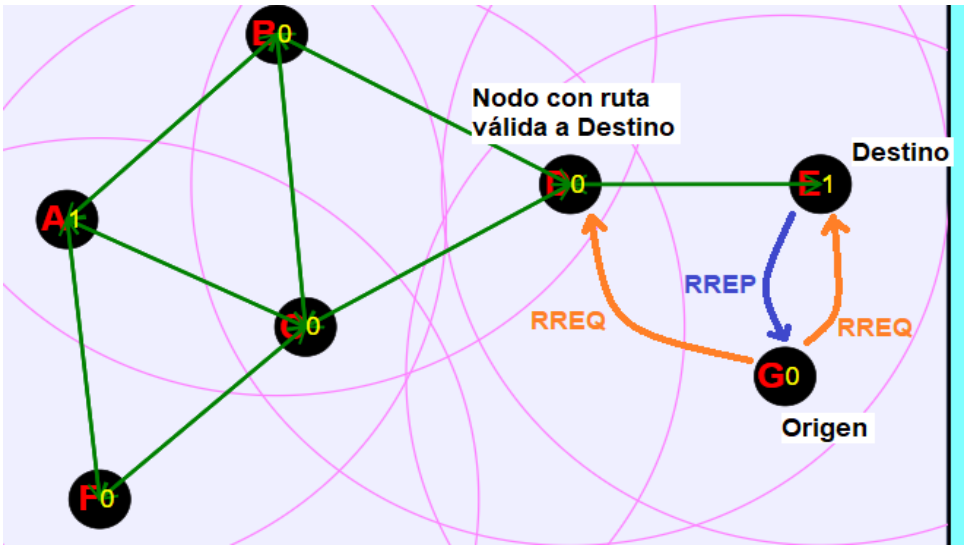


Figura 112: Ejemplo donde se aprecia el 2º orden de prioridad, ya que un RREQ ha alcanzado directamente al nodo destino. Fuente: propia

3) Si todos los nodos han difundido RREQ, uno de estos alcanzó a uno o varios nodos intermedios que poseen una ruta válida hacia el destino y ninguno de estos RREQ

difundidos alcanzaron al nodo destino, el nodo intermedio que mejor posicionado y más actualizado este con respecto al nodo destino, devolverá el RREP.

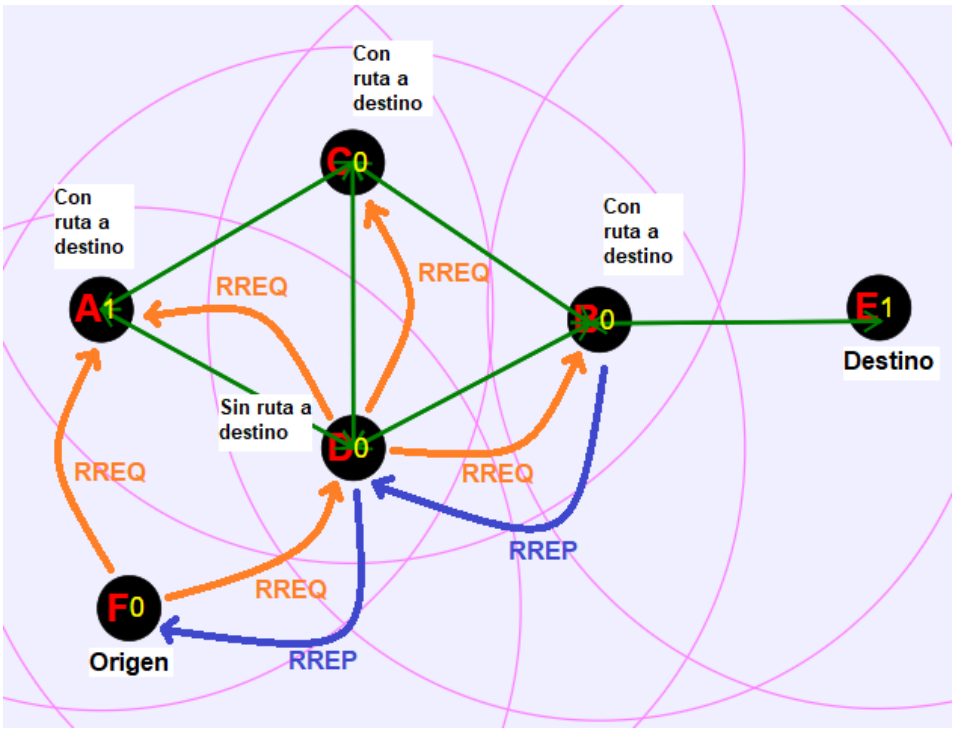


Figura 113: Ejemplo donde se aprecia el 3º orden de prioridad, ya que el RREQ no alcanza directamente el nodo destino. Fuente: Propia

Como se puede apreciar en la figura 113, el nodo intermedio A, B y C tienen ruta válida hacia el nodo destino, pero responde B con un RREP puesto que la ruta que pasa por él es la más corta para el origen (Está a tres saltos del destino).

Una vez comentados estos aspectos, se puede decir que la función ComprobarDestino desempeña el papel de comprobar si un nodo posee una ruta válida hacia el nodo destino y, si existen varias, determinar cuál es la mejor ruta.

5.3.3.7 Modificación de los valores del Bit G y Bit D

Tanto el Bit G, como el Bit D, aparecen indicados dentro de los mensajes RREQ

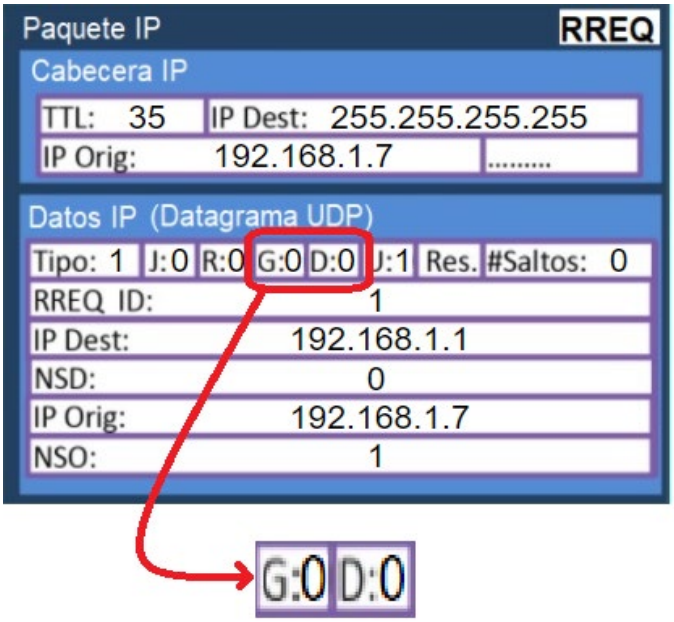


Figura 114: Localización del Bit G y el Bit D en un mensaje RREQ. Fuente: Propia

- Bit D:

Si este Bit se encuentra a 0, los nodos intermedios pueden responder a un mensaje RREQ con un RREP si estos poseen una ruta válida hacia el destino solicitado. En cambio, si el Bit D es igual a 1, estos nodos intermedios que reciban un mensaje RREQ, tendrán que reenviarlo a sus nodos vecinos. En esta situación, solo el nodo destino puede generar un mensaje RREP.

El simulador recrea este comportamiento creando una condición dentro de la función *ComprobarDestino* explicada anteriormente, forzando que los nodos intermedios con rutas activas al destino, reenvíen mensajes RREQ.

```
NoRREQ=1;
NumeroSaltos=0;
Time=tiempo+0.1;
}//if de si es nodo Origen
if(nodo!=NumOrigen){ ///////Tengo que forzar que entre en la parte de u

if(NSDRREQ<=TablaA[i][2] && BITD==0){ //Asegurarnos que el numero de se
RREPintermedio=1;
if(llego==0){
  if(Nuevo==0){//Variable nuevo se utiliza para conocer si durante la e
    Nuevo=1;
    NSDintermedio=TablaA[i][2];
    TotalSaltos=parseInt(TablaA[i][9])+(NumeroSaltos-1);//Para saber sa
    //NumeroSaltos=TablaA[i][9];
```

Figura 115: Condición que permite recrear el comportamiento de tener activado el Bit D. Fuente: Propia

- Bit G:

Si este es igual a 1, el nodo intermedio que genera el RREP hacia el nodo origen en respuesta a un mensaje RREQ, está obligado a generar otro mensaje RREP, pero esta vez dirigido al nodo destino, donde se le ofrecerá la ruta para comunicarse con el nodo origen.

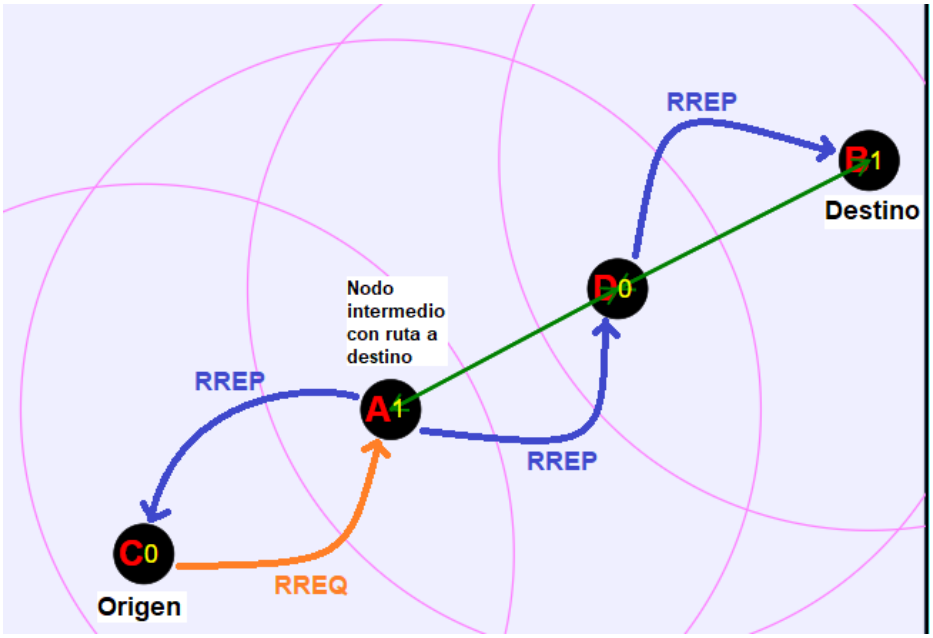


Figura 116: Representación gráfica del proceso que se realiza cuando el Bit G esta activo. Fuente: Propia

El nodo C aprenderá la ruta hacia el nodo destino B, pero gracias a que el Bit G es igual a 1, tanto el nodo D como el nodo B, aprenderán la ruta hacia el nodo origen C.

Si el Bit G se encuentra desactivado, el nodo intermedio no genera RREP en dirección al nodo destino.

Para activar o desactivar estas características, basta con entrar en los ajustes del simulador.

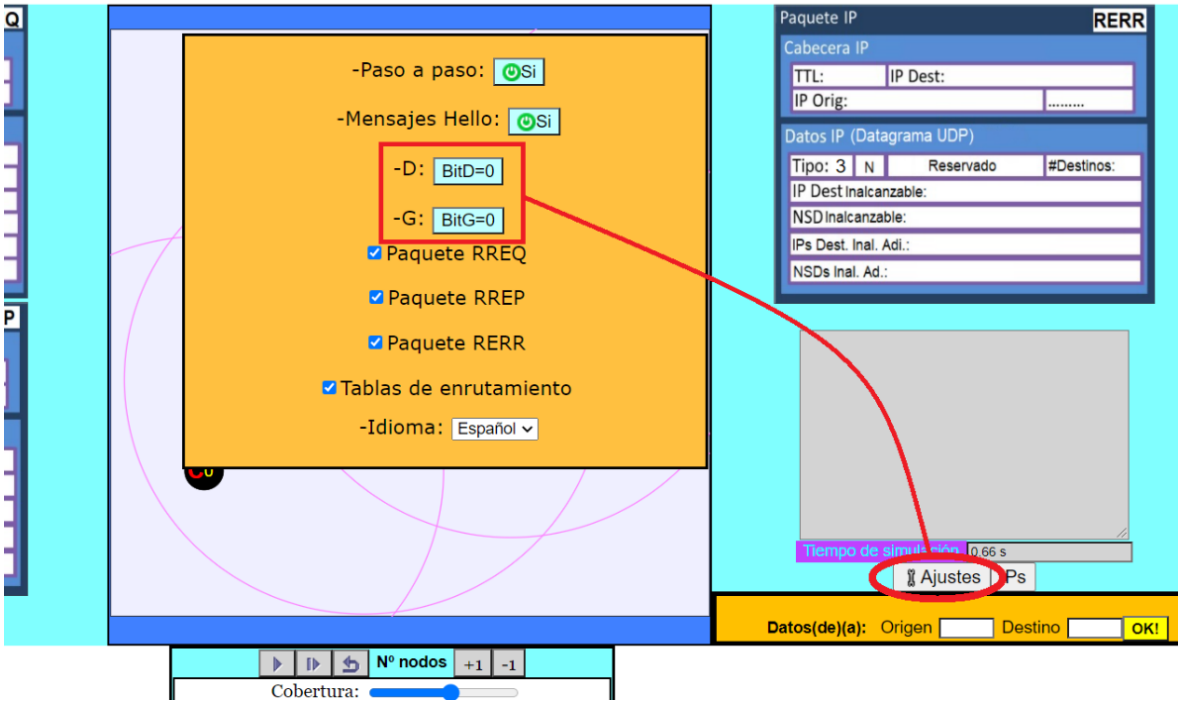


Figura 117: Como activar/desactivar el Bit G y D en la interfaz de simulación. Fuente: Propia

5.4 Desarrollo de las actividades/tutorial del simulador AODV

Para conocer cómo se utiliza el simulador AODV, se han desarrollado una serie de actividades sobre este, las cuales permiten conocer cómo funcionan cada uno de los aspectos que presenta el simulador. Estas actividades vendrían a ser una especie de tutorial previo antes de acceder al simulador AODV completo.

Cuando comienzan estas actividades, se nos muestra la siguiente interfaz:

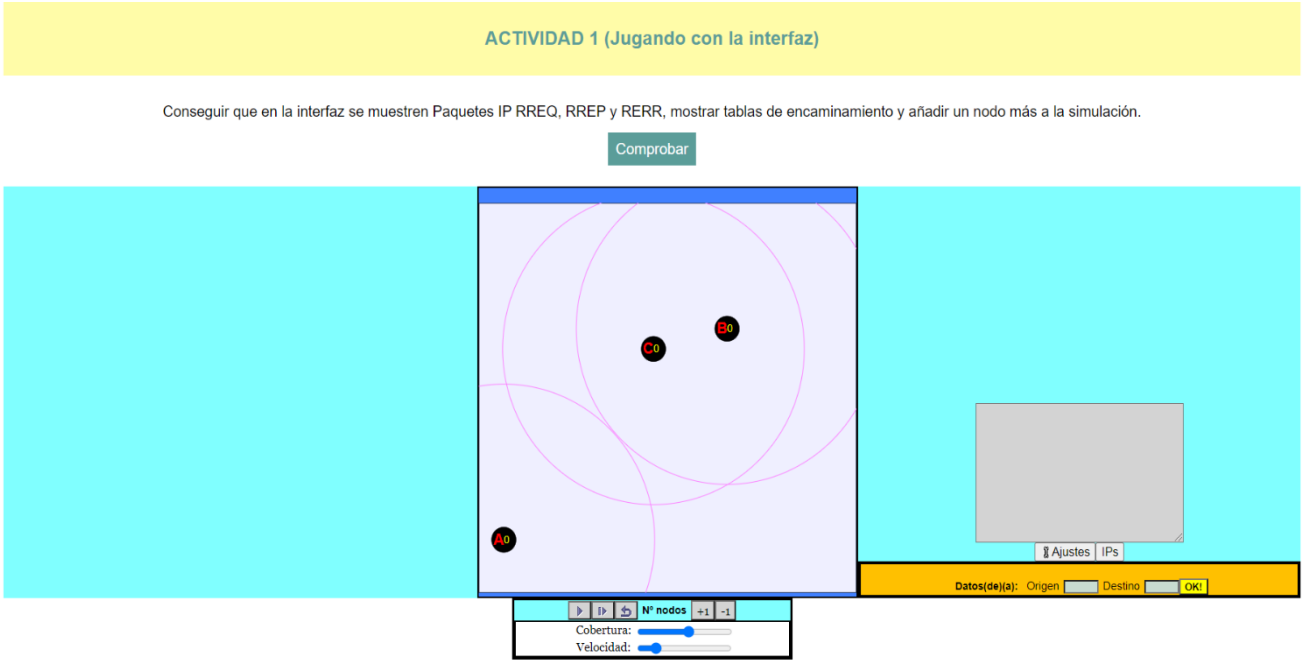


Figura 118: Primer vistazo a las actividades del simulador. Fuente: Propia

Como se puede apreciar, existe un botón verde donde se puede leer “Comprobar”.

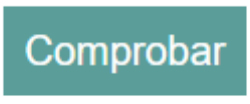


Figura 119: Botón para comprobar si la actividad se ha realizado o no correctamente. Fuente: Propia

Este botón tiene asociado todo lo necesario para navegar entre actividades, es decir, el botón tiene definido todas las comprobaciones necesarias para saber si un ejercicio realizado es correcto, y así pasar a la siguiente tarea.

Al comenzar, la interfaz muestra la actividad 1. Esta tarea consiste en hacer que el alumno sea capaz de hacer aparecer las imágenes que simulan el formato de los diferentes mensajes que AODV utiliza para comunicarse, las tablas de enrutamiento, además de añadir un nodo más a la simulación.

Para que el alumno pueda avanzar a la actividad 2, se deben completar todas las tareas descritas previamente.

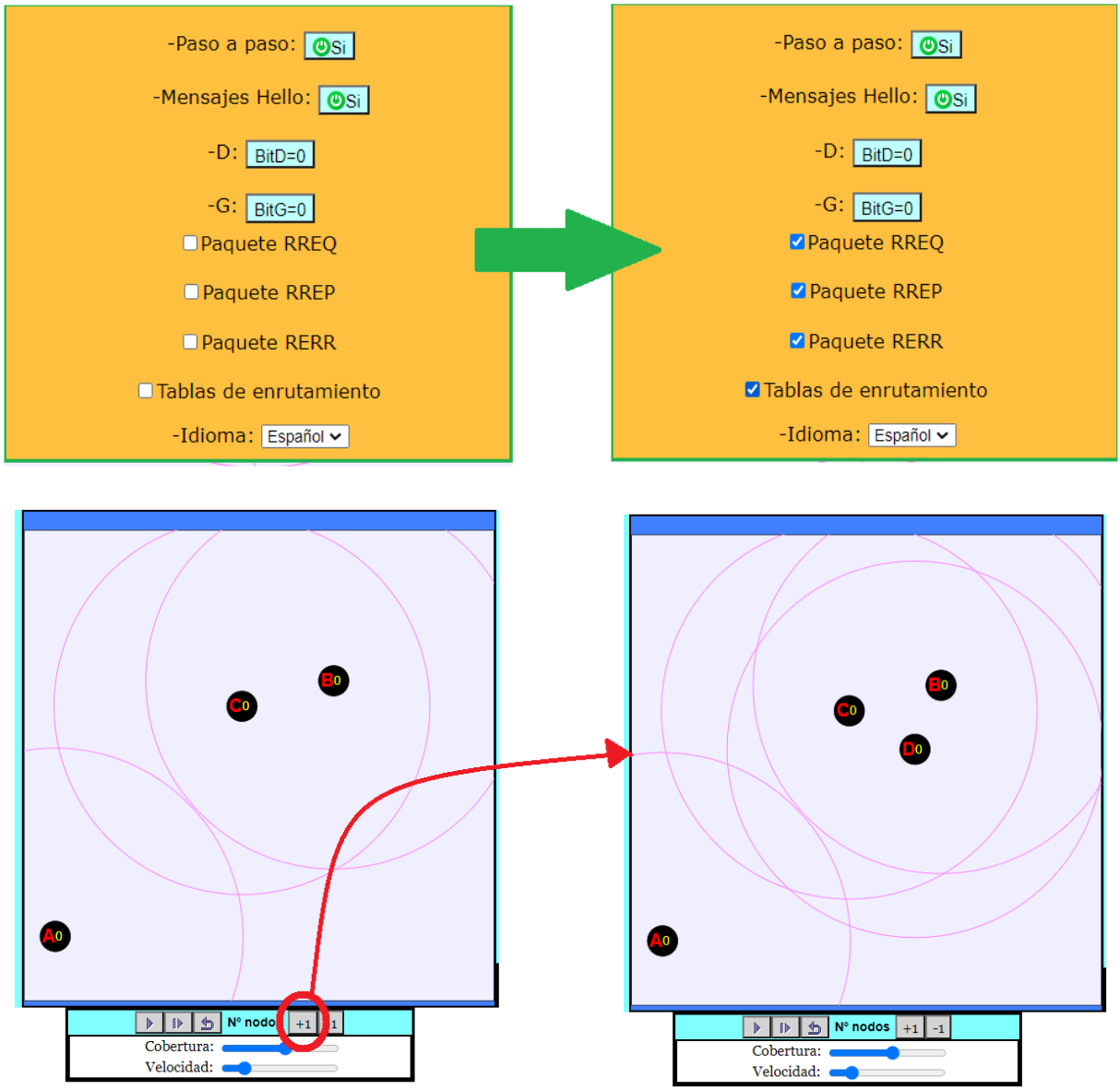


Figura 120: Pasos para solucionar la Actividad 1. Fuente: Propia

Una vez realizadas las tareas de esta primera actividad, se debe pulsar sobre el botón “Comprobar”. Esto hará que se muestre una ventana emergente indicando si la tarea se ha realizado o no correctamente.

Figura 121: Mensaje de respuesta incorrecta. Fuente: Propia

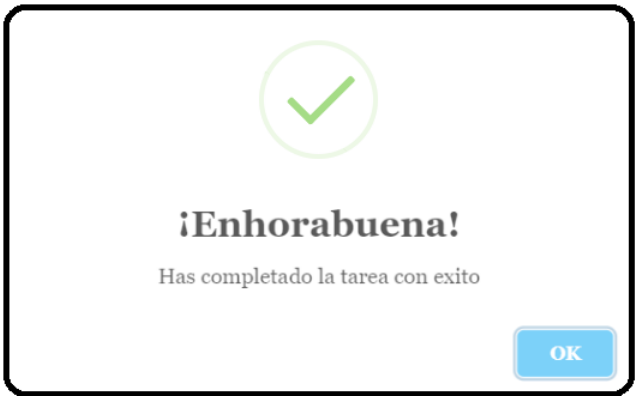


Figura 122: Mensaje de respuesta correcta. Fuente: Propia

Figura 123: Mensaje de alerta. Fuente: Propia

Estos mensajes aparecen en la comprobación de todas las actividades.

Cuando se realiza correctamente la actividad 1, se accede a la actividad 2.

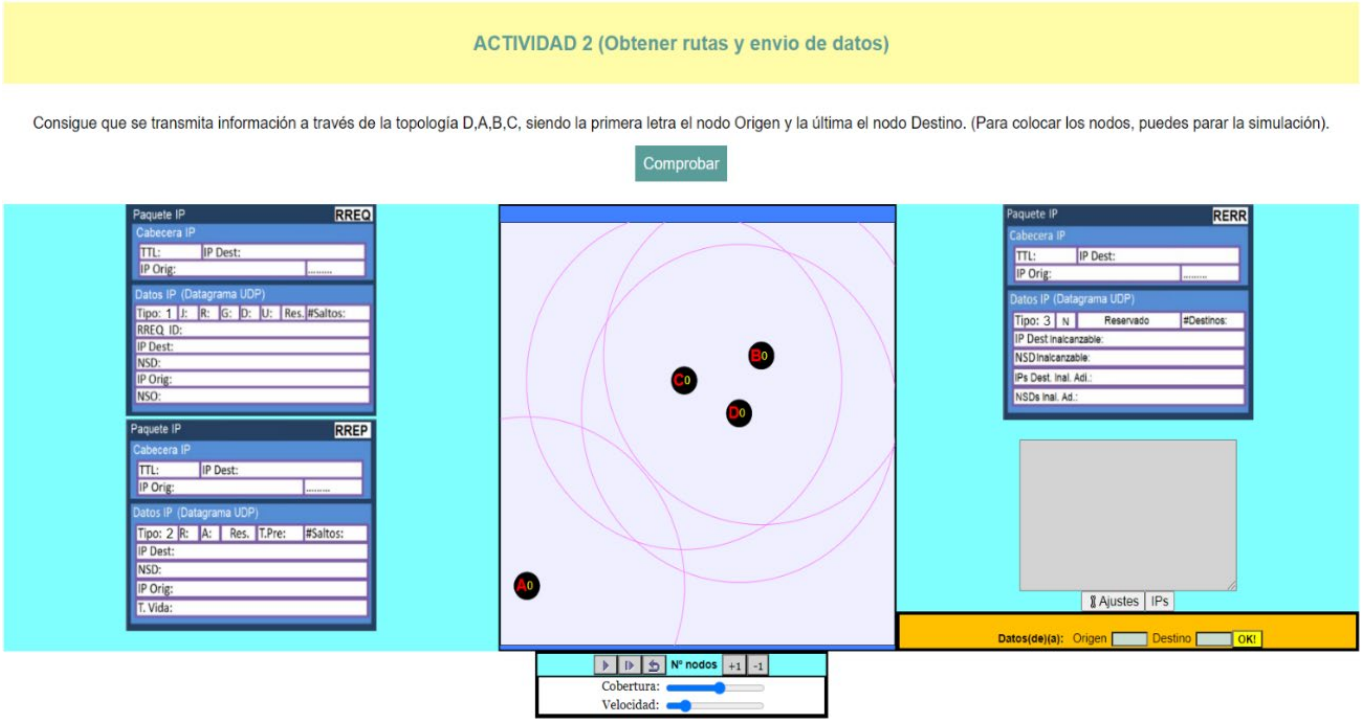


Figura 124: Interfaz mostrada cuando se alcanza la Actividad 2. Fuente: Propia

En esta tarea, el alumno deberá aprender a formar topologías arrastrando los nodos por el espacio donde estos se mueven. El simulador indicará una topología aleatoria formada entre 4 nodos, siendo el alumno el encargado crearla y establecer rutas entre el nodo origen y destino indicado.

Considerando, como ejemplo, el ejercicio de la figura 124, el simulador indica que hay que formar la topología D, A, B, C.

Para formar esta topología, el alumno debe pulsar sobre los nodos y arrastrarlos

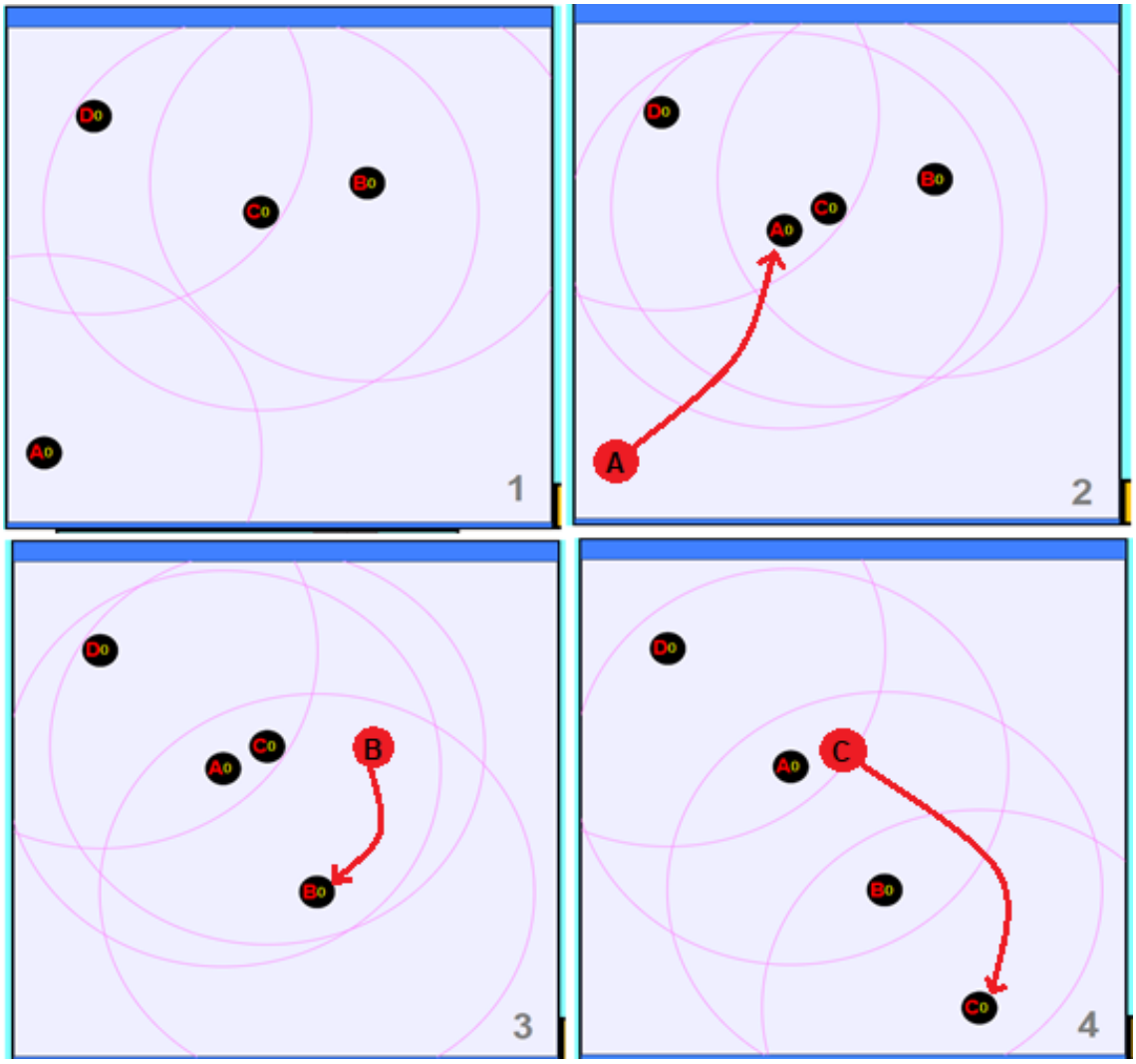


Figura 125: Representación gráfica del proceso de formación de topologías arrastrando los nodos. Fuente: Propia

Una vez que la topología está formada, tal y como se muestra en el cuarto paso de la imagen superior, hay que hacer que los nodos extremos consigan comunicarse. Para ello, tal y como se explicó en el desarrollo de la simulación, se utiliza el espacio de comunicación, indicando nodo origen y destino.

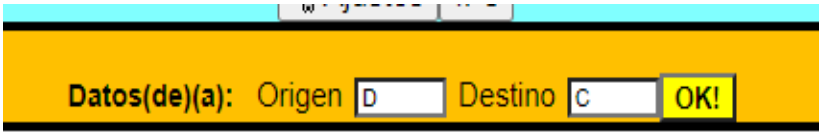


Figura 126: Utilización del espacio de comunicación para indicar nodo origen y destino. Fuente: Propia

Una vez introducidos estos campos, se pulsa sobre el botón “OK!” y comenzará la fase de descubrimiento de ruta. En este momento, solo se tendrá que ir viendo los diversos pasos que está llevando a cabo el protocolo AODV para que el nodo origen pueda enviar datos al nodo destino, pulsando sobre el botón PLAY cada vez que el simulador muestre una explicación, ya que se irá pausando para que se vaya leyendo lo que está ocurriendo en cada momento.

Una vez que ha establecido la comunicación, el espacio de narración de la interfaz mostrará el siguiente mensaje:

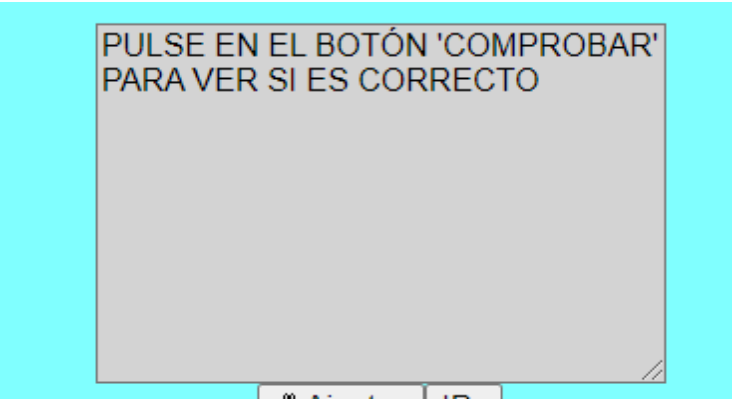


Figura 127: Espacio de narración indicando que se ha finalizado la tarea y se requiere comprobación. Fuente: Propia

En este punto, hay que pulsar en el botón “Comprobar” para saber si se realizó correctamente la tarea o no.

Si no es correcto, el simulador generará otra topología aleatoria y se tendrá que volver a repetir la tarea. En caso contrario, se pasará a la actividad 3.

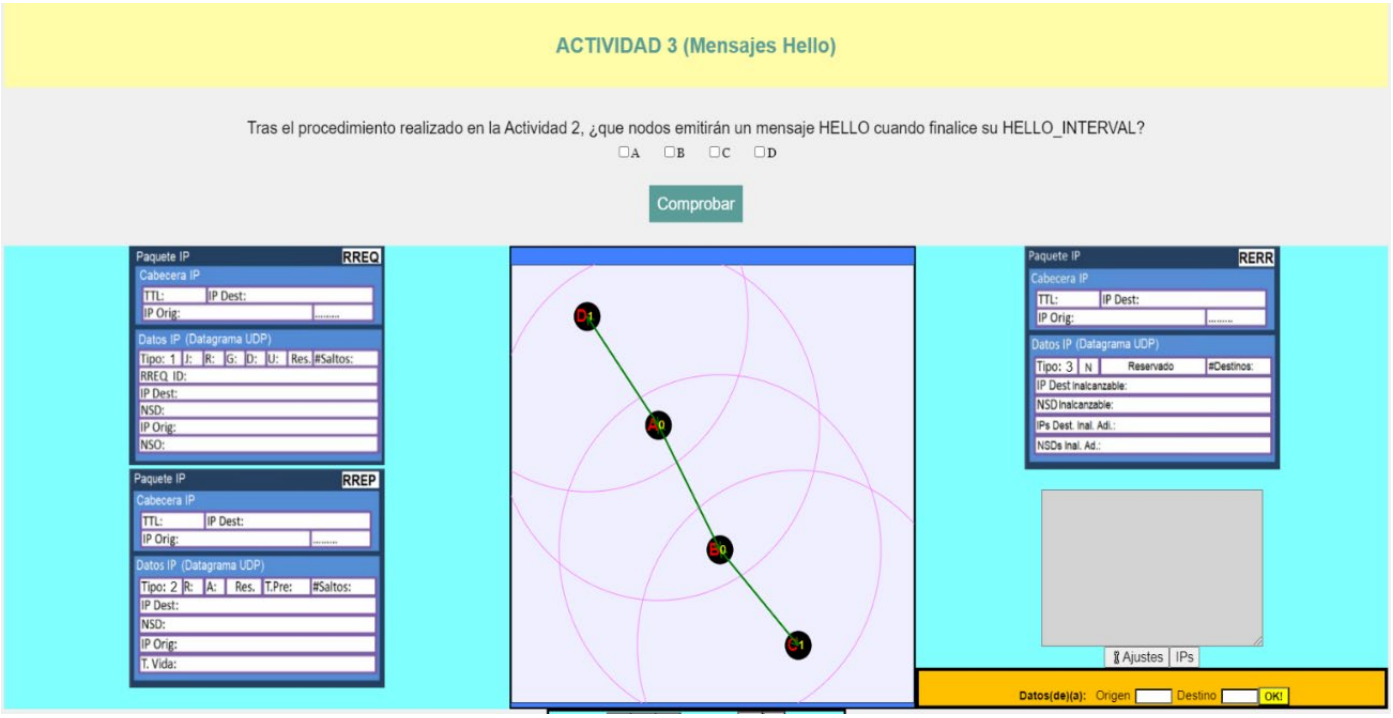


Figura 128: Interfaz mostrada cuando se alcanza la actividad 3. Fuente: Propia

Esta tarea tiene la finalidad de mostrar cómo se generan los mensajes HELLO una vez que los nodos poseen rutas activas. Antes de mostrar esto, el simulador hace una pequeña pregunta “trampa”. Esta consiste en indicar que nodos deberían enviar un mensaje HELLO justo después de haberse descubierto las rutas entre estos.

Para responder a esta pregunta, se debe conocer la teoría, la cual indica que cuando un nodo registra una ruta activa en su tabla de enrutamiento, comienza a correr su contador Hello. Si durante dicho contador, ese nodo difunde un mensaje RREQ, al finalizar el contador, el nodo no emitirá un mensaje HELLO. [19]

Sabiendo esto, en el ejercicio anterior, cuando se estuvieron descubriendo rutas entre nodos, habría ocurrido lo siguiente:

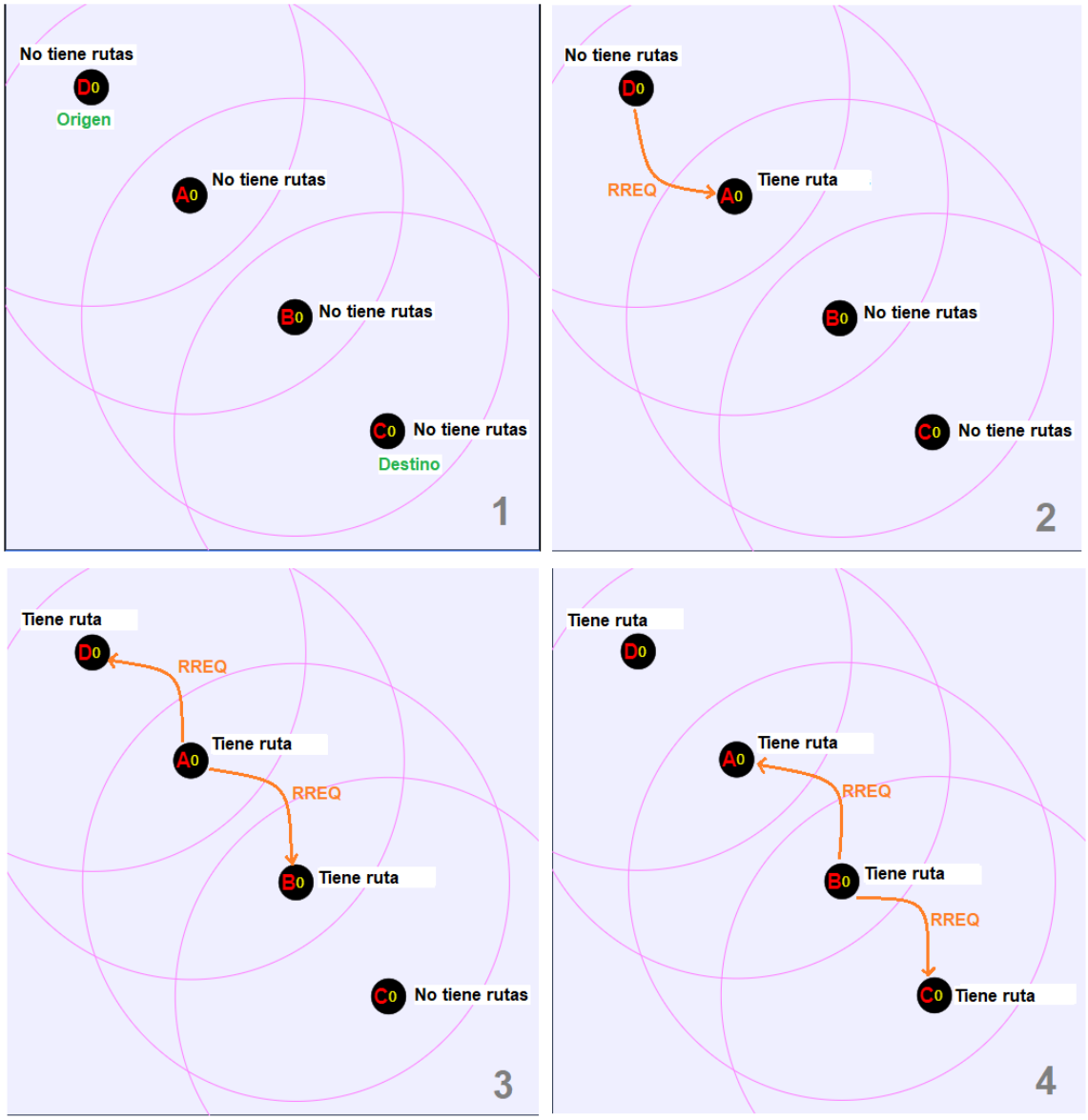


Figura 129: Representación gráfica del proceso desarrollado durante la Actividad 2. Fuente: Propia

Como se puede observar, los únicos nodos que emiten un RREQ teniendo ya rutas son el nodo A y B. Por lo tanto, los nodos que emitirían un mensaje HELLO cuando finalice su contador serían el D y C. Todo esto se resume diciendo que únicamente emitirían un mensaje HELLO el nodo origen y el nodo destino.

Por lo tanto, tras marcar las opciones correctas, se pulsará en el botón “Comprobar”.

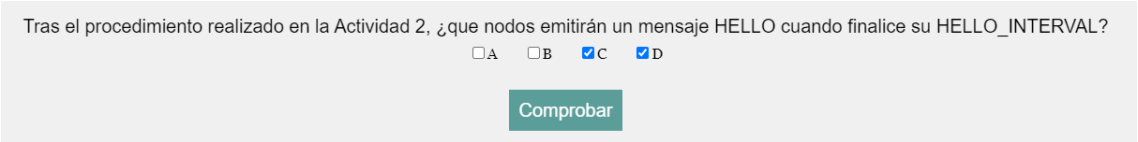


Figura 130: Para resolver la Actividad 3. Fuente: Propia

Una vez comprobado y siendo correcto, el simulador nos indica que realicemos lo siguiente antes de continuar con la siguiente tarea:

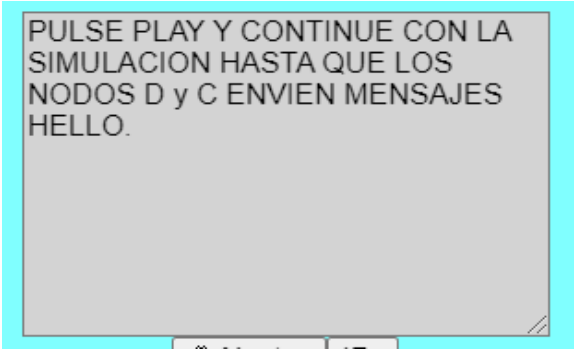


Figura 131: Indicación del espacio de narración para comprobar cómo se realiza el envio de mensajes HELLO. Fuente: Propia

Como indica el espacio de narración, se debe continuar con la simulación hasta que los nodos emitan el mensaje HELLO. Con esto se pretende forzar al alumno a ver cómo se emite este tipo de mensaje antes de continuar.

Una vez realizada dicha tarea, aparecerá lo siguiente:

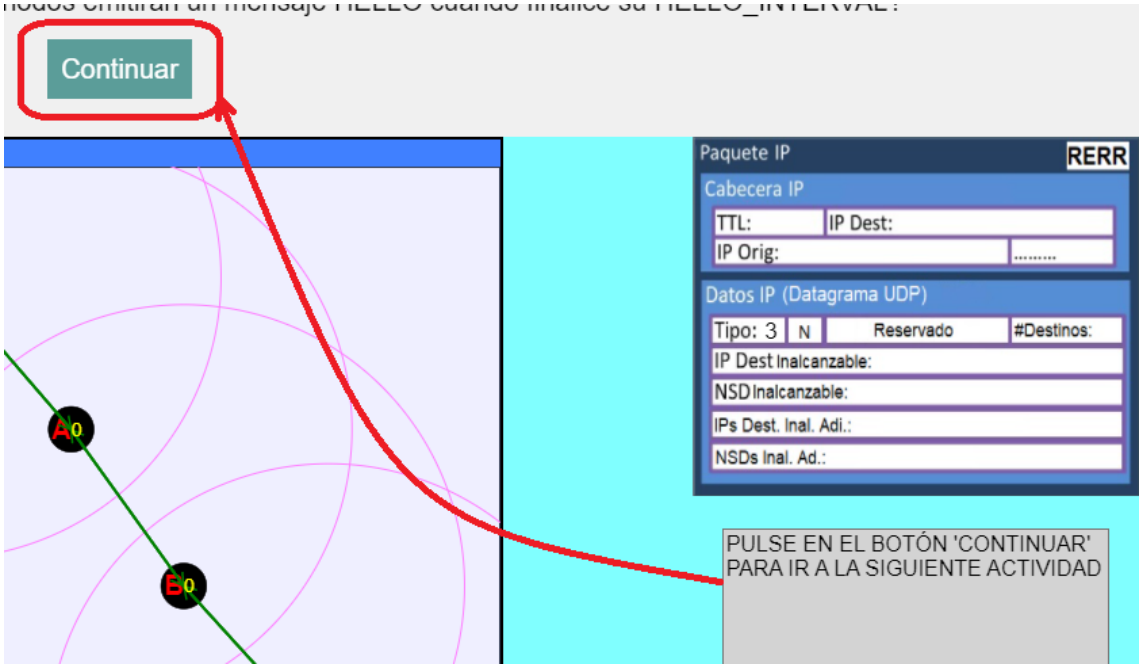


Figura 132: Indicación para acceder a la Actividad 4. Fuente: Propia

Cuando se pulsa sobre “Continuar” se accederá a la última actividad.

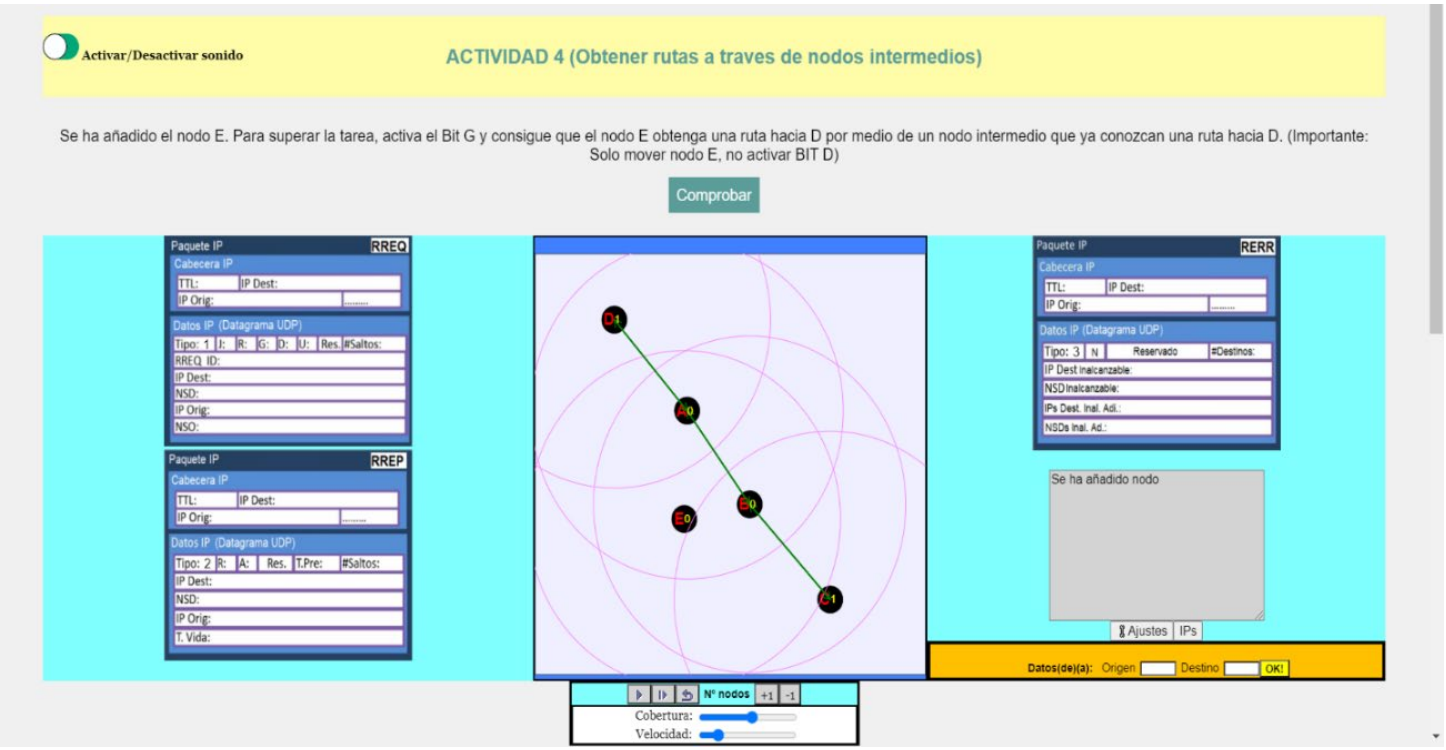


Figura 133: Interfaz mostrada cuando se alcanza la Actividad 4. Fuente: Propia

El objetivo de esta tarea es que el alumno conozca como los nodos pueden obtener rutas hacia un nodo destino por medio de un nodo intermedio que posee una ruta válida hacia dicho destino.

Para ello, el simulador añade un quinto nodo a la simulación e indica que, para superar esa tarea, el alumno tiene que conseguir comunicar el nuevo nodo, con el nodo D (Este puede cambiar dependiendo de la topología aleatoria que el simulador nos proponga en el ejercicio 2). Las únicas condiciones que impone la actividad es que, hay que activar el bit G y hay que conseguir que el nodo recién añadido obtenga la ruta hacia D a través de un nodo intermedio y no directamente con un RREP emitido desde el nodo D.

Para ello, únicamente se debe arrastrar el nuevo nodo, dejando este en cobertura con cualquiera de los otros nodos, pero nunca en cobertura directa con el nodo D (Es importante no arrastrar ninguno de los cuatro nodos que ya estaban situados, solo arrastrar nodo recién añadido).

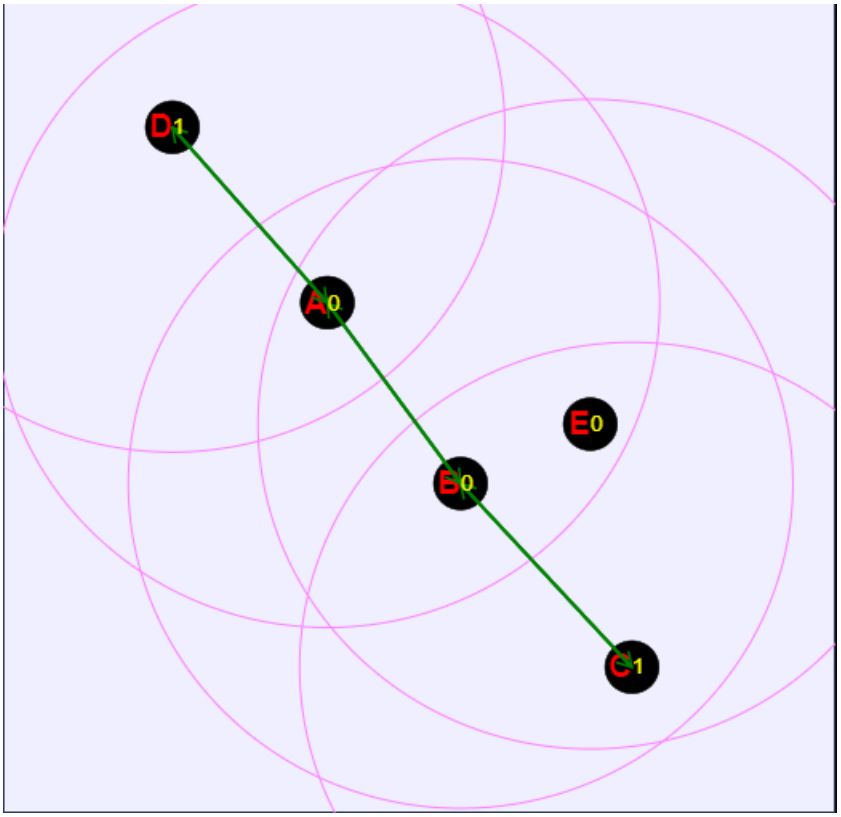


Figura 134: Ejemplo de topología válida para realizar esta actividad correctamente. Fuente: Propia

Una vez situado correctamente el nodo E y activado el Bit G, se introduce en el espacio de comunicación el nodo origen y destino.

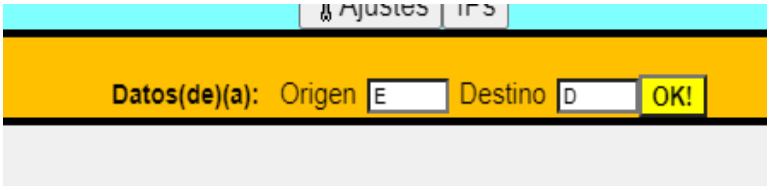


Figura 135: Indicar en el espacio de comunicación el nodo origen y destino. Fuente: Propia

Cuando el nodo E emite el mensaje RREQ, estos son recibidos por A, B y C. Como se explicó anteriormente, existe un algoritmo en la simulación que permite que únicamente responda al RREQ el nodo que mejor camino hacia el destino pueda ofrecer. Como el nodo A puede ofrecer el mejor camino, aparecerá el siguiente mensaje:

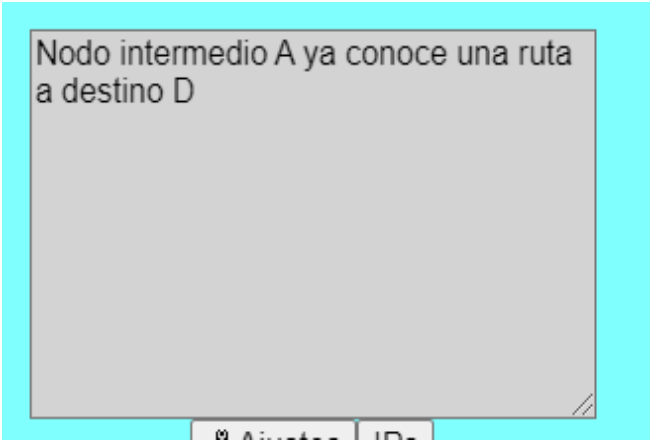


Figura 136: Información contada por el espacio de narración indicando que un nodo intermedio posee una ruta. Fuente: Propia

El nodo A emitirá un mensaje RREP hacia el origen, proporcionando la ruta hacia el destino D, y además emitirá un mensaje RREP hacia el destino, proporcionando la ruta hacia el origen E, ya que el Bit G=1.

Una vez se ha realizado todo este proceso, el espacio de narración indica que se pulse sobre el botón “Comprobar” para ver si la tarea se ha realizado correctamente.

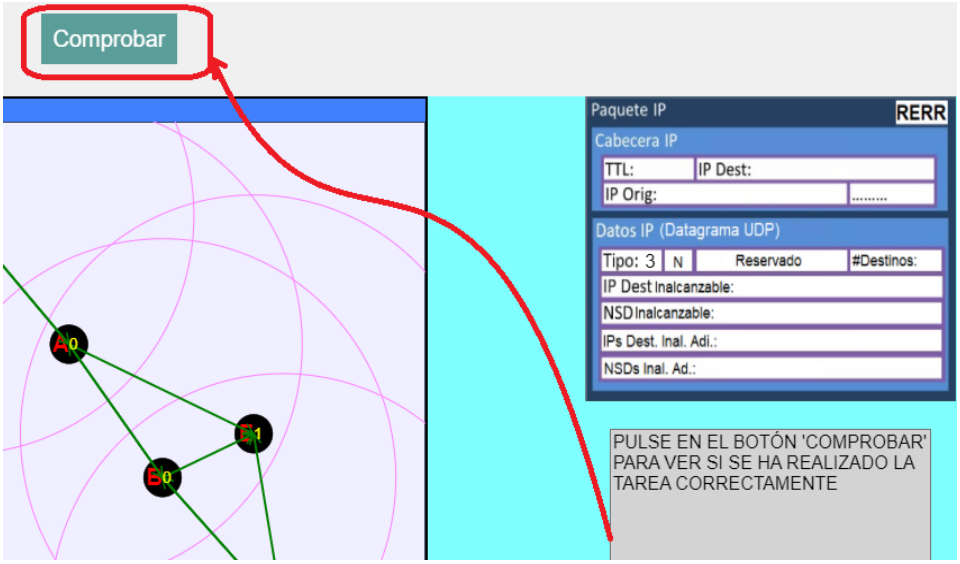


Figura 137: Espacio de narración indica que se debe comprobar el ejercicio realizado para saber si es correcto o no. Fuente: Propia

Si no es correcta, se tendrá que repetir de nuevo este ejercicio.

Si la tarea se ha realizado correctamente, el alumno estará preparado para acceder a la simulación AODV completa.

5.5 Manipulación e integración SCORM

Una vez que la simulación ha sido desarrollada en la herramienta EJS, se exportará el proyecto como un paquete SCORM.

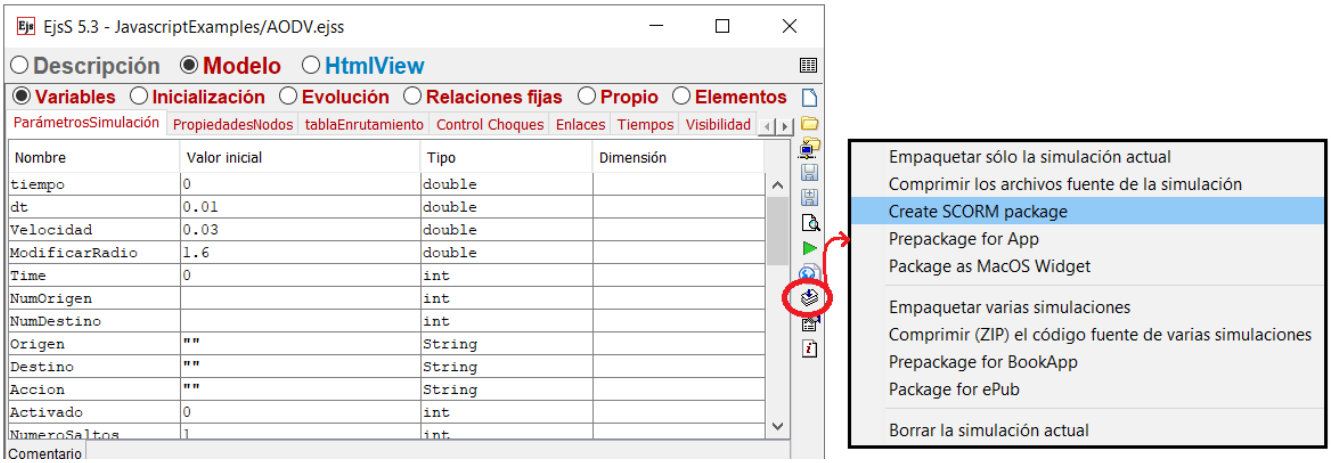


Figura 138: Como exportar un proyecto SCORM desde la herramienta EJS. Fuente: Propia

Al hacer esto, se genera un fichero .zip que contendrá un archivo llamado imsmanifest.xml.

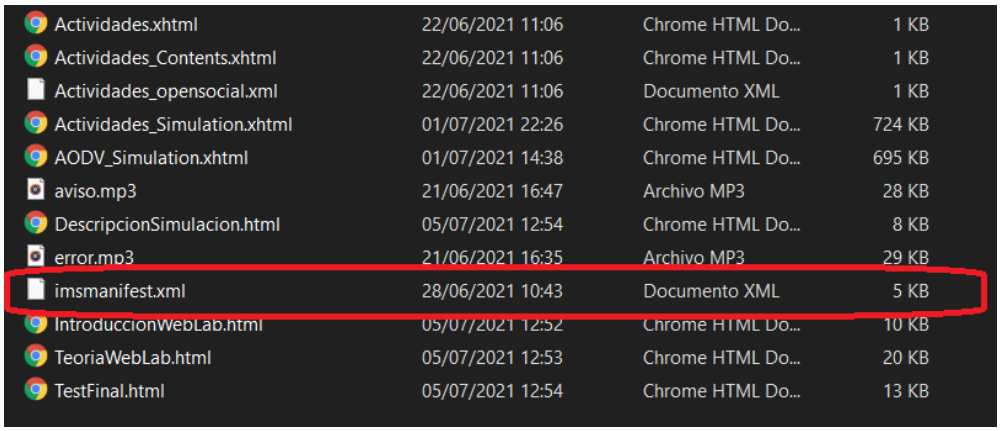


Figura 139: Ubicación del archivo imsmanifest.xml generado. Fuente: Propia

En este archivo, se establece la información necesaria para que el LMS interprete el contenido de forma adecuada. Además, indica la secuenciación de los SCOs dentro del módulo SCORM.

```
    </item>
    <item identifier="SCO2" identifierref="RES2" isvisible="true">
      <title>Teoria</title>
    </item>
    <item identifier="SCO3" identifierref="RES3" isvisible="true">
      <title>Explicacion_Simulacion</title>
    </item>
    <item identifier="SCO4" identifierref="RES4" isvisible="true">
      <title>Actividades_Simulation</title>
    </item>
    <item identifier="SCO5" identifierref="RES5" isvisible="true">
      <title>Simulacion_Completa</title>
    </item>
    <item identifier="SCO6" identifierref="RES6" isvisible="true">
      <title>Test Final</title>
    </item>
    <imsss:sequencing>
      <imsss:controlMode flow="true" choice="true"/>
      <imsss:rollupRules rollupObjectiveSatisfied="true" rollupProgressCompletion="true" objectiveMeasureWeight="0"/>
    </imsss:sequencing>
  </organization>
</organizations>
<resources>
  <resource identifier="RES1" href="IntroduccionWebLab.html" adlcp:scormType="sco" type="webcontent">
    <file href="IntroduccionWebLab.html"/>
  </resource>
  <resource identifier="RES2" href="TeoriaWebLab.html" adlcp:scormType="sco" type="webcontent" xml:base="/">
    <file href="TeoriaWebLab.html" />
    <dependency identifierref="API-DEP" />
    <dependency identifierref="API-DEP1" />
  </resource>
  <resource identifier="RES3" href="DescripcionSimulacion.html" adlcp:scormType="sco" type="webcontent" xml:base="/">
    <file href="DescripcionSimulacion.html" />
    <dependency identifierref="API-DEP" />
    <dependency identifierref="API-DEP1" />
  </resource>
  <resource identifier="RES4" href="Actividades_Simulation.xhtml" adlcp:scormType="sco" type="webcontent" xml:base="/">
    <file href="Actividades_Simulation.xhtml" />
    <dependency identifierref="API-DEP" />
    <dependency identifierref="API-DEP1" />
  </resource>
  <resource identifier="RES5" href="AODV_Simulation.xhtml" adlcp:scormType="sco" type="webcontent" xml:base="/">
    <file href="AODV_Simulation.xhtml" />
```

Figura 140: Definición de la secuenciación de los diferentes SCOs desarrollados. Fuente: Propia

Posteriormente, dentro de cada una de las páginas HTML que componen el módulo, se incluirán varias librerías que permitirán integrar los SCOs dentro del LMS, ofreciendo a este información sobre las interacciones que los usuarios realizan en dichas páginas.

```
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Teoría WebLab</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href="_ejs_library/css/Estilos.css" rel="stylesheet" type="text/css"/>
        <script type="text/javascript" src="_ejs_library/scripts/principal.js"></script>
        <script type="text/javascript" src="_scorm/scripts/sweetalert2.all.min.js"></script>
        <script type="text/javascript" src="_ejs_library/scripts/APIWrapper.js"></script>
        <script type="text/javascript" src="_ejs_library/scripts/RTE.js"></script>
        <!--<script type="text/javascript" src="_scorm/scripts/uja_scorm_rte.js"></script>-->
    <script>
        var _scorm = new RTE("2004");
        var _scorm_terminated = false;
        var banco = '_ejs_library/html/TestInicial.xml';
            var numeroPr = 5;
            var time = 0;
            var prAleat = 'si';
            var opAleat = 'si';
            var comunicaciones = 'si';
        function Iniciar() {
```

Figura 141: Inclusión de las diferentes librerías utilizadas para dar soporte a SCORM. Fuente: Propia

Una de estas librerías es la RTE.js, la cual fue desarrollada por Ildefonso Ruano Ruano.

Para conseguir mostrar ventanas emergentes en el módulo SCORM, se ha utilizado la librería SweetAlert2[20]. Esta es una librería JavaScript que permite crear ventanas emergentes con un diseño profesional y fácil de personalizar e implementar, además es compatible con la mayoría de navegadores web.

Figura 142: Ejemplo de ventana emergente generada gracias a la librería SweetAlert2. Fuente: Propia

Algunos de los SCOs desarrollados contendrán preguntas tipo test sobre la teoría explicada con el fin de evaluar la comprensión de los usuarios que acceden al WebLab.

Para añadir estas preguntas tipo test a los SCOs se han utilizado varias librerías desarrolladas por Sergio Díaz Fuentes:

- principal.js
- creaPreguntas2004.js
- creaPreguntasSinCom.js

Para incluir un test en una determinada página, primero es necesario introducir en dicha página la librería principal.js e indicar la ubicación del banco de preguntas a incluir.

```
<head>
    <title>Teoría WebLab</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script type="text/javascript" src="_ejs_library/scripts/principal.js"></script>
    <script type="text/javascript" src="_ejs_library/scripts/APIWrapper.js"></script>
    <script type="text/javascript" src="_ejs_library/scripts/RTE.js"></script>
    <!--<script type="text/javascript" src="_scorm/scripts/uja_scorm_rte.js"></script>-->

<script>
    var _scorm = new RTE("2004");
    var banco = '_ejs_library/html/TestInicial.xml';
        var time = 0;
        var prAleat = 'si';
```

Figura 143: Referenciación hacía la librería principal y el test. Fuente: Propia

El WebLab cuenta con dos test diferentes, un test inicial de teoría y un test de evaluación final. Estos test se han obtenido reutilizando algunas de las preguntas ya existentes dentro de un banco de preguntas en la plataforma Docencia Virtual, además de añadir nosotros mismos algunas preguntas extra.

Gracias a la inicialización de todo esto y el desarrollo de unos determinados estilos con CSS, se obtendrán preguntas tipo test como las que aparecen a continuación:

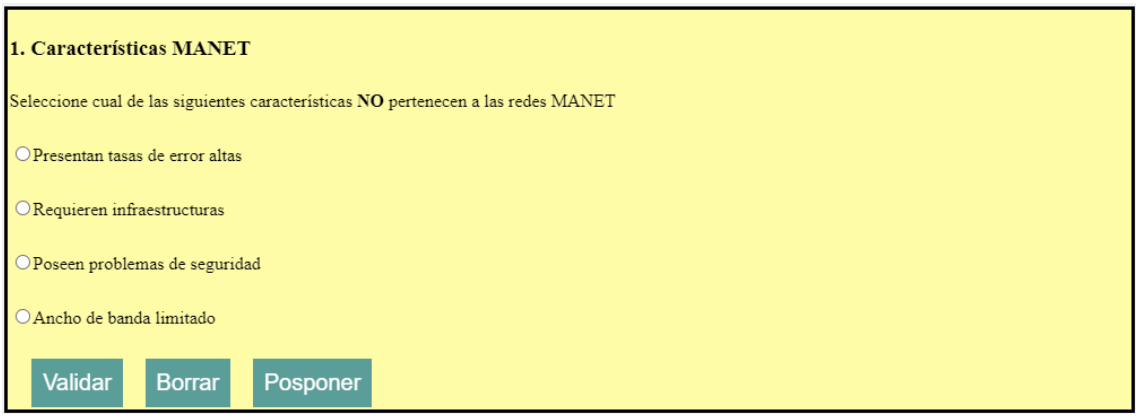


Figura 144: Aspecto que presentan las preguntas que se incluyen dentro del módulo SCORM desarrollado. Fuente: Propio

Además, se proporcionará un visor que permitirá al usuario conocer durante la ejecución del test que puntuación está obteniendo:

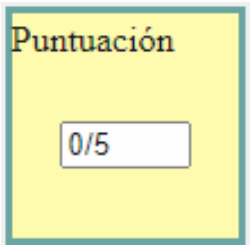


Figura 145: Visor de puntuación. Fuente: Propia

Sobre la parte de la librería creaPreguntas2004.js encargada de mostrar la puntuación obtenida en una respuesta en concreto se ha realizado una pequeña modificación ya que esta utiliza la librería smoke.js para las ventanas emergentes, presentando el siguiente aspecto:

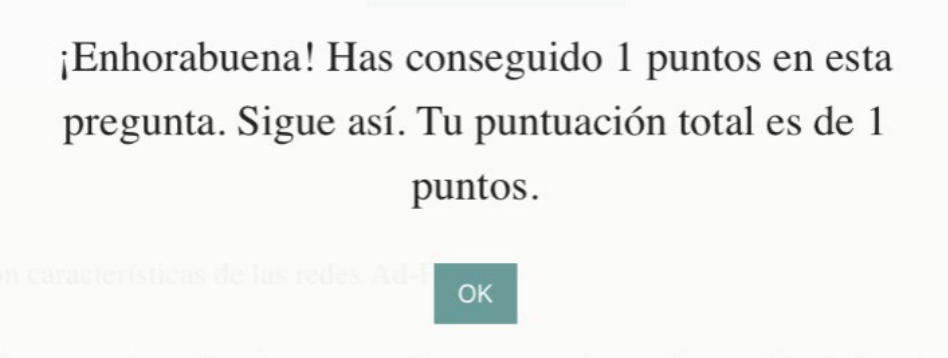


Figura 146: Ejemplo de ventana emergente generada por la librería smoke.js. Fuente: Raúl Sánchez Salido

Como este proyecto ha utilizado SweetAlert2 para las ventanas emergentes, se ha realizado la siguiente modificación

```
    }
    //Actualizamos la puntuación y mostramos la siguiente pregunta
    puntuacion = Math.round(puntuacion * 100) / 100;
    actualiza(puntuacion, total);

    if (puntos > 0)
        Swal.fire({

                title: "¡Enhorabuena!",
                icon: "success",
                html: "Has conseguido <b>" +puntos+ "</b> puntos en esta pregunta. Sigue así.",
                });
        //smoke.alert("¡Enhorabuena "+_scorm.getLearnerName().split(" ")[0]+"! Has conseguido "+puntos+ " puntos en esta pregunta. Sigue así.");
    else
        Swal.fire({

                title: "¡Ups!",
                icon: "error",
                text: "Has fallado esta pregunta... Pero no te desanimes. ¡A por las siguientes preguntas!",
                });
        //smoke.alert("¡No te desanimes "+_scorm.getLearnerName().split(" ")[0]+"! ¡A por las siguientes preguntas!");

    muestra(ent);
}
```

Figura 147: Modificaciones realizadas sobre la librería creaPreguntas2004.js. Fuente: Propia

En contraste con la Figura 146, la utilización de SweetAlert2 proporciona ventanas emergentes con el siguiente aspecto:

Figura 148: Ventana emergente con información de puntuación empleando la librería SweetAlert2. Fuente: Propia

Figura 149: Ventana emergente indicando fallo empleando la librería SweetAlert2. Fuente: Propia

Tras haberse realizado todas estas modificaciones, el proyecto es subido a ILIAS.

5.6 Desarrollo del módulo de aprendizaje SCORM

El módulo desarrollado consta de 6 SCOs (páginas), las cuales introducirán al alumno progresivamente hacía la explicación del protocolo AODV, objetivo principal de este bloque de aprendizaje. A continuación, se explicará detalladamente cómo se desarrolla el contenido de este módulo:

- Entrando en la Página 1:

Cuando el alumno entra en la página 1, este verá una ventana emergente diferente dependiendo del número de veces que se haya accedido a dicha página.

Figura 150: Ventana emergente de cuando se accede por primera vez en la página 1. Fuente: Propia

Figura 151: Ventana emergente de cuando se accede por más de una vez en la página 1, mostrando el resultado de la última vez que el usuario la página. Fuente: Propia

El contenido de la página 1 es simplemente una introducción a lo que se va a explicar en el WebLab, es decir, muestra la estructura en la que se desarrollará el módulo de aprendizaje, así como los objetivos que se buscan con la realización del mismo.

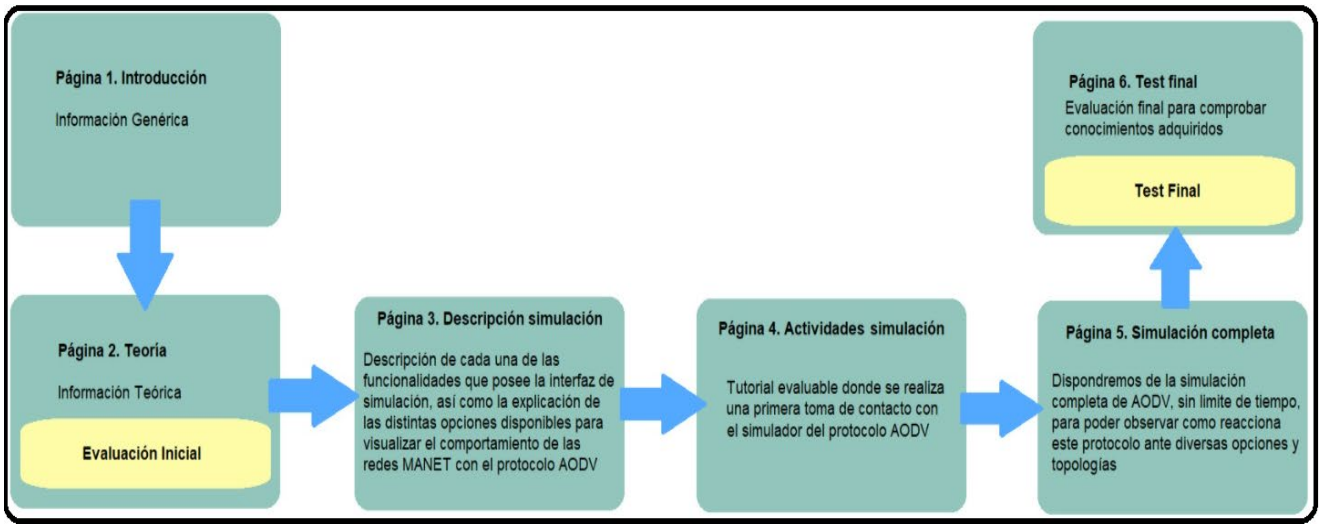


Figura 152: Representación de la estructura en la que se va a desarrollar el WebLab. Fuente: Propia

Una vez comprendida esta primera página, se pulsará sobre el botón que aparece al final de la página, permitiendo acceder a la siguiente.

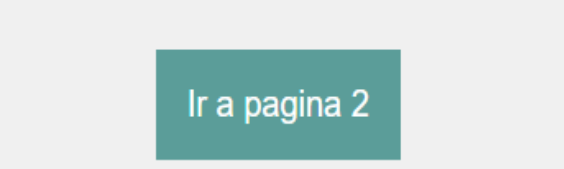


Figura 153: Botón para acceder a la página 2. Fuente: Propia

- Entrando en la Página 2:

Cuando se accede por primera vez a esta página, aparecerá una ventana emergente similar a la que apareció cuando se accede por primera vez a la página 1. Sin embargo, como esta página contiene un test inicial, si se accede más de una vez a la página 2, además de indicar si se consiguió aprobar la última vez que se accedió, muestra que nota se obtuvo.

Figura 154: Ventana emergente mostrada en la página 2 cuando se accede más de una vez a esta. Fuente: Propia

En cuanto al contenido de esta página, se trata de una breve teoría sobre lo que se debe saber previamente antes de explicar el protocolo AODV. Primeramente, se explican qué son las redes Ad-Hoc, seguido de lo que son las redes MANET para finalmente explicar en qué consiste AODV.

Una vez se haya leído todo el contenido de esta página, empleando únicamente la información que aparece en ella, el usuario podrá realizar un test de 5 preguntas que aparecerá al final de dicha página al pulsar este botón:

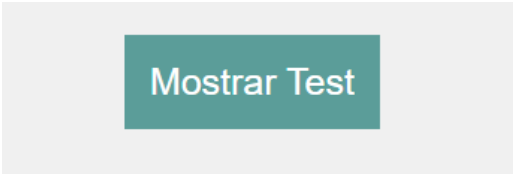


Figura 155: Botón para mostrar el test relacionado con la teoría explicada en la página 2. Fuente: Propia

Una vez completado el test, se pulsa sobre el botón “Mostrar resultados” que aparece más abajo.

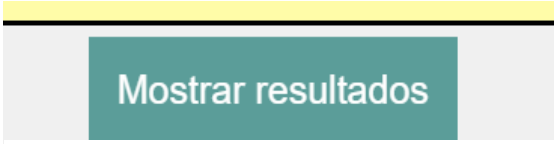


Figura 156: Botón para mostrar los resultados del test realizado. Fuente: Propia

Al pulsarlo, el SCO se encargará de calcular, en base a los resultados obtenidos, si el alumno ha aprobado o no el test. Posteriormente, se ofrecerá una tabla la cual contiene un resumen de los resultados obtenidos en dicho test:

Resultados del Test	
Alumno:	José Ramón
Fecha de inicio:	2021-07-18T14:25:40
Fecha de finalización:	2021-07-18T14:43:58
Tiempo empleado en el test:	00:18:18
Contador de interacciones:	5
Estado de terminación:	Completado
Calificación:	Aprobado
Puntuación mínima:	0 puntos
Puntuación máxima:	5 puntos
Puntuación obtenida:	4 puntos
P1. Respuesta, latencia y puntos:	rrep,rreq,hello,rerr || PT00H17M30.26S. || 1 puntos
P2. Respuesta, latencia y puntos:	a,a,b,d,b,b,b,c || PT00H00M30.82S. || 1 puntos
P3. Respuesta, latencia y puntos:	a,c,d,e || PT00H00M08.75S. || 1 puntos
P4. Respuesta, latencia y puntos:	Requieren_infraestructuras || PT00H00M05.29S. || 1 puntos
P5. Respuesta, latencia y puntos:	No contestada || Sin latencia || 0 puntos

Figura 157: Tabla informativa sobre los resultados obtenidos en el test que se acaba de realizar. Fuente: Propia

Después de observar esto, se podrá acceder a la siguiente página.

- Entrando en la página 3:

Al igual que las otras páginas, al acceder a esta aparece una ventana emergente indicándonos cuántas veces hemos accedido a ella y si está aprobada o no.

En ese SCO viene explicado todo lo necesario para aprender a utilizar el simulador AODV, así como las diferentes opciones con las que se puede configurar el mismo.

Lo primero que aparece en esta página es una imagen de la interfaz que presenta el simulador, indicando en qué consiste cada uno de sus componentes.

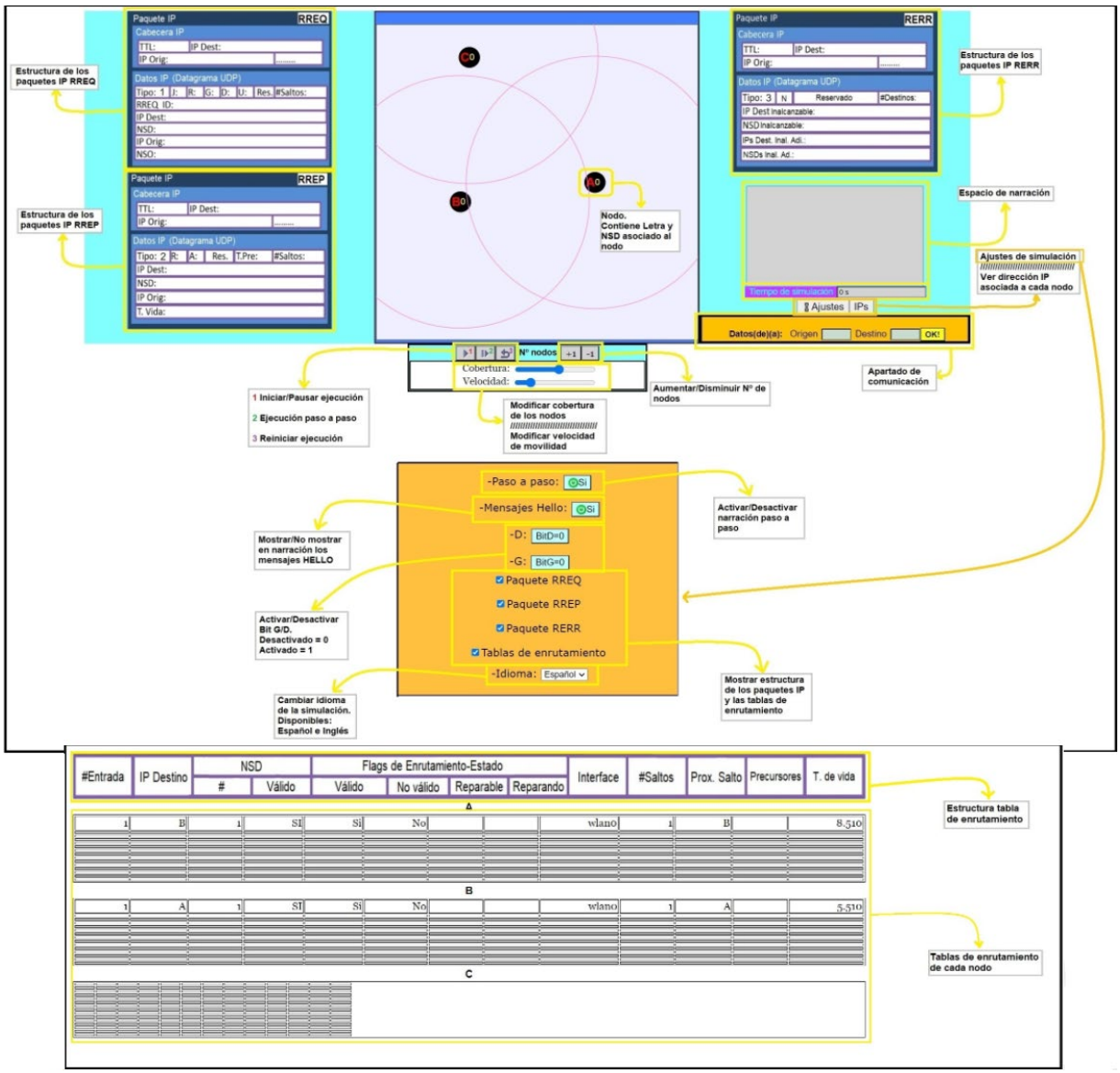


Figura 158: Imagen descriptiva sobre los diferentes componentes que posee la interfaz del simulador desarrollado. Fuente: Propia

Además de esta imagen, en la página aparece un video, el cual se encarga de explicar cómo se utiliza el simulador AODV antes de acceder al mismo.

Figura 159: Captura del video desarrollado para explicar el funcionamiento del simulador. Fuente: Propia

Una vez se haya entendido el contenido de esta página, se podrá acceder a la siguiente.

- Entrando en la página 4:

El alumno accede a las tareas del simulador AODV. Esta página no es más que una especie de tutorial que supondría una primera toma de contacto del alumno con el simulador antes de acceder al simulador AODV completo. No existe límite de tiempo, ni se puede suspender. El objetivo es que el alumno complete todas las tareas, ya que, si no, no podrá acceder a la siguiente página.

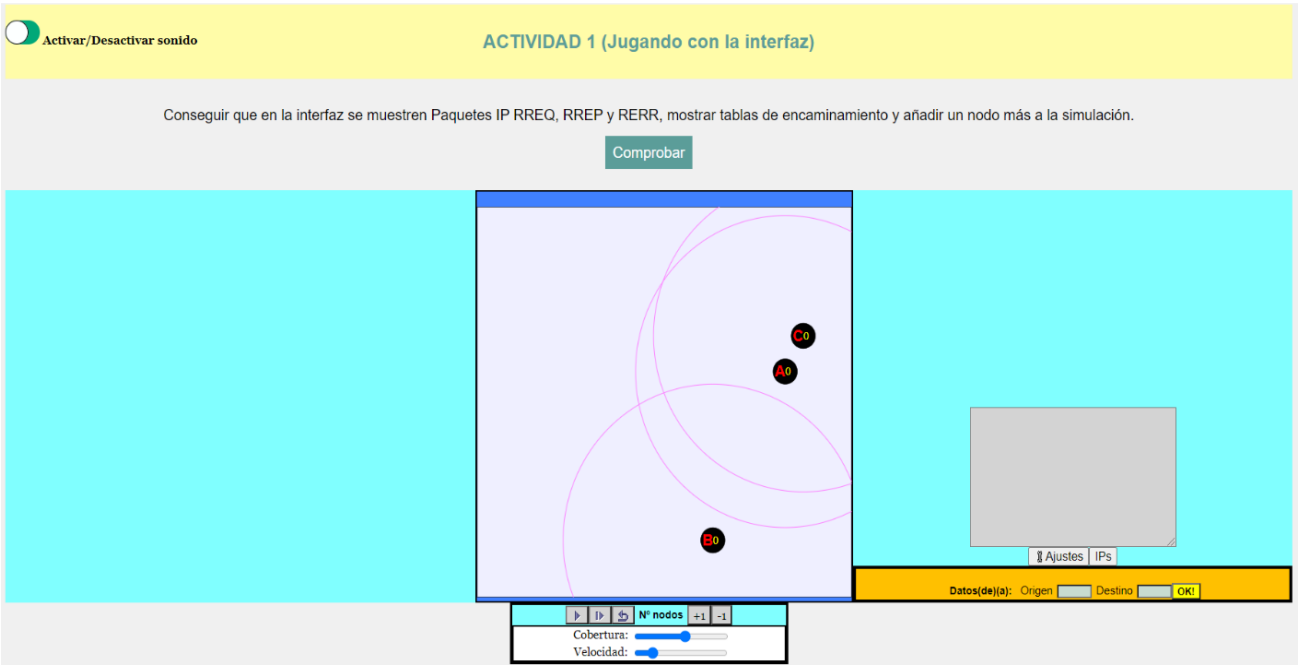


Figura 160: Actividades (tutorial) sobre el simulador AODV. Fuente: Propia

De este SCO, se le comunicará al LMS aquella información relacionada con el tiempo que tarda el alumno en completar todas las tareas, el idioma en el que se realizan (inglés o español) y si se realizaron con o sin sonido.

Una vez se han completado todas las tareas, aparecerá el botón que dará acceso a la siguiente página:

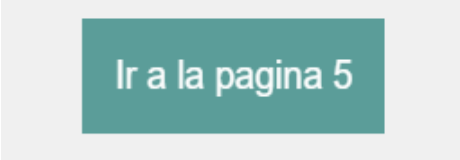


Figura 161: Botón para acceder a la página 5. Fuente: Propia

- Entrando a la página 5:

Es el SCO más importante del módulo SCORM ya que este alberga el simulador AODV completo. Puede ser utilizado sin límite de tiempo y no se puede suspender. El objetivo es que el alumno experimente con el protocolo AODV, observando cada uno de los pasos que este realiza a la hora de establecer comunicaciones, como es el formato de los mensajes empleados por el protocolo, cómo evolucionan las tablas de enrutamiento, que ocurre si se establece una u otra topología, o se activa el bit G… En resumen, que el alumno observe todas las posibilidades que tiene este protocolo de encaminamiento por medio del simulador desarrollado.

Una vez que el alumno decide avanzar a la sexta y última página, este SCO comunicará al LMS cuánto tiempo dedicó el alumno a experimentar con el simulador y el idioma utilizado para la simulación.

- Entrando a la página 6:

Cuando se accede a esta, volverá a aparecer la ventana emergente avisando si es la primera o n vez que se accede a la página, junto con el último status (Aprobado o suspenso) y puntuación obtenida.

Este último SCO contiene un test de 10 preguntas, el cual aparece al pulsar el botón “Mostrar test”

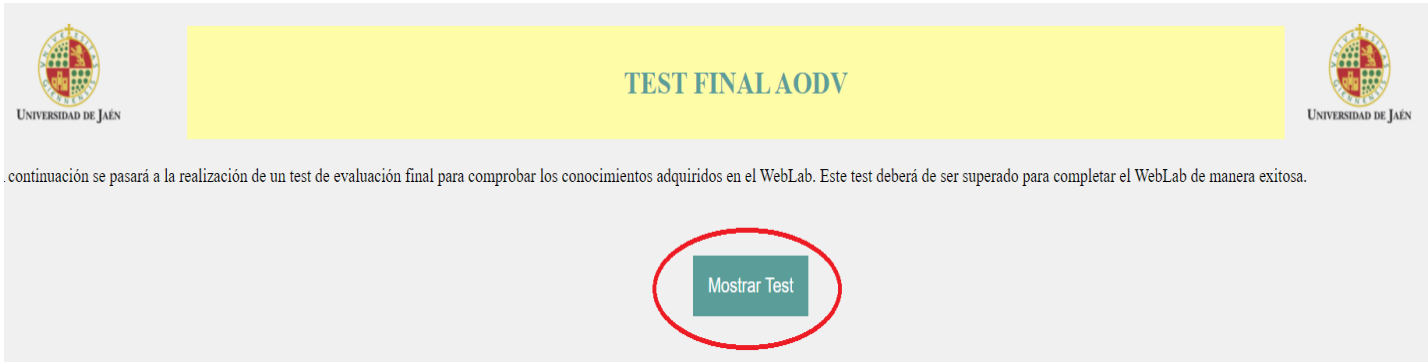


Figura 162: Contenido de la página 6. Fuente: Propia

Cuando se pulsa, el usuario accederá a las 10 preguntas finales, todas ellas relacionadas con el tema visto a lo largo del módulo SCORM. Una vez se hayan respondido todas las preguntas, se podrán ver los resultados, tal y como se hizo en la página 2.

Resultados del Test	
Alumno:	José Ramón
Fecha de inicio:	2021-07-18T18:27:19
Fecha de finalización:	2021-07-18T18:40:51
Tiempo empleado en el test:	00:13:32
Contador de interacciones:	10
Estado de terminación:	Completado
Calificación:	Aprobado
Puntuación mínima:	0 puntos
Puntuación máxima:	10 puntos

Figura 163: Tabla que proporciona el resumen de los resultados obtenidos en el test final. Fuente: Propia

Tras ver esta tabla, se podrá finalizar el WebLab pulsando sobre “Finalizar módulo”.

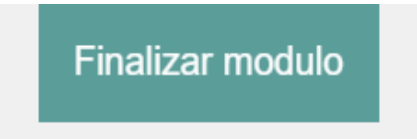


Figura 164: Botón para finalizar el WebLab. Fuente: Propia

El desarrollo de este WebLab está inspirado en otros dos módulos SCORM (OLSR[21] y VANET[22]) que fueron creados en 2018 por Raúl Salido Sánchez. En dichos módulos, se explican otros protocolos de enrutamiento diferentes a AODV para redes Ad-Hoc, pero ambos presentan una estructura y presentación similares. Tanto el WebLab que se ha desarrollado sobre AODV, como los dos módulos desarrollados en 2018 están contenidos dentro de la asignatura Redes Basadas en Dispositivos Móviles. A fin de unificar la apariencia de los tres WebLabs contenidos en la asignatura, para desarrollar el WebLab sobre AODV, se han reutilizado algunos de los elementos utilizados en los otros dos módulos, como la presentación de las tablas donde se nos muestra el resultado de los test, los colores del WebLab y las cabeceras que aparecen en la parte superior de los SCOs.

Figura 165: Ejemplo de los estilos y presentación empleados en los WebLabs. Fuente: Propia

5.7 Compatibilidades

Se ha comprobado el funcionamiento del módulo de aprendizaje desde un ordenador a través de los navegadores más utilizados[23]. Además, también se ha comprobado su funcionamiento desde un smartphone a través del navegador Google Chrome. Las conclusiones son las siguientes:

- Desde PC:

Tanto en Microsoft Edge, como en Mozilla Firefox y Google Chrome, el desempeño del WebLab es correcto, aunque presentan ciertas diferencias gráficas cuando se utiliza uno u otro navegador como, por ejemplo, ciertos componentes de la interfaz del simulador AODV.

Figura 166: Captura realizada desde el navegador Microsoft Edge. Fuente: Propia

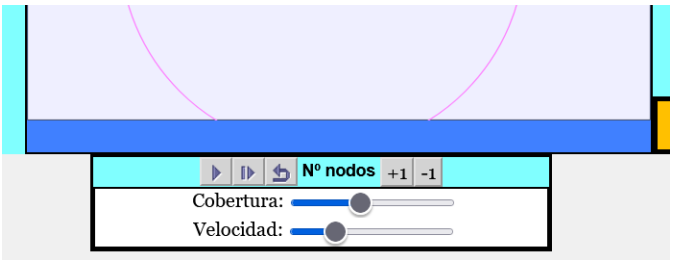


Figura 167: Captura realizada desde el navegador Mozilla Firefox. Fuente: Propia

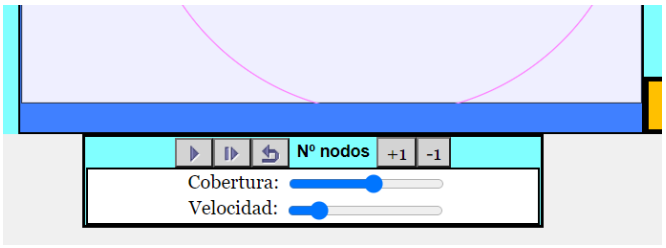


Figura 168: Captura realizada desde el navegador Google Chrome. Fuente: Propia

- En smartphone:

Tras comprobar el funcionamiento del WebLab en un smartphone se llega a la conclusión de que, la mejor manera de visualizar el módulo de aprendizaje es activando la opción Vista Ordenador en el smartphone y manteniendo este en orientación vertical.

Figura 169: Pasos para visualizar correctamente el módulo de aprendizaje desde un smartphone. Fuente: Propia

6. RESULTADOS Y DISCUSIÓN

El resultado final de este TFG ha consistido en el desarrollo de un simulador interactivo y personalizable capaz de recrear la formación de redes MANET entre nodos empleando AODV como protocolo de enrutamiento, siendo el principal objetivo de dicho simulador mostrar al usuario interesado cuales son las diferentes tareas (a nivel didáctico) que el protocolo AODV realiza durante toda la ejecución de la simulación.

Además, se ha creado un módulo SCORM centrado en la enseñanza del protocolo AODV, incluyendo en dicho SCORM, el propio simulador AODV desarrollado.

Gracias a la tecnología del LMS, cuando un usuario realiza el módulo SCORM, se almacena la información relacionada con la ejecución del módulo de aprendizaje, permitiendo al profesor o administrador del SCORM observar estos datos posteriormente.

Todas las páginas o SCOs que componen el SCORM tienen su importancia, pero entre las más importantes destacan TeoriaWebLab.html y TestFinal.html, ya que estas han sido establecidas como páginas imprescindibles de aprobar si se quiere superar el WebLab.

Figura 170: Definición de las páginas imprescindibles de aprobar para superar el WebLab. Fuente: Propia

Existen dos atributos que permiten conocer si se ha visitado una página y si dicha página se ha aprobado o no. Estos son los siguientes:

- Completion_status: Indica si una página ha sido ya visitada. Los valores que pueden aparecer son incompleted (no visitada) o completed (visitada).
- Succes_status: Indica si la página visitada está aprobada o suspensa. Existen páginas que solo son informativas, por lo que, para aprobar estas páginas, basta con pasar a las siguientes. En cambio, en las páginas donde existen preguntas tipo test, es necesario aprobar los test para aprobar las páginas. Los valores que puede

presentar este atributo es passed (aprobado) o failed (no aprobado).

Figura 171: Ejemplo de estados que pueden mostrar los atributos Completion_status y Succes_status. Fuente: Propia

7. CONCLUSIONES Y TRABAJO FUTURO

Con el desarrollo de este TFG se pretende que, por medio de los diversos recursos didácticos desarrollados que han sido explicados anteriormente, se permita simplificar la enseñanza de ciertos contenidos, sirviendo como apoyo a la labor del profesorado. Desde un inicio, se pensó únicamente en el desarrollo de una simulación básica capaz de explicar los aspectos principales del protocolo AODV, así como el desarrollo de un módulo SCORM sencillo. Finalmente, no se ha desarrollo un simulador capaz de recrear los aspectos más básicos del protocolo AODV, sino que es capaz de recrear cualquiera de los aspectos descritos en la RFC[2] que estandariza este protocolo, excluyendo algunos que se indicarán a continuación. Además, el simulador cuenta con varias opciones que permiten personalizar la simulación, incluyendo la elección del idioma (disponible en inglés y español).

Recapitulando, los objetivos que se han alcanzado con el desarrollo de este TFG han sido:

- Implementación de un simulador que permite a los usuarios externos comprender (gráficamente) el funcionamiento del protocolo AODV en redes MANET formadas por nodos.
- Implementación de un conjunto de actividades sobre el simulador ya desarrollado.
- Desarrollo de un video explicativo sobre cómo funciona el simulador AODV.
- Desarrollo de un módulo de aprendizaje SCORM que incluye todo lo nombrado anteriormente además de información teórica y tests sobre el tema explicado.
- Integración del módulo SCORM desarrollado en LMS, lo que permite tener un registro sobre las interacciones que los usuarios realizan dentro del WebLab.
- Generación de un informe final indicando el nombre del usuario y las calificaciones que ha obtenido en el módulo de aprendizaje.

En cuanto a los aspectos no implementados dentro del simulador AODV[24] que vienen descritos dentro de la RFC 3561 son:

- Técnica por búsqueda de anillo expandido: Se basa en una técnica utilizada por AODV para evitar la difusión innecesaria de paquetes RREQ por toda la red. Este aspecto no ha sido implementado simplemente por el hecho de que existen solo 10 nodos simulables. Esta técnica estaría bien que fuese implementada en redes MANET formadas por una gran cantidad de nodos.

- Reparación local: Proceso que permite corregir la rotura de un enlace de una ruta activa.
- Mensajes RREP-ACK: No son obligatorios y se trata de mensajes de reconocimiento de respuesta de ruta, los cuales deberían enviarse en respuesta a mensajes RREP con el Bit A activo.
- Formación de redes agregadas: Cuando un grupo de nodos móviles operan con relaciones fijas entre ellos y comparten un prefijo de subred, moviéndose juntos en un área donde se ha formado una red ad-hoc.

Por lo tanto, como trabajo futuro se podría:

- Implementar las funcionalidades no implementadas descritas anteriormente.
- Añadir más idiomas a la simulación.
- Incluir más nodos en la simulación.

8. ANEXOS

8.1 Informe de realización del WebLab MANET-AODV

Cuando un alumno finaliza el módulo de aprendizaje, aparece junto a este un enlace que permitirá la descarga de un informe donde se indican los resultados obtenidos por el alumno en dicho módulo. Este enlace solo aparecerá en caso de que se haya aprobado el WebLab.

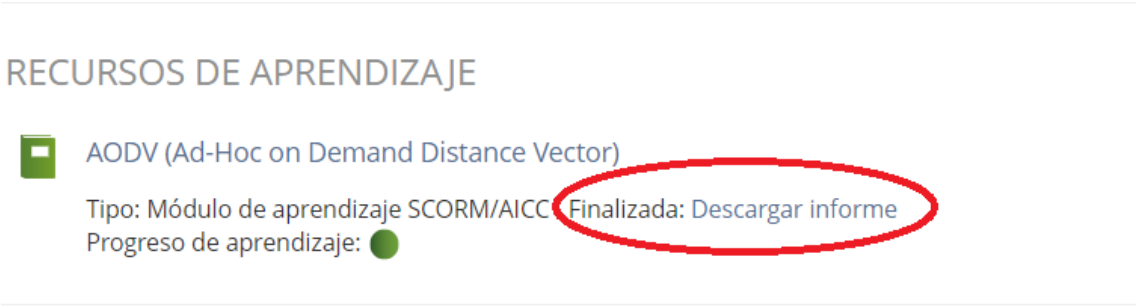


Figura 172: Momento en el que es posible generar un informe con los resultados obtenidos en el WebLab. Fuente: Propia

El informe presentará el siguiente aspecto:

Figura 173: Aspecto del informe generado al finalizar el WebLab. Fuente: Propia

8.2 Siglas

AODV	Ad-hoc On-Demand Distance Vector
CSS	Cascading Style Sheets
DOM	Document Object Model
EJS	Easy Java Simulations
HTML	HyperText Markup Language
IETF	Internet Engineering Task Force
LMS	Learning Management System
MANET	Mobile Ad-hoc Network
OLSR	Optimized Link State Routing Protocol
QTI	Question & Test Interoperability
SCO	Shareable Content Objects
SCORM	Shareable Content Object Reference Model
VANET	Vehicular Ad-hoc Network
WSN	Wireless Sensor Networks
W3C	World Wide Web Consortium
XML	Extensible Markup Language
RFC	Request for Comments
CERN	European Centre of Nuclear Physics
SGML	Standard Generalized Markup Language
ADL	Advanced Distributed Learning
GUI	Graphical user interface
XHTML	Extensible HyperText Markup Language
UDP	User Datagram Protocol

8.3 Índice de figuras

9. REFERENCIAS BIBLIOGRÁFICAS

1. Ruano Ruano A. Protocolos de encaminamiento MANET. Published online 2021. https://dv.ujaen.es/goto_docencia_file_533994_download.html
2. Perkins C, Belding-Royer E, Das S. RFC Ad hoc On-Demand Distance Vector (AODV) Routing. *Req Comments 3561*. https://datatracker.ietf.org/doc/html/rfc3561
3. Perkins C, Ratliff S, Dowdell J, Steenbrink L, Mercieca V. RFC Ad Hoc On-demand Distance Vector Routing Version 2 (AODVv2). https://datatracker.ietf.org/doc/html/draft-perkins-manet-aodvv2-03
4. Ruano Ruano A. Grupo de trabajo IETF en MANET. In: ; 2021. https://dv.ujaen.es/goto_docencia_file_533994_download.html
5. Tipos de RFCs. Published online 2021. https://www.nfon.com/es/servicio/base-de-conocimiento/base-de-conocimiento-destacar/request-for-comments-rfc
6. Ramos R. ¿Qué es JavaScript? https://soyrafaramos.com/que-es-javascript-para-que-sirve/
7. B G. ¿Qué es HTML? https://www.hostinger.es/tutoriales/que-es-html
8. Estructura básica de HTML. https://studentplace98.blogspot.com/2019/04/EstructuraBasica.html
9. Manz. CSS. https://lenguajecss.com/css/introduccion/que-es-css/
10. DOM. https://www.w3schools.com/js/js_htmldom.asp
11. XML. https://www.mundolinux.info/que-es-xml.htm
12. QTI. http://pdi.topografia.upm.es/m.manso/docs/Estandar_QTI_Descripcion.pdf
13. E-Learning. Published 2021. https://cognosonline.com/co/blog/que-es-e-learning/
14. Peñalba A. SCORM. Published online 2021. https://www.homuork.com/es/scorm-que-es-y-por-que-es-clave-en-el-e-learning_226_102.html
15. Laguna MP. Arquitectura del modelo SCORM. https://www.unpa.edu.ar/sites/default/files/descargas/Administracion_y_Apoyo/4.Materiales/2015/T093/SCORM_Standar.pdf
16. Laguna Lozano MP. Versiones SCORM. https://www.unpa.edu.ar/sites/default/files/descargas/Administracion_y_Apoyo/4.Materiales/2015/T093/SCORM_Standar.pdf
17. Herramientas de simulación de red. http://bibing.us.es/proyectos/abreproy/70218/fichero/4.Herramientas+de+Simulación+para+redes+ad+hoc.pdf
18. ns. https://es.wikipedia.org/wiki/Ns_(simulador)
19. Ejercicio mensaje HELLO.

https://riunet.upv.es/bitstream/handle/10251/10081/tesisUPV3452.pdf;jsessionid=748634AFC954E19911B75778958821E6?sequence=1

20. Estrada D. SweetAlert2. Published 2018. https://denisseestrada.com/como-hacer-una-ventana-emergente-con-sweetalert2/
21. Sanchez Salido R. WebLab OLSR. https://dv.ujaen.es/goto_docencia_sahs_934865.html
22. Sanchez Salido R. WebLab VANET. https://dv.ujaen.es/goto_docencia_sahs_934874.html
23. Marquina J. Los navegadores más utilizados. Published 2021. https://www.julianmarquina.es/los-10-navegadores-web-mas-utilizados-en-los-ordenadores-del-mundo/
24. Ruano Ruano A. *Transparencias Protocolo AODV*. https://dv.ujaen.es/goto_docencia_file_533992_download.html