TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

MURAT

Multiagent Urban Traffic Control

Autor
Raúl López Arévalo

Directores
Luis Castillo Vidal

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Septiembre de 2020

MURAT: Multiagent Urban Traffic Control

Raúl López Arévalo

Palabras clave: Sistema Multiagente, SMA, Agentes, Tráfico, Control de tráfico, Vehículos, Semáforos, Congestión vehicular, Cruce

Resumen

Se pretende diseñar un sistema multiagente denominado MURAT (Multiagent Urban Traffic Control) que gestione el tráfico vehicular en zonas de congestión de un área urbana. En este sistema intervendrán distintos tipos de agentes, trabajando en conjunto para lograr una mayor fluidez de tráfico, siendo fundamental una correcta comunicación entre ellos.

Se realizará una simulación de varios cruces en los que es sabido que hay atascos en determinados momentos del día. Los agentes, se encargarán de gestionar el paso de vehículos modificando los semáforos, consiguiendo así reducir el tiempo de espera y agilizando el tráfico.

MURAT: Multiagent Urban Traffic Control

Raúl López Arévalo

Keywords: Multi-agent System, MAS, Agents, Traffic, Traffic Management, Vehicles, Lights, Traffic congestion, Crossroad

Abstract

It is intended to design a multi-agent system called MURAT (Multiagent Urban Traffic Control) which manage the traffic in an urban area. This system will have different kind of agents, working all togheter to claim a better traffic flow, being important a good communication between them.

A simulation will be run in different crossroads, where high traffic jams are known. The agents will manage the traffic flow changing the lights, saving time to the drivers and making traffic more fluid.

Yo, **Raúl López Arévalo**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 25353710M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Raúl López Arévalo

Granada a 2 de Septiembre de 2020.

D. **Luis Castillo Vidal**, Profesor del Área de Sistemas Inteligentes del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***MURAT, Multiagent Urban Traffic Control***, ha sido realizado bajo su supervisión por **Raúl López Arévalo**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 2 de Septiembre de 2020.

El director:

Luis Castillo Vidal

Agradecimientos

A mi familia, por siempre estar ahí y escucharme cuando lo he necesitado. Sin ellos no hubiese sido posible llegar hasta aquí.

A mi pareja, por apoyarme durante todo el desarrollo del proyecto, en los días de más y menos estrés, gracias.

A Luis, mi tutor, por haberme descubierto en su momento el mundo de los sistemas multiagentes, por su forma de enseñar y por la oportunidad de haber podido trabajar en este proyecto.

Índice general

Índice de figuras

Capítulo 1

Introducción

Uno de los mayores problemas existentes en una mediana y gran ciudad, es la congestión del tráfico a determinadas horas del día. Esta situación provoca largas colas y esperas entre los conductores, más polución, retrasos al llegar a los destinos deseados, frustración al volante y sobre todo, nuestro tiempo. La búsqueda de una solución ha sido desde hace tiempo un tema de interés, utilizándose comúnmente hasta ahora sistemas centralizados con una alta participación del factor humano.

Debido a la gran versatilidad de los agentes y sus propiedades, una solución válida para este problema es la utilización de un sistema multiagente, tal y como se expone en[1]. Una de las ventajas principales de usar este sistema sería la anulación por completo del factor humano, quedando relegada su función a la de supervisión.

Realizando un estudio de las políticas de funcionamiento de los semáforos actuales y como influye en la congestión vehicular, podemos asignar diferentes roles y tareas a cada uno de nuestros agentes[2]. Cada agente será encargado de realizar una tarea en concreto, pudiendo intercambiar información entre ellos y juntos tener la capacidad para gestionar el tráfico bajo cualquier situación extrema, en la que, los semáforos convencionales pierden toda utilidad.

Para este proyecto se usará una estructura de agentes jerárquica, donde el agente más simple es el semáforo. El semáforo recibirá instrucciones del agente cruce para cambiar su luz y los cruces a su vez deberán de informar con cierta frecuencia a la central encargada de analizar todos y cada una de las situaciones y generar los informes.

1.1. Objetivos

Este proyecto tiene como objetivo principal construir un sistema multiagente que permita el control del tráfico. Esto debe hacerse evitando cualquier colisión posible entre vehículos de distintas vías, conocer cómo actuar si existe una saturación en algún tramo y, al final, conseguir que el número de vehículos que pasa por un determinado punto, se vea incrementado.

Para conseguir lo deseado, empezaremos por un análisis del problema a solventar. De esta forma podremos elegir con más criterio un sistema que se adecue mejor a las necesidades. A continuación decidiremos el número de agentes a crear, así como sus propiedades y funcionalidades.

Una vez implementado el sistema multiagente, se procederá a la realización de pruebas periódicas para cerciorarnos del correcto funcionamiento del programa y la simulación. Al final, se crearán situaciones extremas en las que analizaremos los resultados obtenidos mediante una comparativa de la simulación con lógica, en la que los agentes deciden cambiar el estado de los semáforos a su criterio, y sin lógica. El proyecto completo puede encontrarse aquí .

1.2. Estructura

Esta memoria está dividida en tres partes bien diferenciadas. En la primera parte, se analizará la situación actual y se entrará en detalle en los componentes de un sistema multiagente, propiedades relevantes y comunicación.

En la segunda parte, se verán en detalle el funcionamiento de todos y cada uno de los agentes que componen MURAT, así como todos los diagramas de diseño.

Finalmente, en la última parte, explicaremos como se realiza la simulación, y se hará una comparativa de resultados de las distintas gráficas obtenidas de una simulación completa de un día, con y sin la lógica.

Parte I

Un Sistema Multiagente

Capítulo 2

Presente

La gestión del tráfico hoy en día está muy cerca de ser un sistema distribuido con autonomía. En cada intersección en la que hay ausencia de cualquier tipo de glorieta, existen semáforos. Su función es la de permitir el paso o no a los vehículos, impidiendo así accidentes y permitiendo un control por estados de los vehículos sin la necesidad de presencia humana.

También es común encontrar al principio de vías principales dispositivos con la finalidad de recabar información sobre el flujo de coches existente, denominados sensores. Esta información es utilizada para estadísticas y análisis futuros.

2.1. Sistemas en uso

El sistema TRYS[3], es un sistema multiagente como ayuda a la toma de decisiones para la gestión del tráfico en una zona urbana en tiempo real. Este sistema se encuentra en funcionamiento en Madrid.

A cada zona problemática, TRYS asigna un agente que monitoriza el tráfico en su área designada, detecta los posibles accidentes que puedan ocurrir y permiten solucionar dichos problemas. Por encima de estos agentes, se encuentra un agente coordinador que supervisa y determina la capacidad de cada uno de los anteriores.

También existe otro sistema, llamado TRYSA, basado en el que acabamos de explicar. La diferencia es que este último no dispone de un agente coordinador y los agentes encargados de solucionar los distintos tipos de problemas se vuelven egoístas y se convierte en un sistema de negocio.

Capítulo 3

Sistema MURAT

Dado que el problema actual se produce cuando existen situaciones de estrés vehicular, construiremos nuestro sistema con el objetivo de ser lo más escalable posible y que sepa responder a la demanda de altos picos de tráfico. De esta forma el funcionamiento limitado de los semáforos actuales sería suprimido.

Tomando como referencia los sistemas actuales en uso, y la división de los distintos tipos de estructuras que hace Dignum[4] decidimos que nuestro sistema será una combinación de una estructura jerárquica y de redes. Existe una jerarquía entre ciertos tipos de agentes, que obedecen a otros pero también existe un interés común y confían los unos en los otros. De esta forma, los agentes nunca negociarán entre ellos.

El proyecto se ha desarrollado en JAVA, utilizando la plataforma de agentes Magentix2[5](versión 2.0.1), desarrollada por la Universidad Politécnica de Valencia, de la cual hablaremos más en profundidad en su correspondiente sección.

Con lo hablado hasta ahora, podemos deducir fácilmente qué función tendrán ciertos tipos de agentes. Así, existirá un agente Sempahore encargado únicamente de cambiar su luz, un agente Crossroad encargado de decidir cuando cambiar de estado los semáforos, un agente Switchboard que procesará y generará informes, un agente Simulator encargado del bruto de la simulación y los agentes Sensor encargados de transmitir los datos de los vehículos que detectan a los agentes de los correspondientes cruces. Se verán en mayor profundidad en el capitulo 4.

3.1. Agentes

Comenzaremos esta sección hablando de lo que se entiende por agente. Su definición ha ido variando a lo largo del tiempo y aún así, no hay un consenso sobre cuál es más representativa. Así, nosotros utilizaremos la propuesta por Wooldridge y Jennings[6], según la cual *un agente es un sistema informático que está situado en un entorno y es capaz de actuar de forma autónoma y flexible.*

Antes de entrar en profundidad en cada uno de los agentes que componen el sistema, hablaremos sobre las propiedades más importantes que presentan. Los agentes que utilizaremos son agentes reactivos, son capaces de responder en un tiempo adecuado a cambios en el entorno en el que se encuentran situados. Son proactivos, capaces de exhibir un comportamiento óptimo, enfocado a obtener sus metas. Son sociales, capaces de interactuar con otros agentes a través de un lenguaje de comunicación entre agentes.

Existen otras muchas propiedades atribuibles a los agentes. A continuación solamente nombraremos aquellas que son aplicables en este caso:

- Continuidad temporal: Se considera un agente como un proceso sin fin, ejecutándose continuamente y desarrollando su fucnión.
- Veracidad: Asunción de que un agente no comunica información falsa a propósito.
- Benevolencia: Asunción de que un agente está dispuesto a ayudar a otros agentes.

3.1.1. Estructura de un agente

En la plataforma utilizada para la comunicación nombrada anteriormente, Magentix, podemos dividir la vida de un agente (desde que se crea, hasta que se destruye) en tres métodos principales. Estos métodos son ejecutados secuencialmente y son los siguientes:

1. initialize: Este método es el primero en ser llamado. En él se deben inicializar los atributos necesarios del agente. Existen varias opiniones al respecto, pero si se quisiese, se podría evitar usar este método y hacer dichas inicializaciones dentro del porpio constructor del agente.
2. execute: Este método es el cuerpo del agente, es el que se ejecuta mientras el agente está activo. La finalización de este método, conlleva

a la finalización del agente. Debido a esto, ejecutaremos infinitamente este método mediante un buble. Dentro de este bucle definiremos los distintos estados específicos de cada agente, de forma que pasará de uno a otro estado según corresponda.

3. finalize: Este método conlleva a la finalización del agente. Es por ello que, dentro de él, introduciremos las últimas acciones que están programadas para ocurrir cuando la vida útil del agente llega a su fin.

3.2. Comunicación

En Magentix2 cada agente tiene la capacidad de poder enviar y recibir mensajes de cualquier otro. A continuación vamos a explicar cada una de estas acciones, y los distintos tipos que existen.

3.2.1. Envío

Para enviar, disponemos del método *send(ACLMessage message)*. Como podemos observar, el envío de mensaje tiene como argumento un objeto de tipo *ACLMessage*, el mensaje. Este mensaje ya contiene todas las propiedades necesarias para su correcto envío y recepción.

3.2.2. Tipo de recepción

Si hablamos de recepción de mensaje, encontramos dos tipos diferentes, *onMessage* y *receiveACLMessage*. El primero se ejecuta en una hebra distinta a la del agente cada vez que se recibe un mensaje, almacenándose en una cola interna. Por otro lado, el método *receiveACLMessage*, extrae el primer mensaje encontrado en la cola anterior. De no ser así, se mantiene a la espera. La clara diferencia entre un ambos tipos de recepciones entonces es, que la segunda es bloqueante.

Debido al comportamiento que tienen los agentes en MURAT y sus objetivos definidos, se ha optado por construir una comunicación bloqueante. Aunque el tamaño de los mensajes y el cómputo realizado pueda parecer de tamaño considerable, es mínimo el tiempo que invierten.

3.2.3. Estructura de un mensaje

La clase *ACLMessage* (Agent Communication Language) sigue el estándar de comunicación desarrollado por *FIPA (Foundation for Intelligent Physical*

Agents). Esta clase posee todo lo necesario para dar formato, utilidad y sentido a un mensaje. De entre todos los atributos que posee, vamos a nombrar a continuación cuales de ellos hemos utilizado y su función:

- Receiver: Es el identificador del agente (AgentID) que recibe el mensaje.
- Sender: Es el identificador del agente (AgentID) que envía el mensaje.
- Performative: Especifica el propósito del mensaje, si es una petición, una orden, un informe, etc.
- Content: Es el contenido del propio mensaje, en formato cadena de texto.
- Encoding: Es la forma mediante la cual ha sido codificado el mensaje.

3.2.4. Contenido y codificación

Es importante utilizar un estándar en cuando a la codificación de mensajes en un sistema multiagente. De esta forma, cada agente sabe cómo tiene que leer el mensaje y qué obtener de él. En MURAT, todos los mensajes están codificados en *JSON* (recordemos el atributo *Encoding* de un mensaje) pasado a cadena de texto, que es lo que admite como parámetro el atributo *Content*.

Además de la codificación utilizada, el mensaje se encapsula en el *value* de un objeto *JSON* con una *key* identificativa. De esta forma, el receptor mediante la performativa del mensaje y esta etiqueta sabe a la perfección lo que está recibiendo, o lo que se le está pidiendo.

3.2.5. Performativas

La performativa de un mensaje es uno de sus atributos más importantes. Es lo que el agente receptor interpretará dependiendo de la performativa que lea. A continuación, nombramos algunas de las utilizadas en este proyecto:

- REQUEST: El emisor le pide al receptor que ejecute una acción. Se sabe que el receptor puede hacerlo y que la acción no está hecha aún.
- SUBSCRIBE: El emisor le pide al receptor el valor de un objeto. Cada vez que dicho objeto cambie, se debe volver a informar al emisor.
- INFORM: Se utiliza tanto como respuesta al *REQUEST* como al *SUBSCRIBE* cuando se ha ejecutado la acción con éxito.

- AGREE: El receptor acepta ejecutar la acción pero en un futuro.

Parte II

Modelado UML y Base de Datos

Capítulo 4

Diagramas de estados

Hemos hablado de los tipos de agentes existentes en MURAT, sus propiedades más importantes y cómo se comunican entre si. En este capítulo veremos más detenidamente cada uno de los agentes, sus estados, cómo forman la sociedad y el flujo de comunicación existente. Los aspectos de la simulación y la lógica los veremos en el capítulo 8.

4.1. Agente Switchboard

Empezaremos hablando del agente Switchboard o centralita. Este es el primer agente en crearse y se crea desde el Main. Lo primero que hace es leer de base de datos las distintas configuraciones de los tramos y los estados de cada semáforo. Hablaremos de la base de datos en el capítulo 7.

Su función principal es la de recolectar los informes que le envían los distintos cruces para, al final de la simulación, generar un archivo con las estadísticas obtenidas. A partir de estos datos realizaremos el estudio de las ventajas obtenidas en cuanto al tráfico. Explicamos los estados del agente Switchboard como se muestra en la figura 4.1:

- Get Data: En este estado, el agente lee de la base de datos todas las configuraciones necesarias para la creación del resto de agentes. Aquí, también construye las estructuras de datos que necesita cada agente para inicializarlos.
- CreateAgent: Mediante los datos leídos de la base de datos y las estructuras creadas, crea e inicializa a todos y cada uno del resto de agentes.
- Subscribe Crossroads Informs: El agente manda un mensaje con la

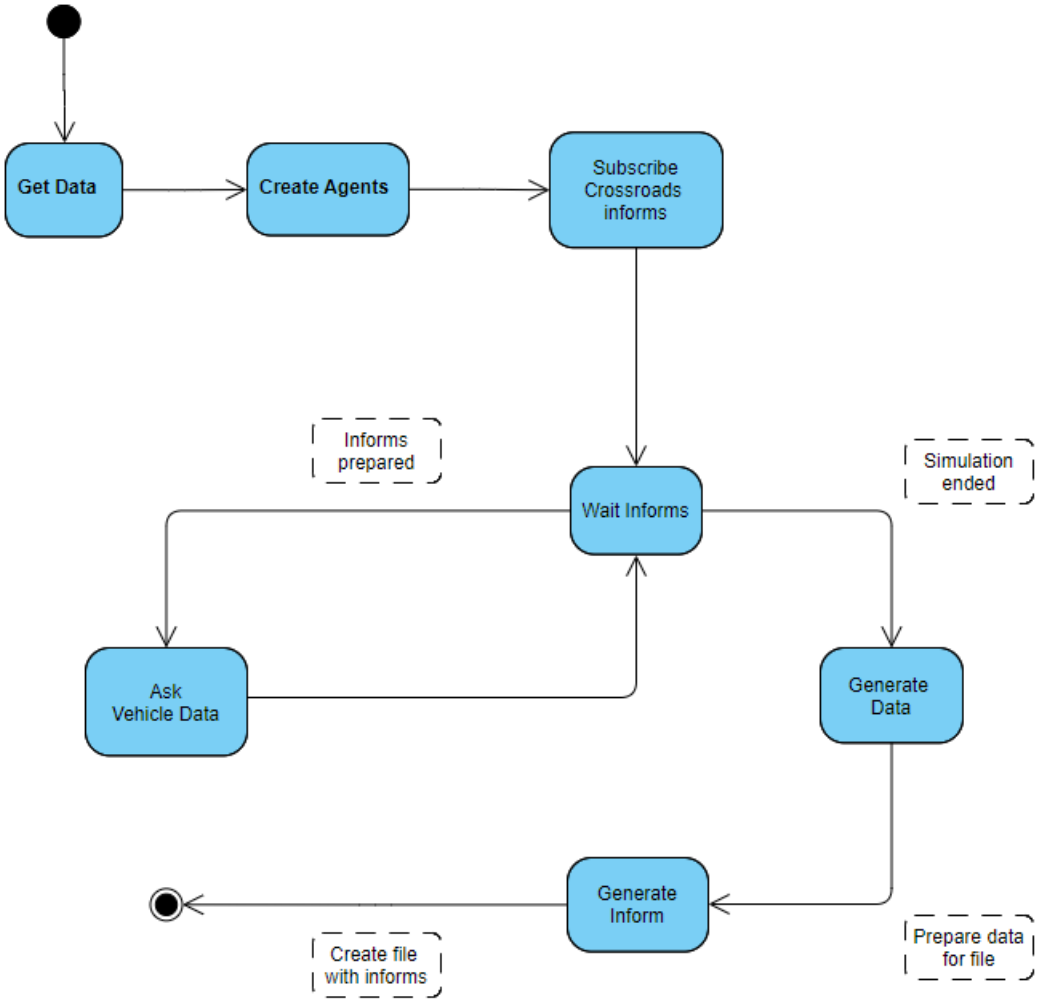

Figura 4.1: Diagrama de estados del agente Switchboard

performativa *SUBSCRIBE* para recibir los informes de los agentes cada vez que estos los tengan disponibles.

- Wait Informs: En este estado, el agente permanece en escucha hasta que recibe un mensaje y lo analiza. Si es algún cruce enviando el informe de tráfico al cual la centralita se había subscrito, procesa y almacena el informe. En caso contrario, recibirá un mensaje del agente Simulator pidiéndole que finalice, ya que la simulación habrá terminado.

- Ask Vehicle Data: Una vez que los informes han sido procesados, la centralita pregunta al simulador por una serie de estadísticas que solamente él conoce. Por ejemplo, el número de vehículos totales en la simulación o el número de vehículos que se han generado y han salido del mundo simulado. Dado que utilizamos una comunicación bloqueante, esta request le indica al simulador implícitamente, que puede continuar con la siguiente iteración de su simulación.

- Generate Data: Si en el estado *Wait Informs*, recibe la orden de finalizar por parte del simulador, llegamos a este estado. Aquí la centralita recopila todos los informes guardados y los prepara para ser escritos en un fichero.

- Generate Inform: Llegados a este punto, el agente crea un fichero por cada cruce con extensión *csv*, con todas las estadísticas generales y las específicas de cada uno de ellos. Siendo así posible un estudio posterior de los datos obtenidos.

- finalize: Cuando el agente va a finalizar porque así se lo ha indicado el simulador, envía la petición de terminar a todos los agentes cruces.

4.2. Agente Simulator

Pese a que el agente Switchboard es el primero en ser arrancado, cabe destacar la gran importancia del agente Simulator o simulador. Aquí recae todo el peso de la simulación y sin este agente, no sería posible llevar a cabo MURAT. En el capítulo 8 veremos en en más detalle cómo se realiza la simulación. Explicamos los estados del agente Simulator como se muestra en la figura

- Wait Subscribe Sensors: El agente permanece en escucha hasta que todos los agentes sensores se han suscrito para recibir, cada segundo, la situación del tráfico.

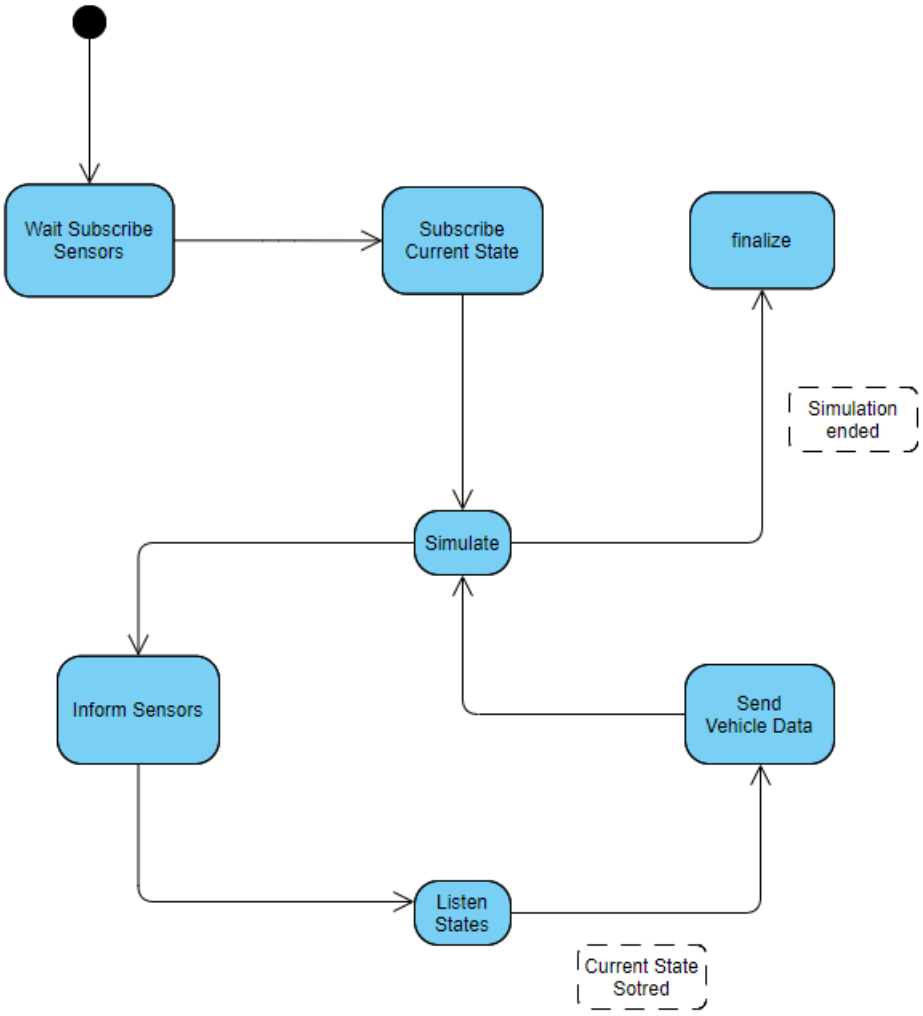

Figura 4.2: Diagrama de estados del agente Simulator

- Subscribe Current State: El agente se suscribe a cada cruce para pedirle que cada vez que cambie de estado de semáforos se lo indique. De esta forma, el agente Simulator siempre conoce el estado actual de cada semáforo y puede tenerlo en cuenta para la simulación.
- Simulate: Este es el estado principal del agente, en el que, cada vez que vuelve a él, significa un segundo simulado. Lo primero que hace en este estado es generar tráfico en todas las calles que sean nodos raíz, siempre y cuando sea posible. La segunda acción que se produce en este estado es la de canalizar el tráfico. Mueve cada vehículo que está dentro de la simulación siguiendo unas reglas descritas en el capítulo 8.
- Inform Sensors: Aquí informa a todos los sensores, cada segundo, del número de vehículos que han pasado por su sección así como el espacio que ocupan.
- Listen States: El agente permanece a la espera, para recibir en caso de que haya habido cambios, los estados de configuración de cada uno de los cruces.
- Send Vehicle Data: Información relevante para los informes de la centralita es envíada a éste agente, como el número de vehículos totales existentes en la simulación, o todos los vehículos que se han generado o quitado de ella.
- finalize: Cuando el simulador ha decidido que ya ha finalizado, pide al agente Switchboard finalizar también. Una vez recibida la confirmación, el agente Simulator finaliza.

4.3. Agent Sensor

Por cada tramo de carretera existente en la simulación, existen dos agentes sensores. Uno al comienzo de dicho tramo, y otro al final. El agente sensor recibe cada segundo la información de tráfico del simulador y transmite esta información al cruce de destino del tramo al que pertenece. Este agente es considerablemente más sencillo que los dos anteriores. Explicamos los estados mostrados en la figura 4.3.

- Wait Subscribe Crossroads: En este estado el agente permanece a la espera de recibir la suscripción por parte de su cruce correspondiente, al que le enviará la información de tráfico.
- Wait Time: El agente Sensor espera a que el agente Simulator le comunique lo que ha percibido durante este segundo. Una vez recibidos los datos, los almacena.

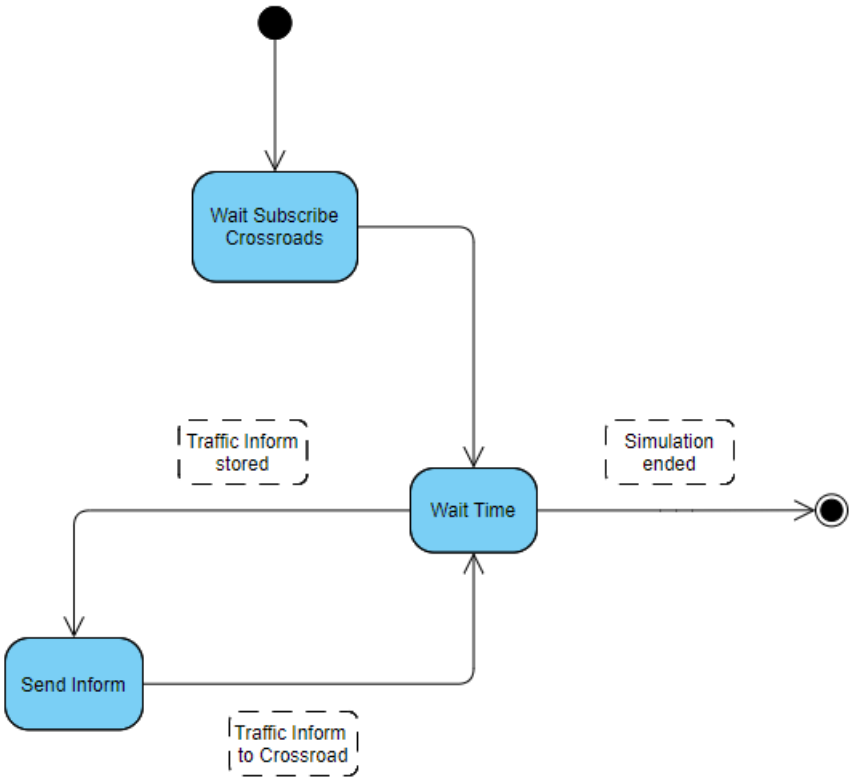

Figura 4.3: Diagrama de estados del agente Sensor

- Send Inform: En este estado, los datos almacenados en el anterior, son enviados al cruce correspondiente.

4.4. Agente Crossroad

Por cada cruce existente en la simulación, tenemos un agente cruce. Este agente se encarga de analizar los datos obtenidos de los sensores de sus tramos de carretera de entrada. Mediante este análisis, el agente Crossroad, decide a qué configuración de semáforos le permite estar más o menos tiempo en funcionamiento. Explicamos los estados mostrados en la figura 4.4.

- Wait Subscribe Switchboard: Aquí el agente permanece a la espera de la suscripción de la centralita para los informes.

- Subscribe Sensors: El agente Crossroad se suscribe a los sensores correspondientes para que estos le envíen los informes o datos de tráfico cada segundo.
- Wait Subscribe Simulator: El agente espera a recibir la suscripción del agente Simulator para que se le envíe el estado actual de la configuración de semáforos del cruce.
- Listen Sensors: El agente queda en escucha para recibir los informes de tráfico de sus sensores gracias a la prevía suscripción de este. Una vez recibidos los informes de cada sensor, construye los informes de cada tramo de carretera de entrada y los guarda. Quedando listos para ser enviados al agente Switchboard y para el análisis del propio cruce.
- Think: Este es el grueso del agente Crossroad. Aquí se encuentra la lógica que decide a qué estados de semáforos les permite estar activos más tiempo o menos.
- Change Semaphores: Los semáforos pertenecientes a este cruce, reciben la petición de cambiar su correspondiente color si así es necesario y se ha decidido en la lógica.
- Send Current State: El agente envía su estado actual de configuración de semáforos al agente Simulator.
- finalize: Una vez que el agente a recibido la orden de finalizar, finaliza también a todos sus semáforos.

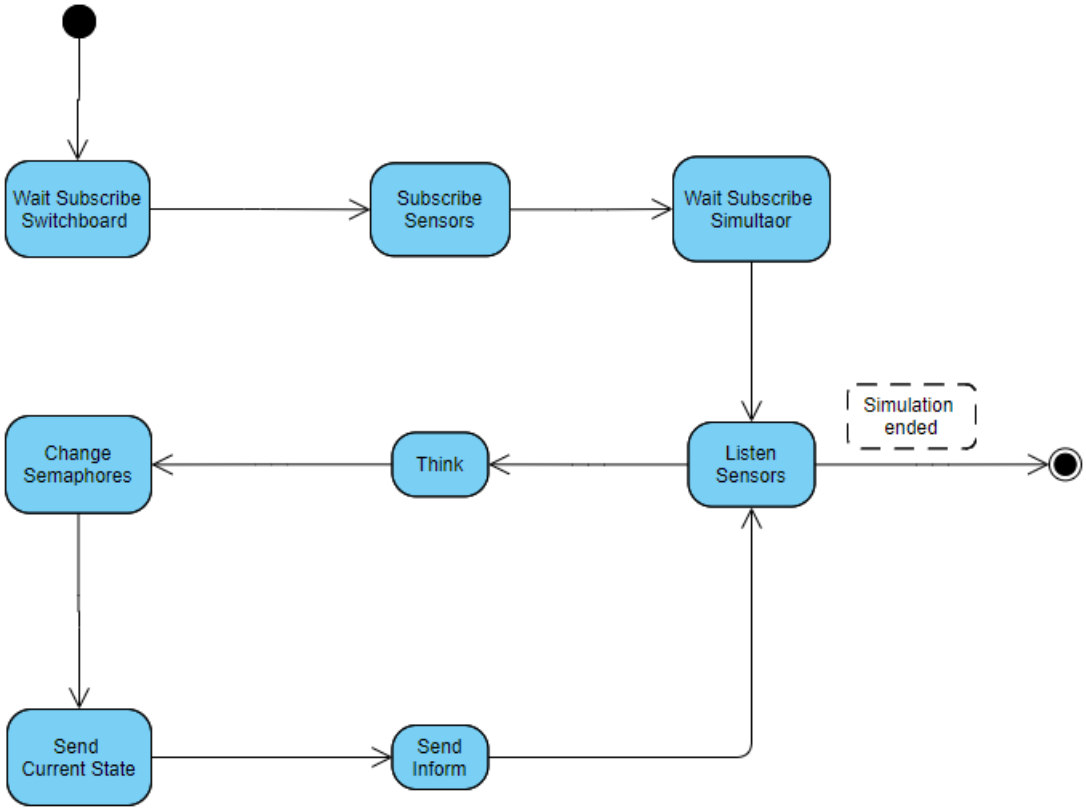

Figura 4.4: Diagrama de estados del agente Crossroad

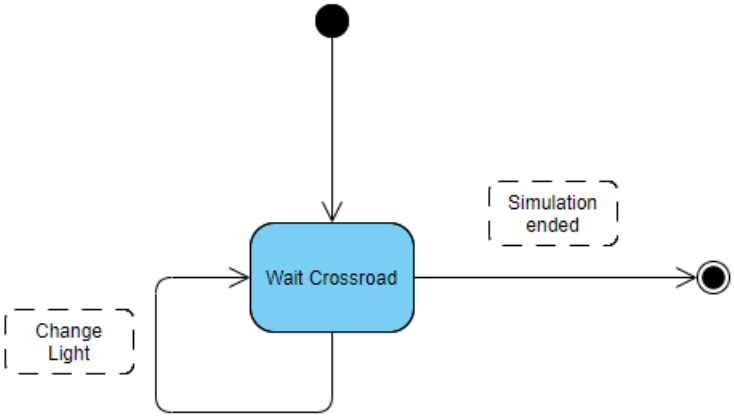

Figura 4.5: Diagrama de estados del agente Semaphore

4.5. Agente Semaphore

Este es el agente más sencillo que existe y está bajo las órdenes de su correspondiente cruce. Su única función es la de cambiar su luz de color, estando por tanto, en una continua escucha esperando a su cruce.

Como podemos observar en la figura 4.5, solamente posee un estado. En este estado, espera el mensaje de su cruce para el cambio de luz, o para finalizar.

Capítulo 5

Diagramas de secuencia

En el capítulo anterior ya se ha visto en profundidad los estados de cada agente. Los agentes pasan por algunos de estos estados una única vez, como puede ser para las suscripciones a otros agentes, o el estado finalizar. Teniendo esto en cuenta vamos a dividir este capítulo en tres partes. Primero, hablaremos de la secuencia de envío de mensajes relacionada con las suscripciones (Figura 5.1). Después, sobre la secuencia de envío de mensajes que se repite en cada iteración de la simulación (Figura 5.2). Por último, mostraremos la secuencia relacionada con la finalización de todos los agentes (Figura 5.3).

5.1. Suscripciones

Lo primero que ocurre en esta parte de la ejecución de MURAT, es la lectura de base de datos por parte del agente Switchboard y la creación del resto de ellos. Todos los agentes a excepción de la centralita permanecen a la espera para recibir y contestar a cada una de sus suscripciones.

La centralita pide a todos los cruces que le envíen los informes. Esto se realiza cada cierto tiempo. Los cruces piden a los correspondientes sensores de sus tramos de carretera de entrada que le envíen los informes de tráfico cada segundo. Los sensores piden al simulador que le envíe los datos del tráfico vehicular cada segundo. Por último el simulador le pide a todos los cruces que lo avisen y envíen la configuración actual elegida de sus estados de semáforos.

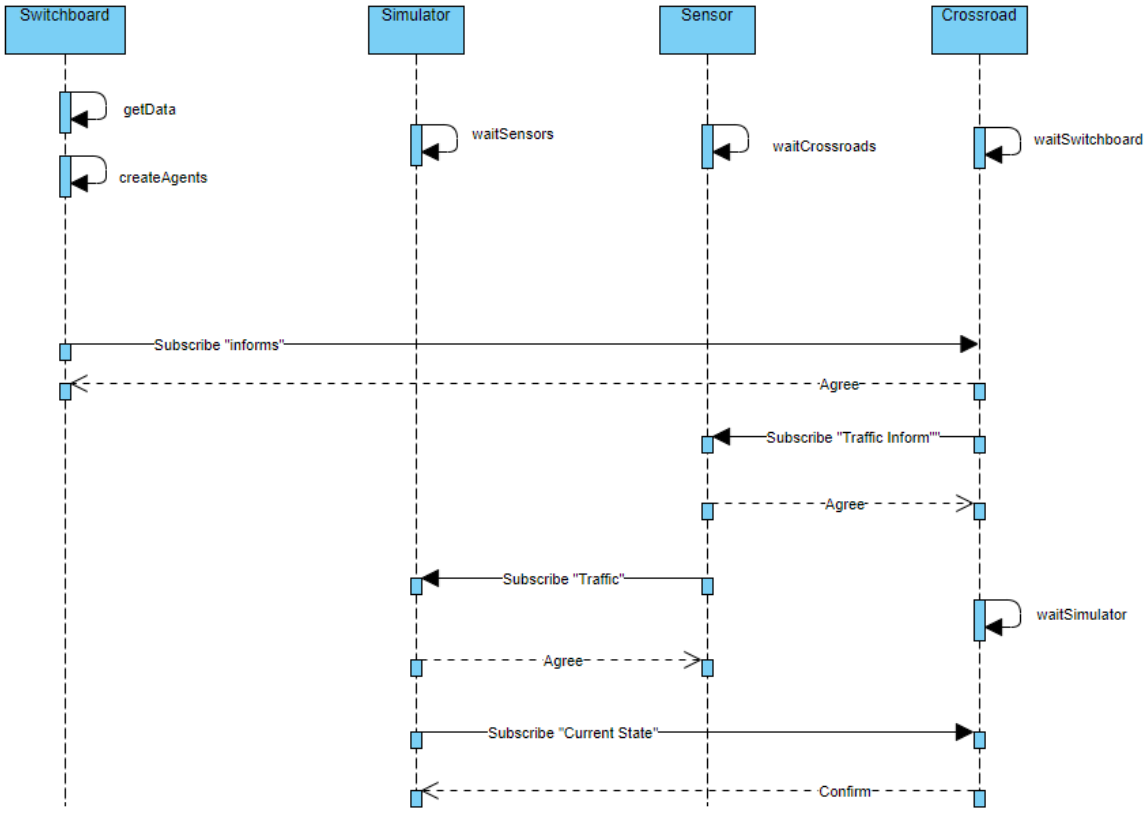

Figura 5.1: Diagrama de secuencia de las suscripciones

5.2. Cada Iteración

Terminadas todas las suscripciones de los agentes, comienza la secuencia de mensajes que se repetirá en cada segundo simulado.

Primero, el agente Simulator realiza la simulación de tráfico de un segundo e informa a todos los sensores con los datos pertinentes. A continuación, estos sensores transmiten la información a sus correspondientes cruces. Los cruces analizan la información de tráfico recibida de los sensores, modifican la duración de cada estado de semáforos según su lógica y piden a los semáforos cambiar su luz si es necesario. Una vez hecho esto, los cruces envían al simulador su configuración actual de semáforos y el informe generado de este segundo a la centralita.

Por último, la central prepara y almacena los informes recibidos de cada uno de los cruces y le pide al simulador que le envíe datos relevantes de la simulación en ese momento. Al recibir esta petición de datos, el simulador los envía y comienza de nuevo a simular el siguiente segundo.

5.3. Finalización

Este paso de mensajes es el más corto en cuanto a duración de tiempo se refiere, pero no por ello menos importante. Ya que es necesario una correcta finalización de todos los agentes para que no se produzca ningún error. Debido a la clase padre del agente Switchboard, éste debe de ser el agente que finalize el último.

Una vez que el agente Simulator decide terminar la simulación, envía la petición de finalizar a la centralita. Ésta le responde con un *AGREE*, confirmando así su finalización pero no ahora. Antes, la centralita genera los informes y los escribe en un fichero con extensión *csv* por cada cruce existente. Hecho esto, igual que hizo el simulador, envía la petición de terminar a todos los cruces y sensores, y permanece durmiendo dos segundos. Los cruces piden finalizar a sus correspondientes semáforos y estos finalizan sin más.

Llegados a este punto, todos los agentes han finalizado excepto la centralita que esta en estado *sleep*. Pasados los dos segundos configurados como tiempo de espera, el agente Switchboard finaliza. Así, se produce una finalización en cadena, en la que el primer agente en finalizar es el Simulator y el último el Switchboard.

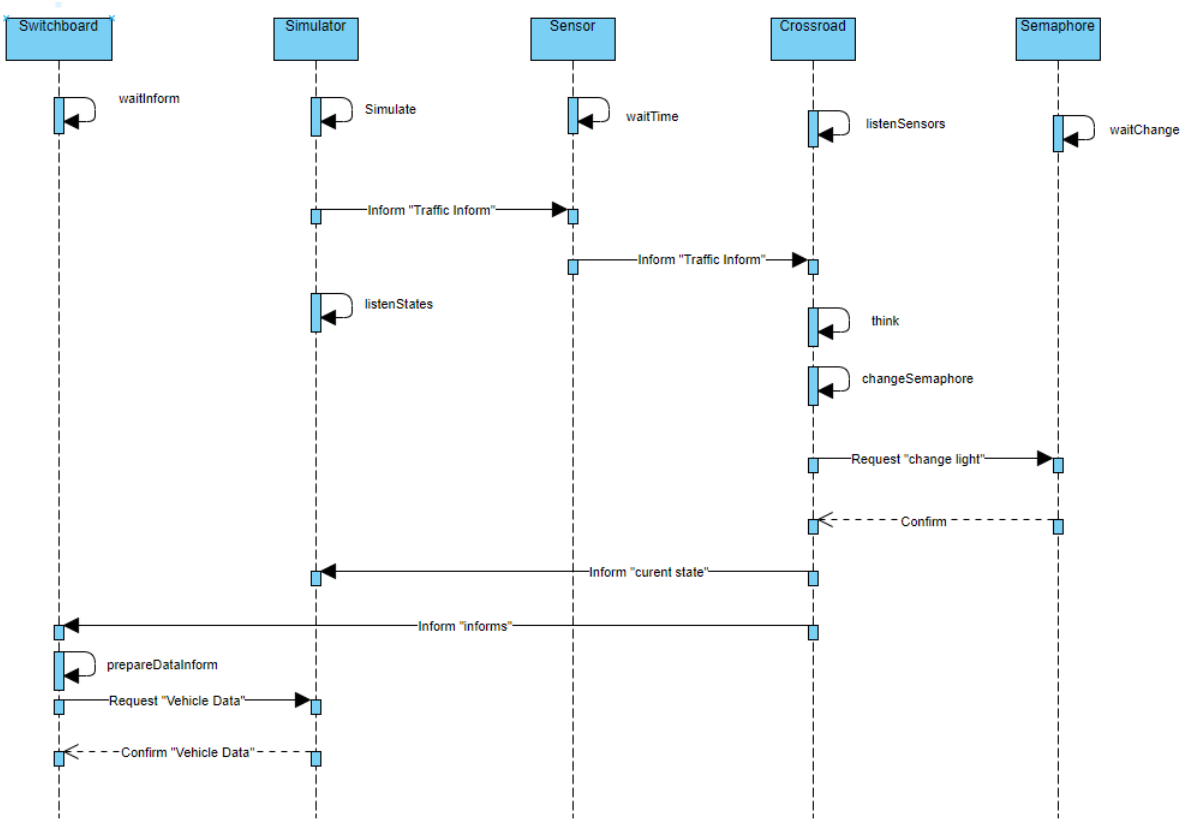

Figura 5.2: Diagrama de secuencia de cada segundo simulado

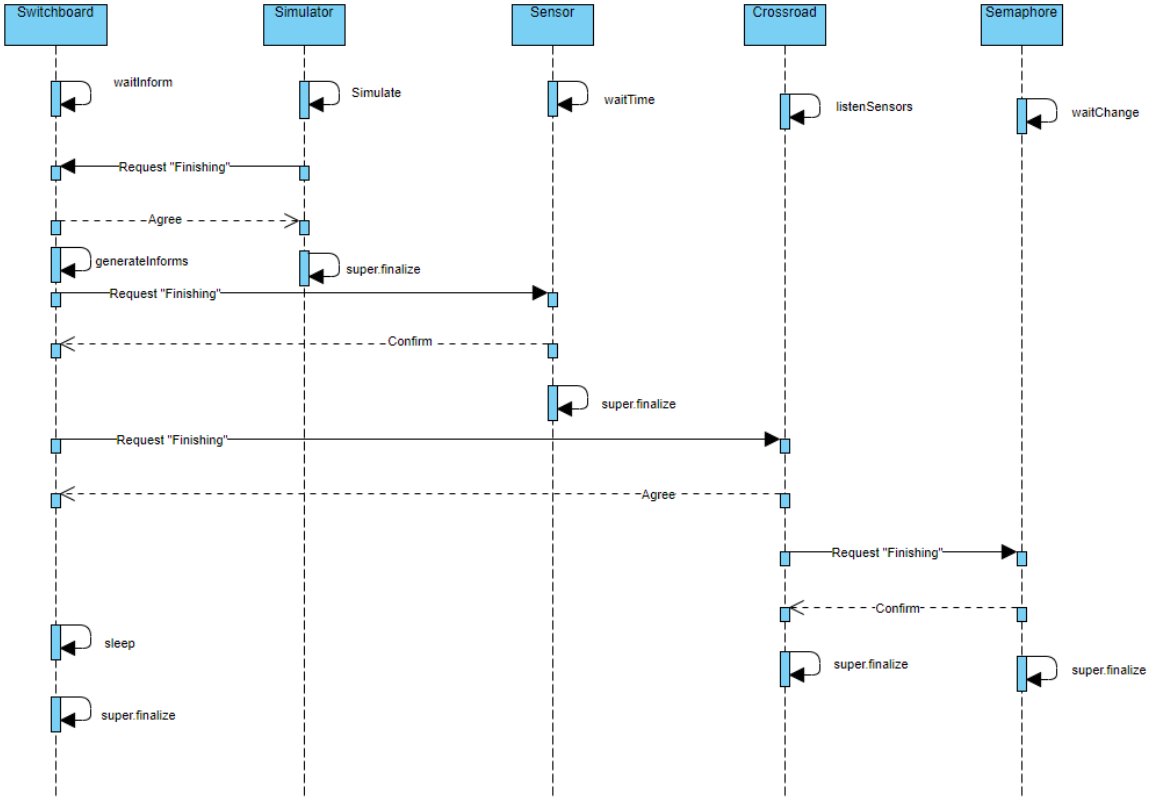

Figura 5.3: Diagrama de secuencia de finalización

Capítulo 6

Diagrama de Clases

En este capítulo veremos todas las clases existentes en MURAT. Por un lado, exponemos un diagrama de clases (Figura6) que contiene únicamente los agentes involucrados y sus clases padre. Estos agentes están representados con sus atributos y no métodos porque consideramos que no son relevantes y dificultarían la visión global del diagrama.

El otro diagrama de clases mostrado (Figura6) muestra las clases existentes del proyecto, únicamente representadas por su nombre de clase. Además de los agentes ya conocidos, encontramos la clase *Constants* y *AgentStates*, las cuales contienen exclusivamente cualquier cadena de texto reutilizable o parámetros necesarios para la simulación. En *AgentStates* se encuentran los distintos identificadores de estados de todos los agentes.

Las clases *SuperAgent* y *ServerAgent* son una mejora con respecto a la plataforma utilizada *Magentix2*, que permiten una mayor capacidad de soportar un envío considerable de mensajes. De esta forma, es posible realizar una simulación de un día completo, segundo a segundo.

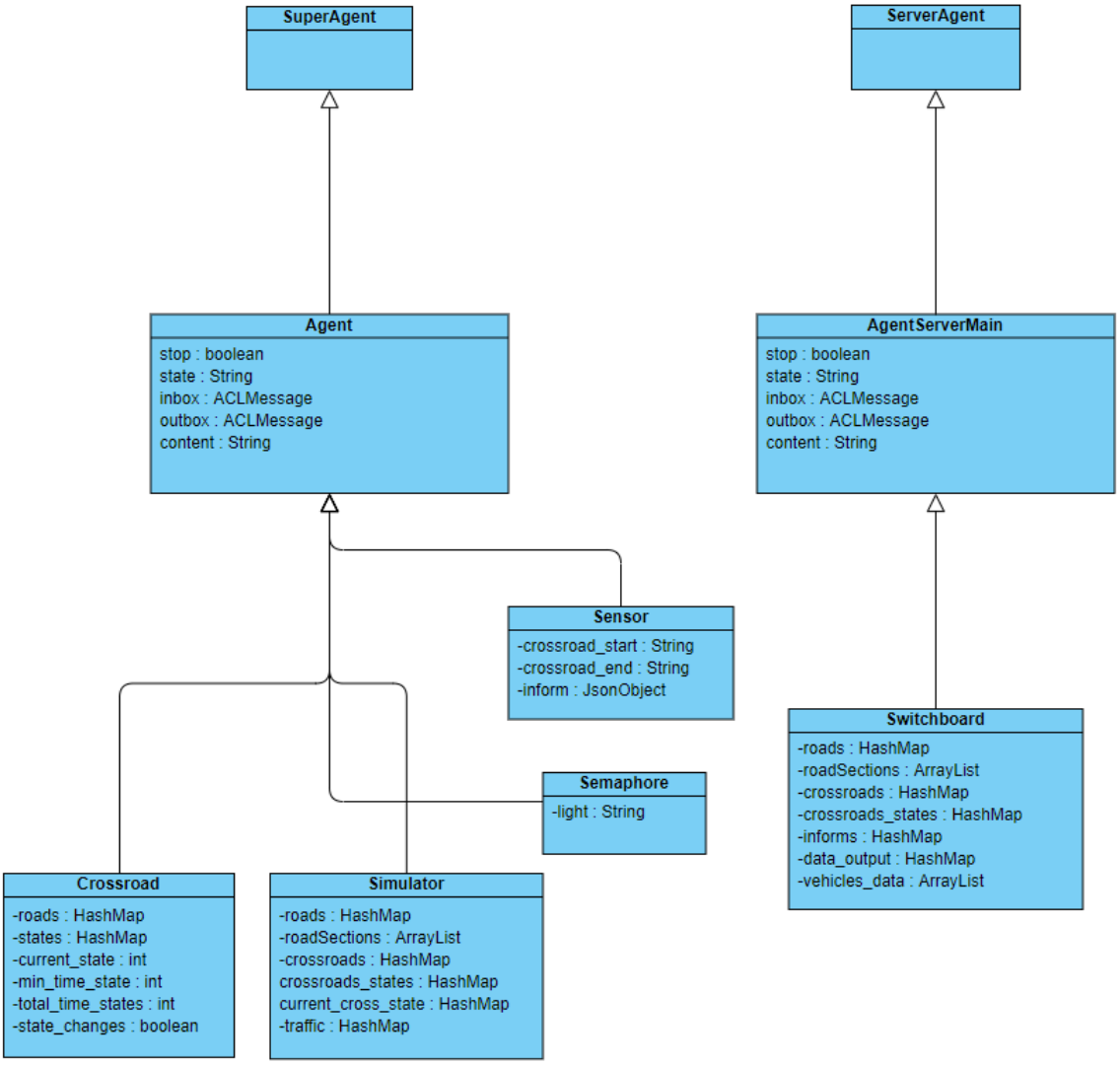

Figura 6.1: Diagrama de clases de los agentes

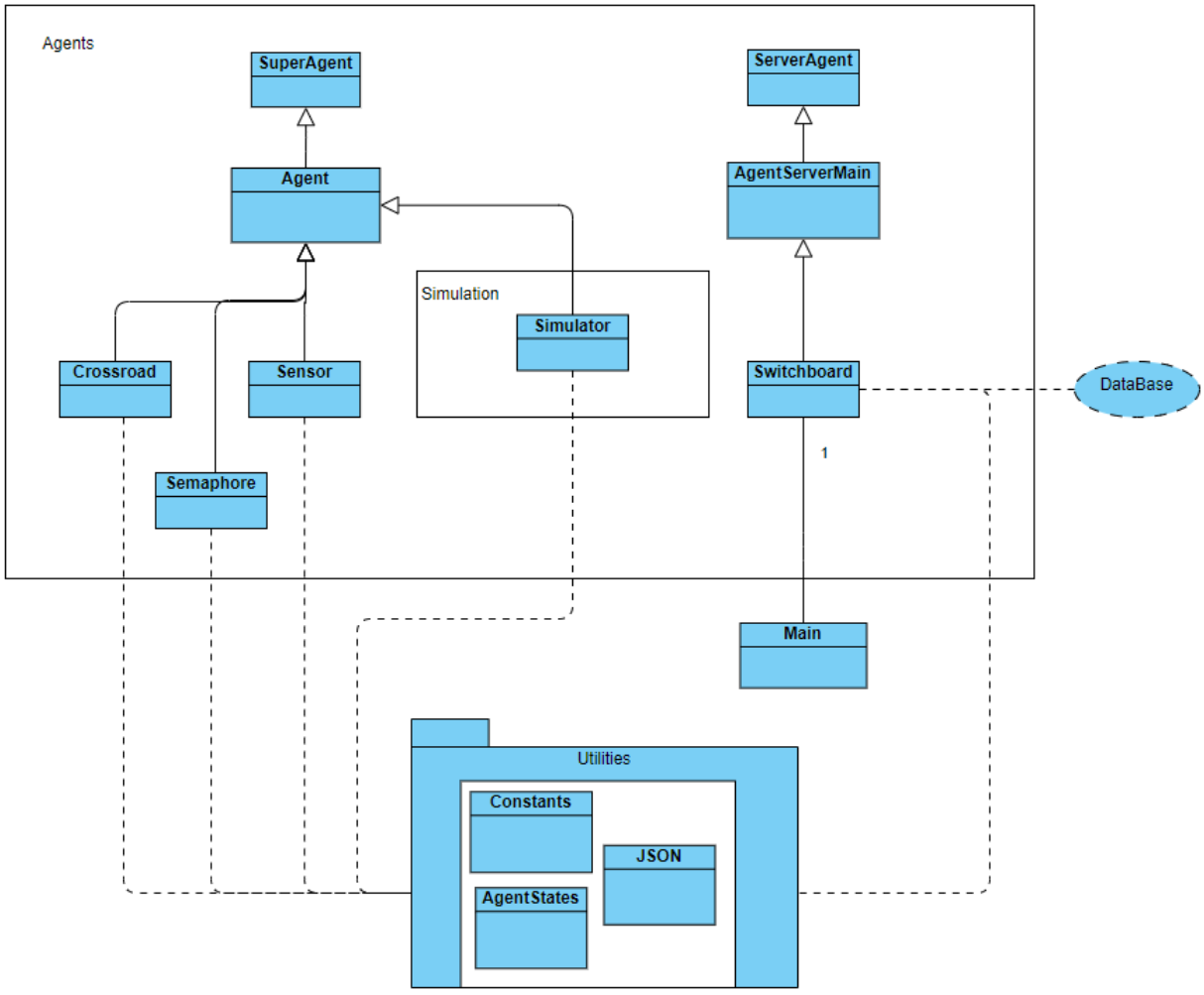

Figura 6.2: Diagrama de clases de MURAT

Capítulo 7

Base de datos

En este capítulo hablaremos de la base de datos y cómo estructuramos los cruces y tramos para poder ser modelados. La información del mundo real únicamente es almacenada en dos ficheros *csv*, uno llamdo *Tramos* (Figura 7.1), el cual contiene los tramos de carretera con sus características y otro llamdo *Cruces* (Figura 7.2), que contiene los distintos estados de semáforos de cada uno de los cruces.

7.1. Tramos de carretera

Se ha considerado un tramo de carretera aquel que en su principio o fin, conecta con un cruce. A continuación, explicamos cada una de sus características:

- ID: Es el identificador o nombre del tramo de carretera.
- CrossroadStart: Es el nombre del cruce que se encuentra al comienzo del tramo.
- CrossroadEnd: Es el nombre del cruce que se encuentra al final del tramo.
- DirStart: Es la orientación relativa del tramo al cruce de inicio. Norte, Sur, Este u Oeste.
- DirEnd: Es la orientación relativa del tramo al cruce de destino. Norte, Sur, Este u Oeste.
- Capacity: Es la longitud en metros de la/s vía/s del tramo.
- NumRoads: Es el número de vías que contiene el tramo de carretera.

ID	CrossroadStart	CrossroadEnd	DirStart	DirEnd	Capacity	NumRoads	MustDir
Calle Rector Marin Ocete-1	root	C1	NA	N	125	2	NA
Calle Rector Marin Ocete-2	C1	C2	S	W	168	2	NA
Calle DR.Jaime Garcia Royo	root	C1	NA	W	180	1	NA
Calle DR.Severo Ochoa-2 NS	root	C2	NA	N	135	2	NA
Calle DR.Severo Ochoa-2 SN	C2	leaf	N	NA	135	1	NA
Calle DR.Severo Ochoa-3	C2	leaf	N	NA	135	2	NA
Calle Manuel Gomez Moreno	C2	leaf	E	NA	88	1	NA
Calle DR.Severo Ochoa-1 NS	C2	leaf	S	NA	150	2	NA
Calle DR.Severo Ochoa-1 SN	root	C2	NA	S	150	2	NA

Figura 7.1: Tramos de carretera

ID	State	N	S	E	W	Time
C2	1	V	V		R	20
C2	2	R	R		V	10
C2	3	V	R		R	15
C1	1	R			V	20
C1	2	V			R	25

Figura 7.2: Configuración de semáforos de cada cruce

- MustDir: Contiene la orientación en la que es obligatoria la circulación de vehículos al salir del tramo de carretera.

7.2. Estados de cada cruce

En este fichero, están almacenados todos los estados de cada configuración de semáforos de todos los cruces. Veamos brevemente su significado:

- ID: Identificador o nombre del cruce.
- State: Identificador del estado de configuración de semáforos.
- Time: Tiempo de duración predeterminado inicial de cada configuración.
- N,S,E,W: Indica la luz del semáforo, si existe, del tramo de entrada con esa orientación.

Parte III

Simulación y Resultados

Capítulo 8

Simulación

Se ha realizado una simulación de uno de los cruces con más congestión vehicular en Granada y uno de sus vecinos. Hablaremos de cruce C2 cuando se trata de la intersección de la calle *Rector Marín Ocete* y *Dr. Severo Ochoa*, y de cruce C1 cuando se trata de su intersección vecina, las calles *Rector Marín Ocete* y *Dr. Jaime García Royo*. Representando la realidad lo mejor posible, se ha establecido una frecuencia de generación de vehículos distinta dependiendo de la hora del día que es:

- 00:00h - 07:00h: Bajo
- 07:00h - 09:00h: Intenso
- 09:00h - 14:00h: Normal
- 14:00h - 16:00h: Intenso
- 16:00h - 22:00h: Normal
- 22:00h - 00:00h: Bajo

En la simulación realizada, todos los vehículos generados se tratan con las mismas características en cuanto a velocidad y longitud. Se ha simulado la duración de un día completo segundo a segundo. Esto es posible gracias al agente Simulator. Podemos ver el mundo simulado en la figura 8.

En cada iteración de su método *Simulate*, el agente genera vehículos por cada tramo de carretera raíz. Es decir, el tramo no tiene ningún cruce asociado de inicio. Se comprueba si existe espacio en cualquiera de las vías del tramo, y si es así, se añade el vehículo al comienzo de dicho tramo.

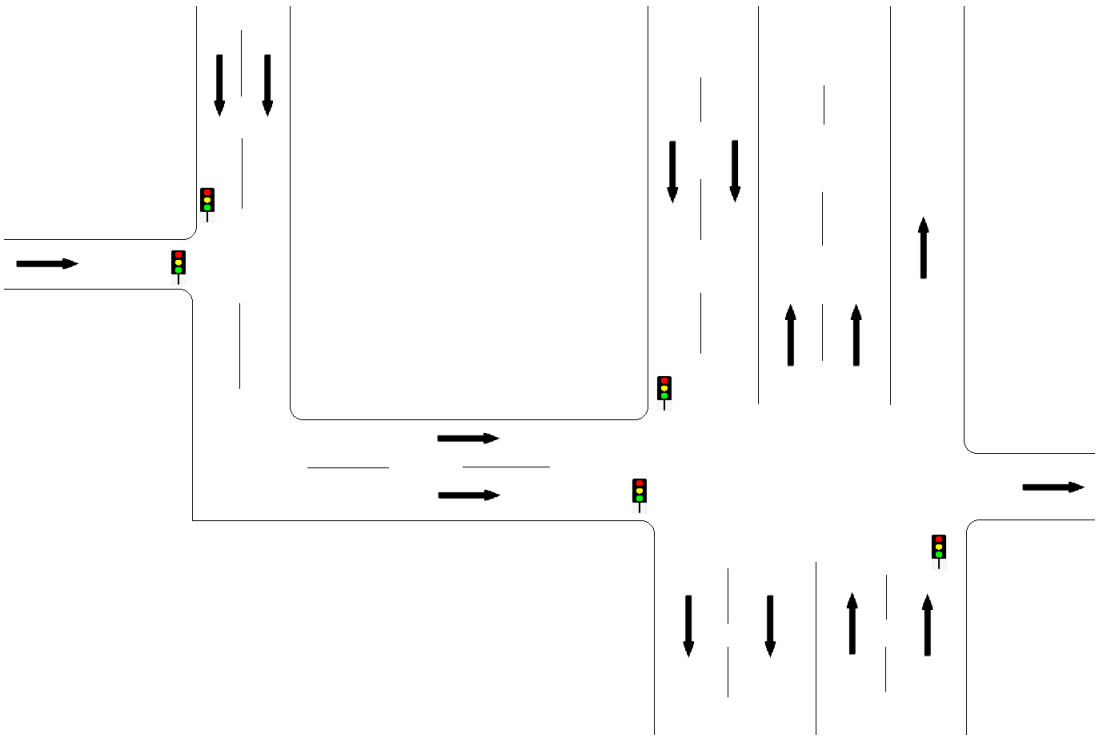

Figura 8.1: Mundo Simulado

Una vez que se han generado vehículos en todos los tramos raíz, siempre y cuando haya sido posible, el simulador prosigue con la canalización de los vehículos. Por cada uno de los vehículos existentes en el mundo simulado, el simulador los avanza según su velocidad media. Siendo posibles varios casos:

- El vehículo no puede avanzar porque el vehículo de delante no se ha movido. Esto ocurre cuando la el semáforo de esa calle está en rojo, o cuando hay tal saturación de tráfico que los vehículos no pueden moverse.
- El vehículo avanza en el tramo en el que se encuentra hasta que recorre su distancia media o hasta que alcanza al vehículo de delante con una separación de un metro.
- El vehículo avanza y sale del tramo. Si nos encontramos en un tramo hoja, el vehículo es sacado de la simulación. Por otro lado si el vehículo pasa a través del cruce, se introduce en una vía de salida de dicho cruce siempre y cuando la configuración de semáforos lo permita.

8.1. Lógica de Cruce

La lógica de los cruces se ha limitado a un sencillo pero potente principio de priorizar el paso a las calles de entrada que tienen un porcentaje de ocupación mayor que un valor establecido. Si se escoge para priorizar un estado de semáforos, no se cambia directamente a él, sino que se modifican los tiempos de cambio. Se aumenta el tiempo del estado que prioriza el paso del tramo afectado y se disminuye el tiempo del resto de estados, manteniendo siempre un tiempo mínimo en cada estado y un tiempo total constante.

Capítulo 9

Resultados

En este apartado vamos a analizar las distintas gráficas obtenidas resultantes de dos simulaciones. Una simulación en la que no se ha tenido en cuenta la lógica de los cruces y otra en la que sí.

Así, las líneas mostradas en las gráficas en azúl, corresponden con la simulación en la que se ha utilizado la lógica, y las líneas en naranja con la simulación sin lógica.

9.1. Vehículos en la simulación

Comenzaremos analizando las Figuras 9.1, 9.2 y 9.3.En la figura 9.1 se representa el acumulado de coches totales generados por el simulador. Observamos que con la lógica el simulador es capaz de generar más de diez mil vehículos al final del día que en la simulación sin lógica.

En la figura 9.1, se representa justo lo contrario, el acumulado de vehículos que han salido de la simulación en un día. Nuevamente existe una diferencia notable de más de diez mil vehículos con y sin lógica. Esto nos demuestra, que no simplemente se han generado más vehículos, si no que, el mundo representado, ha sido capaz de gestionar y absorber esta gran diferencia.

En la figura 9.1, se representa el tiempo medio que ha tardado un vehículo desde que es generado por el simulador, hasta que llega al final de un tramo de carretera que es hoja, y es sacado de la simulación. Esta gráfica es muy representativa de cómo se comporta el tráfico con y sin lógica de cruces.

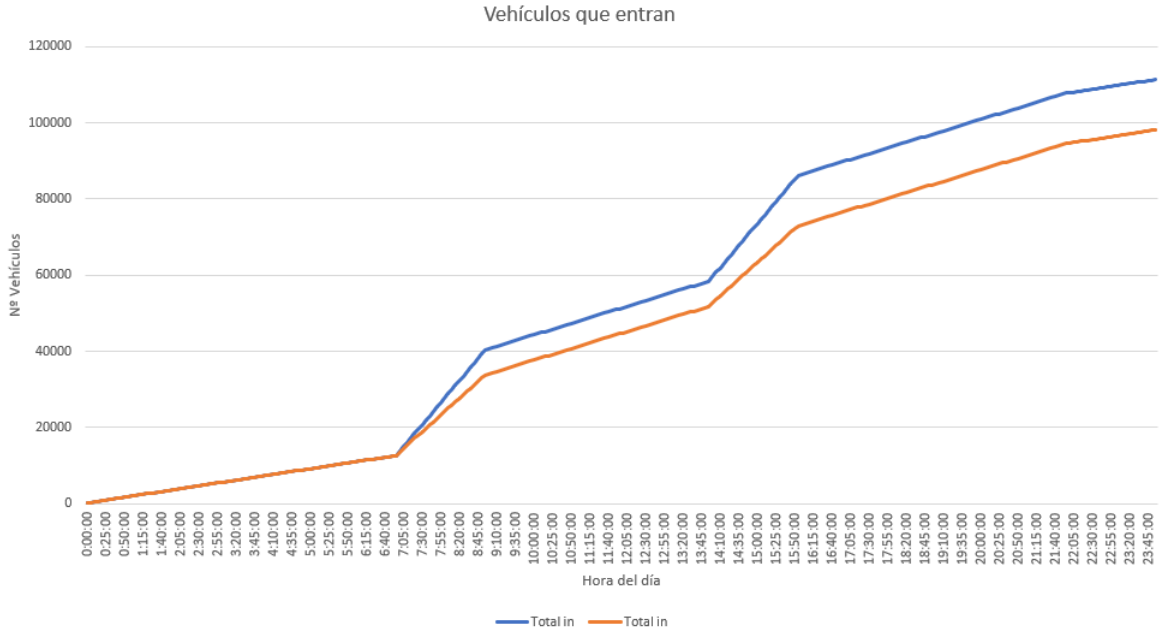

Figura 9.1: Nº de vehículos que han entrado en la simulación

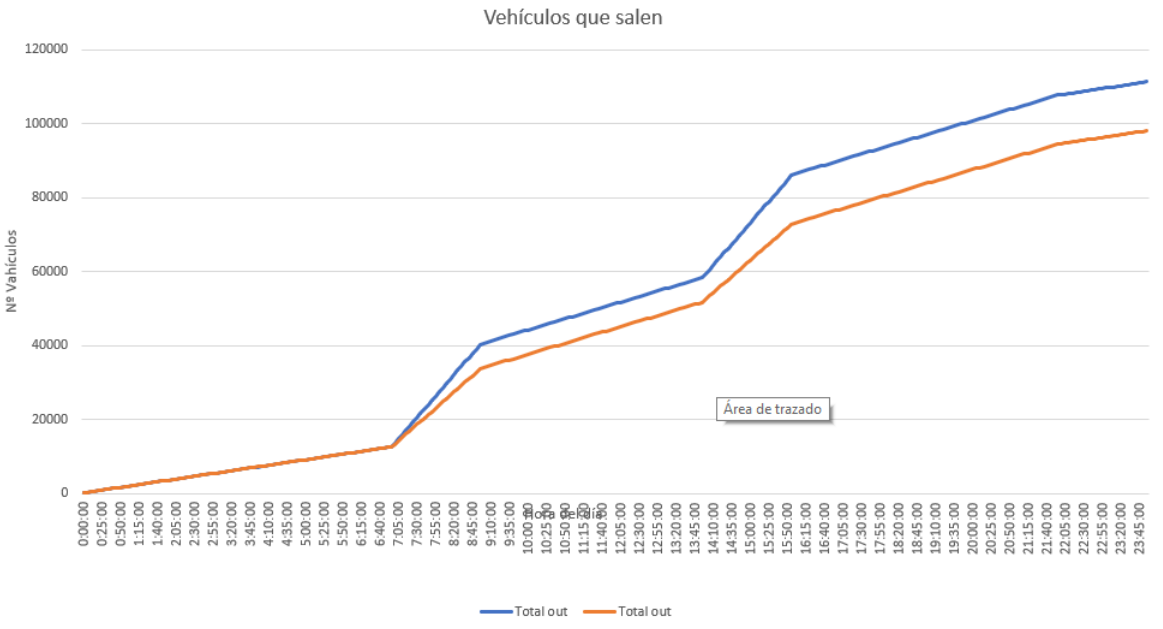

Figura 9.2: Nº de vehículos que han salido de la simulación

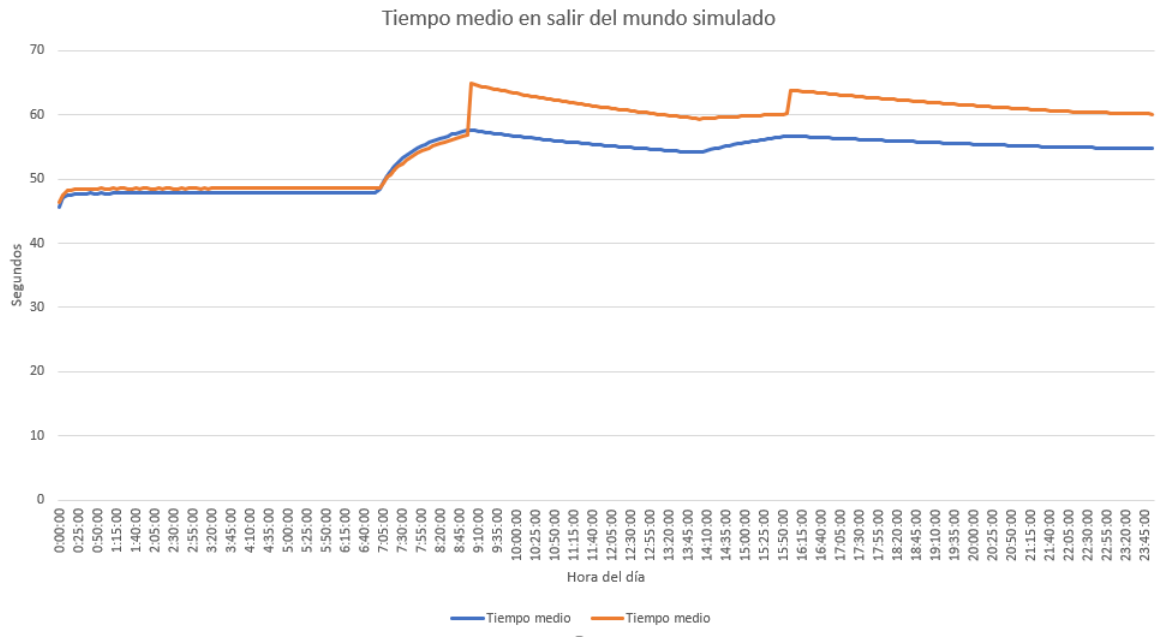

Figura 9.3: Tiempo medio de cada vehículo en salir del mundo

Primero, sabiendo que la simulación ha sido capaz de generar y retirar del mundo una diferencia de diez mil vehículos, además, el tiempo medio de cada uno de ellos en el mundo simulado es menor. Se observa como en los picos de alta congestión vehicular el mundo simulado sin lógica no es capaz de absorber este incremento, presentando así un efecto muelle. En cambio, con la lógica, se aprecian también dichos incrementos de tráfico pero de una forma mucho más suavizada.

9.2. Vehiculos totales

La gráfica 9.2, representa el número total de vehículos en un instante dado existentes en la simulación. De aquí deducimos, que en horas de bajo nivel de tráfico, ambas simulación (con y sin lógica) soportan perfectamente la cantidad de vehículos y muestran datos muy parecidos. La diferencia comienza cuando llega el tráfico intenso, donde se puede observar que con la lógica, la simulación mantiene un número total de vehículos por encima.

Analizando esta vez, los intervalos posteriores a un pico de tráfico, vemos que con la lógica la simulación es capaz de canalizar esa diferencia de vehículos totales. La simulación sin lógica, llegado un momento, no admite más vehículos. No solo eso, si no que después de un periodo de tráfico intenso, tiene más dificultad para aliviar sus tramos.

Para apreciar mejor esta diferencia del número de vehículos totales que admite la simulación, veamos la gráfica 9.2. Aquí, podemos observar el acumulado de coches totales a lo largo del día.

Al comienzo, la diferencia no es muy notable, incluso es negativa. Pero a medida que llegamos a las horas e intervalos de máximo tráfico, la diferencia se dispara. Viendo claramente como de nuevo, la simulación está soportando una mayor cantidad de vehículos.

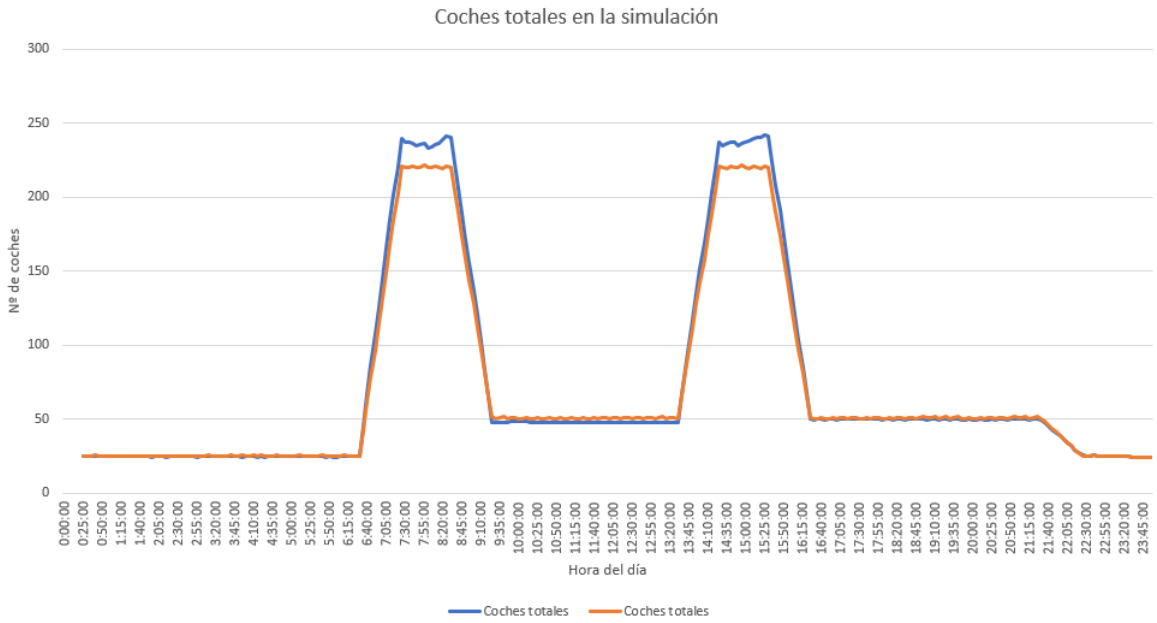

Figura 9.4: Vehículos totales que hay en la simulación en ese instante

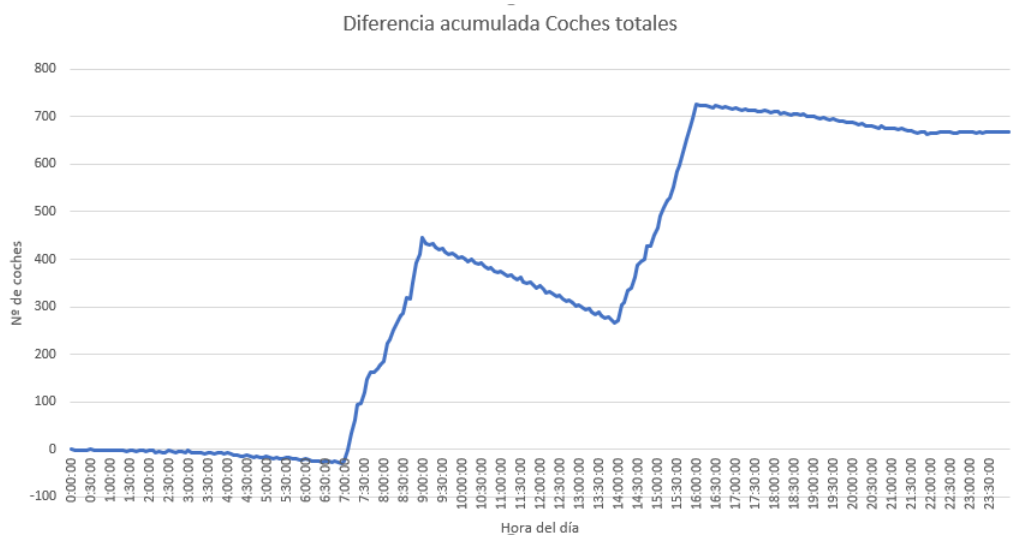

Figura 9.5: Diferencia acumulada del total de vehículos en la simulación

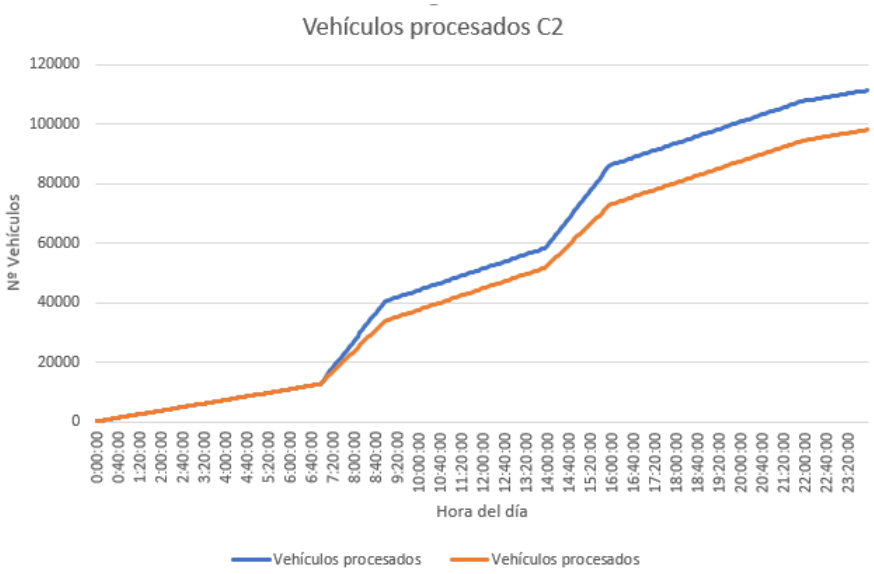

Figura 9.6: Intensidad vehicular de C2

9.3. Cruces

Aquí, hablaremos de la intensidad vehícular que ha soportado cada cruce. Entendemos por intensidad el número de coches que ha pasado por dicho cruce en el tiempo en el que se envía el informe. En este caso cada cinco minutos.

Sabiendo esto, las gráficas 9.3 y 9.3, muestran esta intensidad acumulada a lo largo del día para su correspondiente cruce. Si nos detenemos en el cruce C1, la diferencia al final del día es de más de mil vehículos que han pasado por el cruce.

Analizando la gráfica del cruce C2, vemos el mismo comportamiento. Existe una diferencia notable, siendo esta, igual al número de vehículos procesados por la simulación, más de diez mil. Esto sucede porque todas las calles hoja, salen del cruce en cuestión.

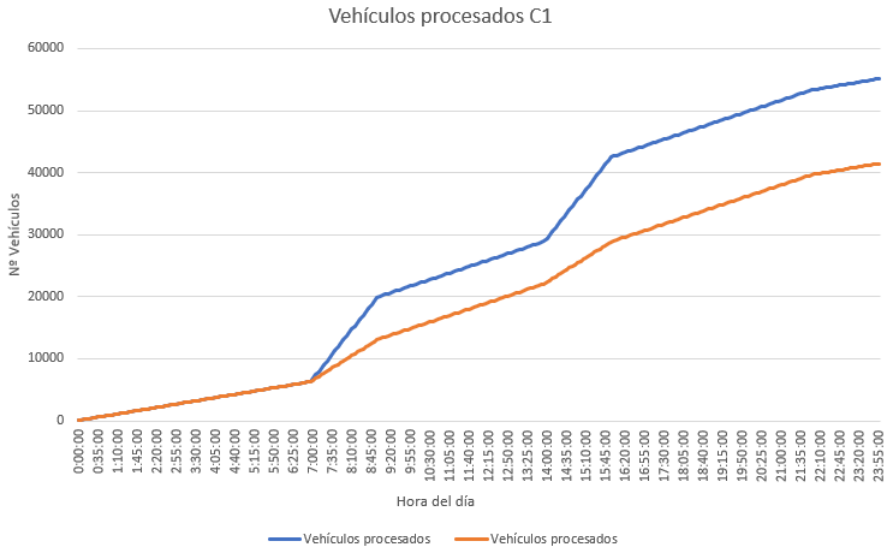

Figura 9.7: Intensidad vehicular de C1

9.4. Conclusiones

Llegados a este punto, podemos afirmar, que el uso de un sistema multiagente para la gestión del tráfico en Granada tendría altos beneficios. Ya conocemos que circularían a lo largo del día una diferencia muy positiva de vehículos y que, su tiempo de espera, sería menor. Esto no solo facilita un tráfico más fluido, si no que ahorra tiempo a las personas. El número de retrasos en los destinos se vería considerablemente reducido, la forma de conducir sería más relajada, etc.

9.4.1. Escalabilidad

MURAT es aplicable a cualquier ciudad y a cualquier número de intersecciones dada su naturaleza arquitectónica. No solo es fácilmente escalable a nivel de instalación, sino a nivel de aplicación. Está preparado para admitir cualquier nueva configuración, alterando únicamente la base de datos.

9.4.2. Mantenimiento

El paso a la realidad de MURAT, en cuanto a costes se refiere, es muy asequible. Bastaría con colocar los sensores en cada tramo de carretera con problemas de congestión vehicular. Hablar de cómo se podría mejorar, incorporar a distintas aplicaciones, escalabilidad

Bibliografía

[1] "Birgit burmeister, afsaneh haddadi, guido matylis, daimler-benz research. systems technology. application of multi-agent systems in traffic and transportation.," 1997.

[2] "Bo chen member ieee, harry h, cheng senior member ieee. a review of the applications of agent technology in traffic and transportation systems," 2010.

[3] "Josefa z. hernández, sascha ossowski, ana garcía-serrano. multiagent architectures for intelligent traffic management systems.," 2002.

[4] "Roger p. roess., elena s. prassas. traffic engineering. fifth edition. . william r. mcshane.," 2018.

[5] "Magentix2. universidad politécnica de valencia. www.gti-ia.upv.es/sma/tools/magentix2/."

[6] "Jennings, n.r., sycara, s., and wooldridge, m. a roadmap of agent research and development. autonomous agents and multi-agent systems i. kluwer," 1998.

[7] "Virginia dignum. javier vázquez-salceda. frank dignum. a model of almost everything: Norms, structure and ontologies in agent organizations," 2004.

[8] "jar-download.com/artifacts/org.json," 2020.

[9] "poi.apache.org/," 2019.