Red Oportunista para la Regulación del Tráfico

Cristian Gómez Rodríguez

Resumen– En este documento tratamos la comunicación de la infraestructura con los vehículos y de la comunicación vehículo a vehículo. Con los actuales avances en el campo de la conducción autónoma, la comunicación entre los diferentes elementos que influyen en la circulación es vital para su éxito. Para dicha comunicación nos basamos en una implementación de redes tolerantes al retardo (DTN) como es la de Active DTN. En este articulo se presenta una solución de comunicación no centralizada y con un alto grado de anonimato. En concreto tratamos el paso de la programación de los semáforos a los vehículos que se ven afectados por estos. Para comprobar su correcto funcionamiento presentamos unas simulaciones donde podemos ver los resultados de la propagación de los mensajes según la densidad del tráfico y el porcentaje de vehículos que llegan al semáforo con la programación obtenida con anterioridad.

Palabras clave– Redes tolerates al retardo (DTN), Active DTN, conducción autónoma, enrutamiento, alcance

Abstract– In this paper, we talk about the communication between the infrastructure and the vehicles and vehicle to vehicle. With current advances in the autonomous driving research, we think that the communication between elements that interfer circulation of vehicles is vital to its success. For this communication we propose an implementation of delay-tolerant network (DTN) as is the Active DTN. In this article a solution of non-centralized communication and with a high degree of anonymity is presented. Specifically we solve the send of cicles of traffic lights to vehicles that are affected by them. To check for proper operation we execute simulations where we can see the results of the spread of the message according to the traffic density and the percentage of vehicles arriving at the traffic lights with programming previously obtained.

Keywords– Delay tolerant networks (DTN), active DTN, autonomous driving, forwarding, scope

1 Introducción

La conducción manual de vehículos causa al año miles de accidentes, la mayoría provocados por fallos humanos. Con los coches autónomos se pretende minimizar el factor humano en la conducción, por consiguiente, los accidentes provocados por estos también se verían disminuidos.

La conducción autónoma es una de las ramas en la que más esfuerzos se están poniendo en el sector automovilístico en los últimos años. Dentro de ésta, podemos englobar los sistemas de ayuda a la conducción que cada vez están más presentes en los vehículos y que algunos de ellos hace unos años los consideraríamos de película de ciencia ficción. Las ayudas a la conducción podrían ser los detectores de carriles o la auto-frenada del coche al detecta una frenada brusca del vehículo de delante.

Un coche autónomo tiene que tener la capacidad de detectar carriles, peatones, señales de tráfico, pero también debe poder comunicarse con otros vehículos o con elementos de la infraestructura para conocer que están haciendo o que harán. Este proyecto trata de esto último, de la comunicación de los vehículos con la infraestructura y entre vehículos.

El problema que motiva este trabajo es la necesidad de conocer el estado actual y el futuro en los actores que intervienen en la circulación y la única manera saberlo con seguridad es que estos mismos nos lo comuniquen. La solución que se plantea está dentro de la parte de comunicación de elementos entre la infraestructura y vehículos y entre vehículos. En concreto lo que tratamos es cómo podemos hacer llegar la programación de un semáforo a los vehículos que tengan una alta probabilidad de pasar por este. También se plantea una manera de codificar la programación de los semáforos y como se muestra la información obtenida dentro de los vehículos. Con la solución que se ofrece en este

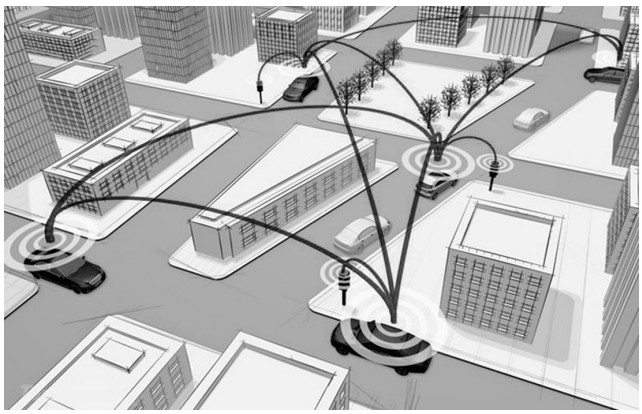

Fig. 1: Esquema general del paso de mensajes entre los semáforos y los diferentes vehículos

documento los vehículos autónomos podrían re-calcular su ruta ideal conociendo el estado de los semáforos o ajustar su velocidad para mejorar en temas de eficiencia energética.
Uno de los puntos que se pretende conseguir es un alto nivel de anonimato por parte de los nodos, por este motivo descartamos una infraestructura de red centralizada y optamos por una red de comunicación basada en el modelo DTN. Utilizando este tipo de red, los semáforos crean mensajes para que luego los vehículos vayan retransmitiendo a los diferentes nodos utilizando unos criterios que más tarde definiremos. Al utilizar un modelo DTN se consigue una solución más económica y con un nivel de disponibilidad mayor, esto es debido a que no hay ningún nodo critico como en un sistema centralizado.
Para conseguir todo lo mencionado en los parrafos anteriores proponemos un algoritmo de encaminamiento y otro de alcance que son los que configuraríamos en la plataforma aDTN y se encargarían de transmitir el mensaje a los nodos o eliminarlo cuando ya no sea valido. Para comprobar que nuestra propuesta de algoritmos de encaminamiento es correcta, realizamos unas simulaciones y presentamos los resultados de estas en forma de porcentaje de nodos que llegan a los semáforos con la programación de estos.
El objetivo principal de este trabajo final de grado es implementar un sistema que permita la comunicación de los semáforos con los vehículos que pasaran por ellos para así poder transmitir la información del estado de estos tal y como podemos ver en la figura 1. Para poder cumplir con este objetivo nos vemos en la necesidad de realizar con éxito los siguientes subojetivos con la prioridad especificada en la siguiente lista:

1. Dominar los conceptos básicos sobre las redes DTN para poder aplicarlos en el proyecto.

2. Definir un algoritmo de forwarding de los mensajes para que estos lleguen a los vehículos y se transfieran entre los diferentes nodos que en algún momento dentro de un rango de distancia y tiempo necesiten la programación del semáforo que lleva el mensaje.

3. Crear un algoritmo donde el mensaje se autodestruya debido a la falta de sentido de este, ya sea a causa de estar el vehículo muy lejos del semáforo o que la programación de este ya este obsoleta (Scope).

4. Definir un mensaje donde se contemple todos los estados de los semáforos.

5. Crear una aplicación para obtener las posiciones GPS de la ruta que tiene planificada un vehículo.

6. Utilizar la plataforma aDTN de Senda para poder implementar los algoritmos de forwarding y scope que más tarde se probaran en el simulador.

7. Crear un sistema visual utilizando un mapa para poder ver en los vehículos los estados de los semáforos de los cuales tenemos información y el tiempo restante hacia el cambio de estado.

8. Realizar una simulación visual de los algoritmos implementados y valoramos su rendimiento en términos de alcance del mensaje y porcentaje de vehículos que llegan al semáforo sabiendo la programación de este, utilizando el software de simulación de redes DTN, “The One” (The Opportunistic Network Environment simulator)[2].

El documento empieza poniendo en contexto al lector del actual estado de las redes DTN y explicaremos que es active DTN en la sección 2. Continuamos en la sección 3 exponiendo la metodología usada en el proyecto y la planificación seguids. En la sección 4 se especifica todos los desarrollos realizados y se propone un algoritmo de forwarding y otro de scope. Para acabar se prueba la efectividad de los anteriores algoritmos en unas simulaciones y presentamos los resultados de estas en la sección 5.

2 Estado del Arte

En esta sección se hace una breve explicación sobre el funcionamiento las redes DTN y la implementación de estas con Active DTN.

2.1. Redes DTN

La comunicación en Internet se basa en la conmutación de paquetes. Los paquetes se dividen y son parte de un bloque de datos de usuario que viajan independientemente del origen al destino a través de una red conectados por los routers.
Cada paquete que conforma un mensaje puede tomar un camino diferente a través de la red de routers. Si se desconecta un enlace, los routers redirige los paquetes a un enlace alternativo. La cabecera de estos mensajes contienen un destino y otra información que determina la forma en que el paquete viaja de un router a otro.
Existen casos donde la comunicación tradicional que se usa en Internet no es eficiente o es imposible de que funcione. Algunos de estos casos podrían ser en conexiones intermitentes donde no siempre existe un camino entre origen y destino. Otro caso son las conexiones que sufren de largos retardos, esto hace que algunos de los protocolos que usa Internet sean ineficientes o simplemente no funcionan.
DTN[3] consigue evitar los problemas anteriormente comentados. Este método envía un mensaje entero o un fragmento de este a través de una serie de nodos, estos nodos se guardan el mensaje y cuando encuentran un candidato idóneo se lo pasan con la intención de que este nuevo nodo

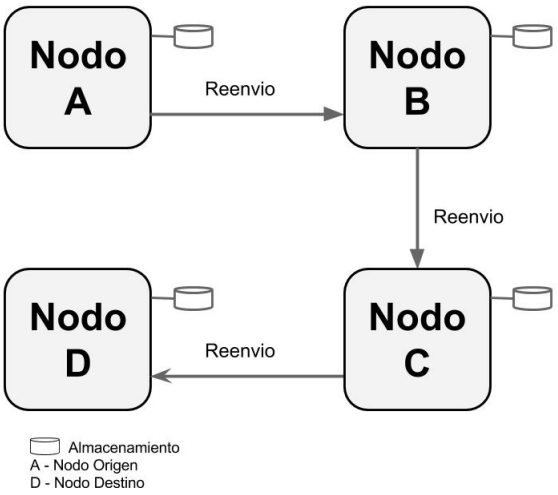

Fig. 2: Estructura de almacenamiento y envío de mensajes en una red DTN

portador sea capaz de realizar un encaminamiento a través de otros nodos hasta el destino final como vemos en la figura 2.

Cuando los nodos que se comunican están en movimiento, los enlaces de esos pueden ser inestables y provocar una conectividad intermitente. En Internet, cuando tenemos una conectividad intermitente sufrimos pérdida de datos. Los paquetes que no se pueden transmitir se descartan, provocando que el protocolo TCP haga una retransmisión cada vez más lenta. Si esta situación persiste, TCP finaliza la sesión y ese mensaje no es enviado. Por este motivo, considerando el continuo movimiento de nuestros nodos (vehículos), decidimos que la forma más eficiente de realizar este proyecto es utilizando una arquitectura de red DTN, en concreto utilizando la implementación de Active DTN que se explica en la siguiente sección.

2.2. Active DTN

Active-DTN[4] es una variación de la arquitectura DTN desarrollada por el grupo de investigadores SENDA[7]. En esta arquitectura los paquetes se comportan de una forma activa. Los paquetes no solo transportan datos, sino que tienen la capacidad de transportar código para definir su futuro enrutamiento. Esta variación de DTN es completamente compatible con su antecesora, si un mensaje aDTN llegase a un nodo DTN, este ignoraría la parte del código de encaminamiento y aplicaría el que tuviese predefinido en su red. El código que nos permite transportar es el referente a su encaminamiento (forwarding), así como a la vida del mensaje (scope) o la prioridad de este.
Hemos visto una breve explicación de el modelo de las redes DTN y como con Active DTN las podemos utilizar aplicando unos mecanismos de encaminamiento diferentes. Ahora seguiremos con la explicación de la metodología que se ha seguido para la realización del proyecto.

3 METODOLOGÍA

La metodología que utilizaremos para realizar el proyecto será una metodología Waterfall ya que se ha realizado una división del trabajo por tareas y teniendo un único recurso, las hemos realizado de manera secuencial.

3.1. Planificación

La planificación la podemos dividir en tres etapas diferentes: preparación y estudio, desarrollo y simulaciones.

3.1.1. Etapa 1: preparación y estudio

Esta primera etapa la enfocamos en definir el proyecto, sus objetivos y el alcance de este. También nos dedicamos al estudio del estado del arte sobre las redes DTN y en buscar las diferentes aplicaciones de esta tecnología. En esta fase del proyecto realizamos el aprendizaje necesario sobre la plataforma Active DTN y hacemos un seguido de pruebas para comprender mejor este tipo de redes. Incluimos en esta etapa la realización del informe inicial del proyecto, donde se explica la definición de este y la metodología que se seguirá.
Si seguimos la lista de tareas de la Figura 3, esta etapa equivale a las tareas 1, 2, 3 y 5.

3.1.2. Etapa 2: desarrollo

En esta etapa nos centramos en el desarrollo de todos los módulos necesarios para poder llevar a cabo el proyecto.
Empezamos realizando un script donde indicando un punto de origen y un destino, este, utilizando la API de Google Directions, nos devuelve las coordenadas GPS que compone la ruta que une estos dos puntos.
Seguimos con la implementación del algoritmo de forwarding que será el que irá dentro de la plataforma aDTN y decidirá si un mensaje es encaminado a un nodo o no. Esta decisión la tomará en función a la ruta que hemos obtenido en el script mencionado en el párrafo anterior. Otro algoritmo que desarrollamos es el de Scope, este decide la vida que tendrá el mensaje que enviamos. Para este algoritmo nos basamos principalmente en la distancia actual hasta el origen y del tiempo que ha pasado desde la generación del mensaje.
Siguiendo con esta segunda etapa nos vemos con la necesidad de crear una codificación de los estados de un semáforo para poder transmitirlo en el cuerpo del mensaje. El objetivo de este proyecto no es la de realizar una codificación de los semáforos, por ese motivo la hemos realizado de una manera que consideramos sencilla para poderla procesar y que satisfaga nuestras necesidades.

Para seguir con esta segunda etapa desarrollamos una interfaz gráfica para facilitar la visualización de la información que nos llega desde los semáforos. Este visor es una interfaz web que nos inserta los semáforos de los cuales tenemos información en un mapa y su estado actual.

Para finalizar con esta etapa realizamos el informe de progreso 1 donde explicamos el estado actual del proyecto.
Si seguimos la lista de tareas de la Figura 3, esta etapa equivale a las tareas 4, 6, 7, y 8.

1	Primera reunión
2	Estado del Arte DTN
3	Aprendizaje y pruebas aDTN
4	Obtención de rutas GPS
5	Informe inicial
6	Algoritmo de forwarding y scope
7	Informe de progreso 1
8	UI estado de los semáforos
9	Aprendizaje del simulador The One
10	Informe de progreso 2
11	Implementar simulación en The One
12	Interpretar resultados The One
13	Articulo
14	Dosier
15	Poster
16	Preparación de la defensa
17	Defensa

Fig. 3: Planificación global del proyecto

```
{
  "semaforos": [
  {
   "id": 1,
   "lat": 41.498275,
   "lon": 2.111384,
   "state": 2,
   "stateChange":[1466877600, 1466877610,
                  1466877620, 1466877630,
                  1466877640, 1466877650,
                  1466877660, 1466877670,
                  1466877680, 1466877690,
                  1466877700, 1466877710],
    "lifeTime" : 1466877800
   }
  ]
}
```

Fig. 4: Ejemplo de codificación de los mensajes que los semáforos crearan para comunicar su programación.

3.1.3. Etapa 3: Simulaciones

Esta etapa la consideramos la más importante de todo el proyecto debido a que por falta de tiempo y recursos nos vemos incapaces de probar el sistema en un escenario real. Por este motivo diseñamos una simulación utilizando el simulador de redes DTN The ONE.
En esta etapa nos dedicamos la mayor parte del tiempo al aprendizaje de esta herramienta de simulación y haciendo los cambios precisos para adaptarla a nuestras necesidades. También se realiza el informe de seguimiento 2, donde se muestra el estado actual del proyecto y los resultados que hemos obtenido, en nuestro caso en forma de herramientas o algoritmos. Si seguimos la lista de tareas de la Figura 3, esta etapa equivale a las tareas 9, 10 y 11.
Las tareas que quedan fuera de las etapas las consideramos dentro de una etapa final donde documentamos y preparamos el final del proyecto.

Hemos realizado una explicación sobre la metodología y la planificación de tareas que se han realizado al largo del proyecto. A continuación entraremos en la sección de desarrollo donde explicaremos con mas detalle estas tareas.

4 DESARROLLO

A continuación se explicará el desarrollo del proyecto, se ha dividido en diseño y implementación.

4.1. Diseño

En esta sección se presentara una propuesta de codificación de los semaforos, un algoritmo de forwarding y otro de scope.

4.1.1. Codificación de los semáforos

El objetivo de este proyecto no es la de realizar una codificación de los semáforos, por ese motivo la hemos realizado de una manera que consideramos sencilla para poderla procesar y que satisfaga nuestras necesidades. Hemos utilizado el formato json para realizar la codificación, en concreto el formato que podemos ver en la figura 4. Los campos que contiene este mensaje son los siguiente:

- semáforos: array de objetos que representan semáforos.
- lat y lon: posicionan el semáforo en el mapa.
- state: 0 = rojo, 1 = verde. Este es el estado en el que está el semáforo en el momento de enviar el mensaje.
- stateChange: array de unix timestamp que indican en qué momento se debe cambiar de estado.
- lifetime: la fecha en formato timestamp en la que el mensaje se descartara.

4.1.2. Algoritmo de routing

Este algoritmo será el que decidirá si un nodo concreto recibe el mensaje o no. Para realizar este algoritmo nos basamos en que cada nodo tiene una ruta planificada en forma de array de posiciones como la que extraemos en la subsección API Google Directions.
Con estas posiciones podemos comprobar si el semáforo se encuentra a X metros de algún punto de nuestra ruta. Los metros de distancia del punto al semáforo se han establecido en 100 metros
El mayor problema que nos hemos encontrado es que los puntos que devuelve Google cuando hacemos una petición a su API son los mínimos que se necesita para poder dibujar la ruta en un mapa a base de líneas rectas. Nos encontrábamos con casos de carreteras rectas que la dispersión de los puntos era muy grande. Para solucionar este problema buscamos la distancia del semáforo a la recta que forman dos puntos contiguos del array de posiciones

que forma nuestra ruta.
Para saber si el semáforo está o no a una distancia de la recta que forman los dos puntos que nos da Google utilizamos la siguiente formula:

$$P_1 = (x_1, y_1)$$

$$P_2 = (x_2, y_2)$$

$$dist\,(P_1, P_2, (x_0, y_0)) =$$

$$\frac{\mid (y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1 \mid}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}$$

Donde P1 y P2 son los puntos de nuestra ruta que forman la recta y (x0, y0) es el punto que queremos saber la distancia.

4.1.3. Algoritmo de scope

El algoritmo de scope es el que define la vida del mensaje. El mensaje se descartara pasados 5 minutos de su creación o si está a una distancia mayor a 3 km de su origen.
Para el calculo de la distancia utilizamos la distancia euclidiana entre dos puntos en un plano, la formula es la siguiente:

$$P_a = (x_a, y_a)$$

$$P_b = (x_b, y_b)$$

$$d(P_a, P_b) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

No vemos necesario tener en cuenta la curvatura de la tierra debido a que las distancias entre los puntos que vamos a calcular son muy pequeñas para tener en cuenta dicha curvatura ya que el error obtenido al no contemplarla es insignificante.

4.2. Implementación

En esta sección se explican las tareas de obtener una ruta con la API de Google Direction y la interfaz realizada para ver la información de los semáforos.

4.2.1. API Google Directions

Para poder desarrollar esta aplicación que nos ayudara a realizar la parte de simulación que implementaremos en un futuro nos hemos ayudado en la API de Google Maps, en concreto en Google Maps Directions.

Con esta API realizamos una petición http donde le indicamos el origen, el destino y si queremos algunas restricciones como podría ser evitar peajes o autopistas. El formato para realizar esta petición es el siguiente:

```
https://maps.googleapis.com/maps/
api/directions/json?origin={ORIGIN}&
destination={DESTINATION}&key={KEY}
```

Google contabiliza el número de peticiones que se hacen a sus servicios y lo hace utilizando el campo de KEY que indicamos en la petición anterior. Esta key es única por cada proyecto y se consigue registrando el proyecto en la consola de Google. Se contabilizan las peticiones ya que el servició es gratuito siempre que no se pase de 25.000 peticiones al día.

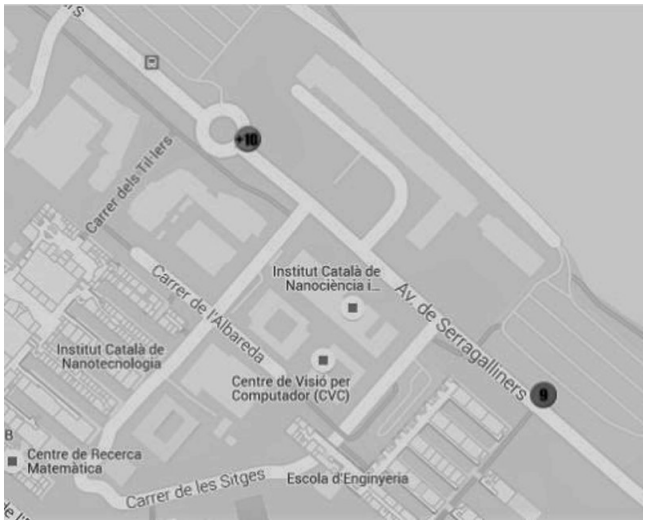

Fig. 5: Ejemplo de visualización de la información de los semáforos activos que un vehículo tiene conocimiento

La respuesta que recibimos es en formato json o xml, según le indiquemos en la petición. La parte que nos interesa a nosotros de la respuesta es el valor del campo "overview_polyline". En este campo, Google nos da las posiciones GPS que forman nuestra ruta en un formato codificado. El aspecto es el siguiente:

4.2.2. UI estado de los semáforos

Esta interfaz la realizamos con el objetivo de que cada nodo visualice los semáforos de los que dispone la programación y su estado actual. La tecnología que utilizamos para desarrollar esta interfaz será JavaScript con la librería de JQuery, la API de Google Maps como visor del mapa. Todo esto lo encapsulamos en un visor web realizado en C++, lenguaje con el que tratamos los archivos donde tenemos toda la información que queremos mostrar.
La interfaz tiene una apariencia como la que vemos en la figura 5.

Cada segundo se van actualizando los semáforos para ver el estado en el que están en el momento actual y cada X segundos, aún no determinados, se cargan los nuevos semáforos que nos han llegado desde la última actualización.
Para conseguir esto hemos tenido que implementar un pequeño script en PHP que nos lea el contenido del fichero cada X segundos y nos lo envíe al navegador para poder actualizar la vista utilizando Ajax. De esta manera y cambiando los iconos de los markers que utilizamos para mostrar los semáforos podemos actualizar el estado de este no solo de rojo a verde, sino el tiempo que le queda para su siguiente cambio de estado.

En este apartado hemos expuesto los diferentes desarrollos realizados al largo del proyecto. En el siguiente apartado veremos los resultados en forma de las simulaciones y los datos obtenidos de estas.

5 Resultados

En esta sección veremos el software utilizado para realizar las simulaciones, The One[2]. También expondremos

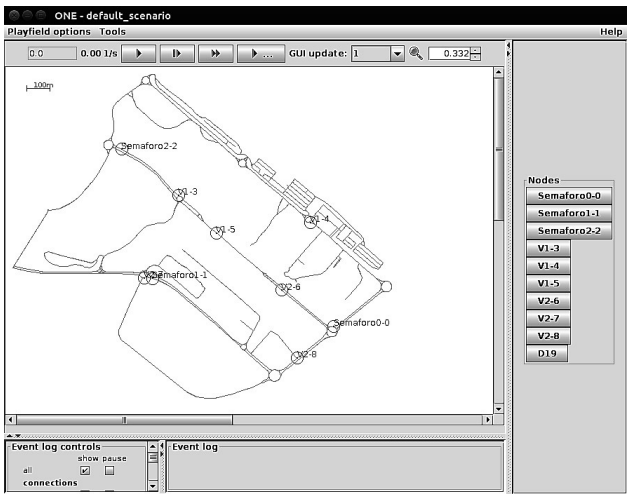

Fig. 6: Ejemplo de visualización del simulador The One

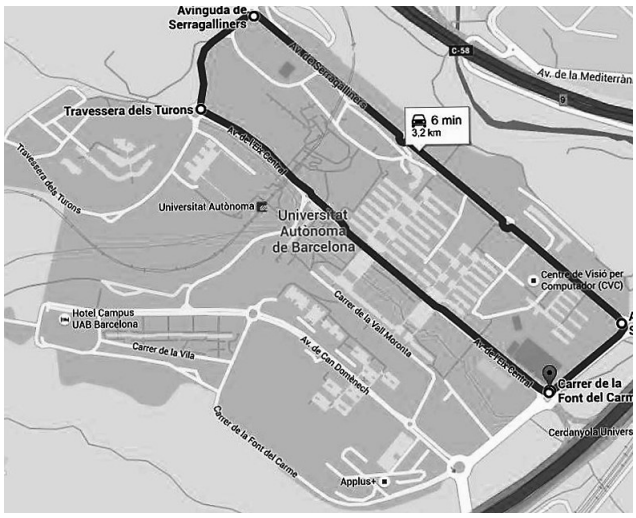

Fig. 7: Ruta que sigue el grupo 1 de vehículos en las simulaciones con The One

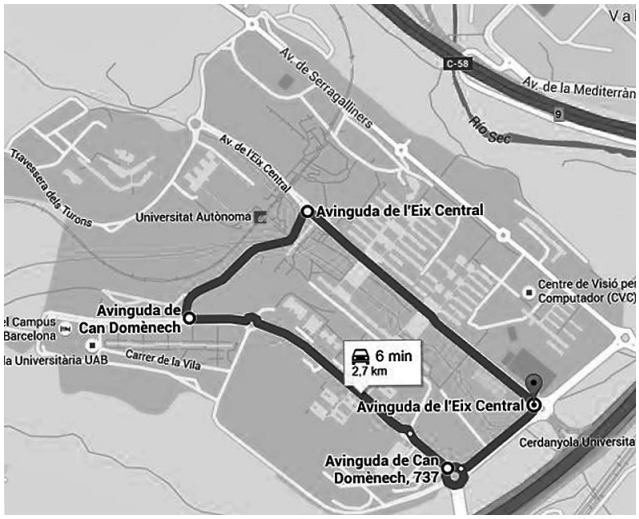

Fig. 8: Ruta que sigue el grupo 2 de vehículos en las simulaciones con The One

las simulaciones diseñadas y los resultados obtenidos de estas.

5.1. Simulador The One

Para poder comprobar la efectividad de los algoritmos de forwarding y scope desarrollados nos vemos en la obligación de probarlos en un entorno simulado debido a que el tiempo que llevaría prepara este sistema en un entorno real sería mayor que el del propio proyecto. Esta simulación la realizamos con el software The One, la interfaz la podemos ver en el figura 6. Este simulador nos permite:

- Generar nodos con diferentes modelos de movimientos en un mapa.
- Enrutar mensajes entre nodos utilizando diferentes algoritmos de encaminamiento DTN.
- Configurar diferentes interfaces en los nodos con distintas características como puede ser intensidad o distancia.
- Visualizar la movilidad de los nodos en tiempo real.
- Producir gran variedad de informes con diferentes estadísticas.

The One está desarrollado en java y es totalmente de código abierto. Las características que mas nos interesan son la posibilidad de crear grupos de nodos con patrones de movimientos diferentes y con rutas predefinidas. De esta manera podemos establecer los diferentes grupos de nodos en función de que semáforo queremos que forme pare su ruta.

5.2. Diseño de las simulaciones

Para poder probar los algoritmos que en apartados anteriores hemos mencionado, se han diseñado una serie de simulaciones. Hemos basado las simulaciones en el mapa de la Universidad Autónoma de Barcelona y en un escenario que simula el tráfico de 4 horas.

Nos basamos en 6 escenarios con diferente densidad de nodos. Se han establecido 3 nodos fijos que simulan los semáforos situados en 3 puntos fijos en el mapa de la Universidad Autónoma de Barcelona que podemos ver en la figura 9. Estos nodos son los encargados de crear los mensajes y enviarlos a los nodos que simulan los vehículos para que estos los vayan retransmitiendo con otros nodos que se va encontrando.

Tenemos 2 grupos de nodos que representan los vehículos, estos grupos tienen definidas dos rutas diferentes, especificadas en las figuras 7 y 8, donde la ruta 1 y la ruta 2 comparten 1 semáforo y tienen otro independiente a la otra ruta, consiguiendo así un total de 2 semáforos por ruta. Los nodos van circulando de manera aleatoria pero siempre respetando esta ruta. Estas rutas tienen una longitud de 3.2 km para la indicada en la figura 8 y 2.7 km la de la figura 7. Con estas distancias decidimos poner un tiempo de vida de los mensajes de 5 minutos. Los escenarios ejecutados para la obtención de resultados han sido un total de 6, donde se han configurado las rutas indicadas y se han realizado con un total de 6, 10, 20, 30, 40 y 50 nodos. Con diferente densidad de nodos simulamos diferentes estados de congestión del tráfico y podemos ver como va ganando efectividad el algoritmo de forwarding cuando aumentamos el número de nodos con los que interactuamos. Configuramos todos los nodos con una interfaz que simula un dispositivo WiFi con

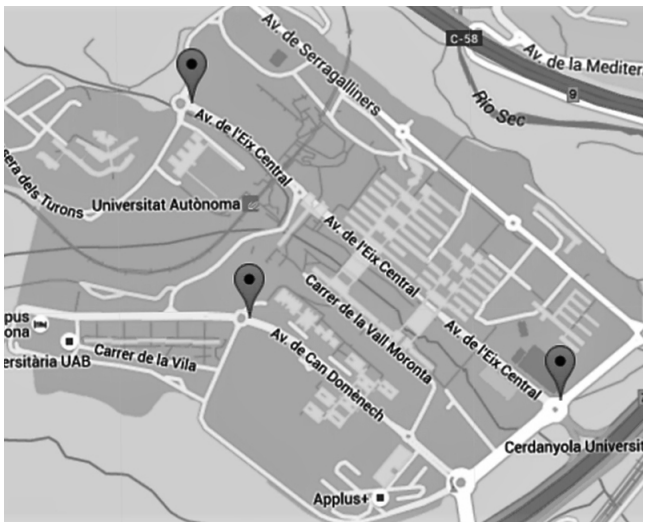

Fig. 9: Posicionamiento de los tres semáforos en el mapa de la Universidad Autónoma de Barcelona

Nodos	Pasadas por semaforo
6	271
10	357
20	530
30	584
40	598
50	611

Fig. 10: Numero de veces que los nodos pasan por alguno de los 3 semáforos

un rango de transmisión de 25 metros para los nodos que simulan a los coches y de 50 metros los que simulan los semáforos. Los semáforos realizan envíos de mensajes cada segundo y los vehículos tienen una velocidad que oscila entre los 5 km/h y los 50 km/h.

Tras la ejecución de los 6 escenarios obtenemos unos resultados que presentaremos en la siguiente sección.

5.3. Resultado de las simulaciones

En esta sección mostraremos los resultados obtenidos tras la ejecución de los 6 escenarios expuestos en la sección anterior. Con estos resultados podremos ver como la solución propuesta va ganando efectividad según el número de nodos aumenta con lo que aumentan el número de retransmisiones.

Los datos que medimos son dos, el número de coches que pasan por los semáforos durante las 4 horas de la simulación y el porcentaje de estos que llegan al nodo que representa el semáforo con la programación de este retransmitida por un nodo que no sea el propio semáforo. Todas estas medidas se presentarán en función al número de nodos que tenemos en nuestros dos grupos de vehículos, siempre estos igualados en número.

Empezamos viendo como en la figura 10 podemos observar como conforme aumentamos el número de nodos aumenta el número de veces que el semáforo se comunica con algún nodo. También podemos observar que si doblamos el número de nodos no significa que doblemos el número de conexiones que recibe el nodo que simula el semáforo. Esto es debido a que el semáforo tiene un limite de conexiones por segundo y si durante un instante este limite es superado por el número de nodos que están al alcance de la interfaz WiFi del semáforo habrán algunos nodos que saldrán del rango del semáforo sin haber realizado conexión alguna con este. Debido a la aleatoriedad de movimiento de los nodos nos ha sido imposible controlar estos casos. Aun sin controlar estos casos se han obtenido resultados ilusionantes.

nº Nodos	% de mensajes llegados
1	0%
6	63,80%
10	71,10%
20	81,10%
30	87,30%
40	88,40%
50	94,80%

Fig. 11: Proporción de vehículos que llegan a los semáforos con la programación recibida por otro vehículo

Continuamos con el análisis que mas nos interesa a nosotros, el número de nodos que llegan a los semáforos con la programación retransmitida por otro vehículo con anterioridad.

Si fijamos nuestra atención en la figura 11 podemos ver como aumentando el número de nodos aumenta el número de casos de éxito al llegar a los semáforos. Podemos ver también como no es proporcional el número de casos favorables con la cantidad de nodos. Se puede observar como con solo 6 nodos, mas de la mitad de veces que pasamos por un semáforo sabemos su estado con anterioridad. Conseguimos un gran número de casos de éxito con muy poco aumento del número de nodos tal y como se muestra en la figura 12.

Si analizamos el caso mas favorable, con 50 nodos, que simularía que 25 vehículos realizan la ruta 1 y 25 vehículos mas circulan por la ruta 2 obtenemos un 94,8 % de casos de éxito. Lo consideramos un número muy elevado de casos favorables donde solamente 31 vehículos de 611 no tendrían la programación antes de llegar al semáforo pero si que la tendrían a 50 metros de este cuando el propio semáforo se la transmita.

6 Conclusiones

En este documento se ha presentado una solución al problema de comunicación que presentan los coches autónomos, en concreto al paso de la programación de los semáforos entre estos y los vehículos.

Se ha empezado el articulo presentado al lector el problema que vemos dentro del campo de la conducción autónoma y se ha presentado una solución para solventar dicho problema. En concreto, la solución propuesta es la utilización de las redes DTN, específicamente se ha utilizado una

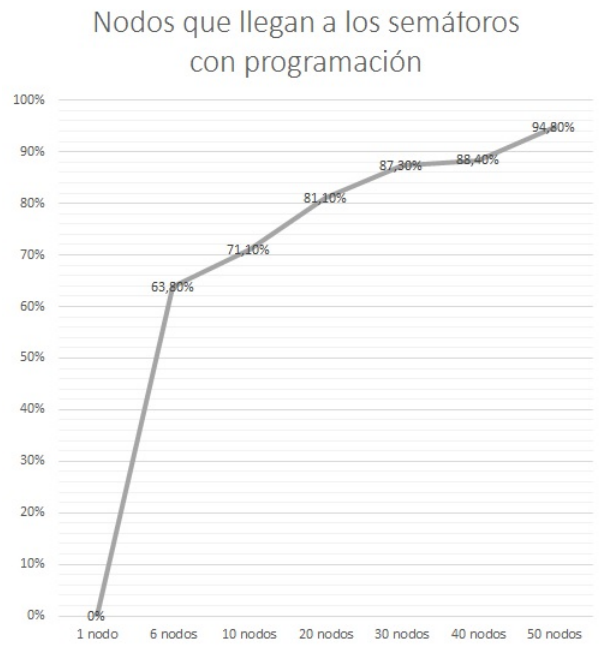

Fig. 12: Aumento de vehículos que saben la programación de los semáforos en función al número de nodos

implementación que nos permite establecer el algoritmo de enrutamiento de una manera mas flexible con la introducción de código en el propio mensaje. Para poder llevar a cabo esta solución nos proponemos una serie de objetivos que enumeramos en la primera sección. Seguidamente, se ha explicado de una manera breve los principales conceptos en redes DTN y como funciona aDTN. A continuación hemos expuesto la metodología y planificación seguida durante este proyecto, explicando las diferentes tareas que se han seguido. Hemos continuado con la explicación sobre el desarrollo del proyecto en si. Lo hemos dividido en una parte de diseño mas teórica y una parte de implementación que ha sido totalmente práctica. Para finalizar, se han presentado las simulaciones diseñadas y se han analizado los resultados obtenidos de estas simulaciones, de los cuales podemos decir que han sido exitosos.

6.1. Líneas de futuro

Este articulo es la visión general del trabajo final de grado realizado en un total de 300 horas, se han quedado muchas cosas por hacer que harían mas completo este trabajo. Las líneas de futuro en este proyecto pueden ser dirigidas a la seguridad del entorno, donde los semáforos podrían firmar los mensajes para evitar la suplantación. También se podría utilizar un cifrado homomórfico[5] para que los diferentes nodos no conozcan las rutas de los retransmisores de los mensajes. Otra propuesta sería la utilización de la probabilidad para tener en cuenta o no el mensaje que nos llega, esta probabilidad se vería afectada por el tiempo que ha pasado des de la emisión del mensaje.
La línea de futuro más inmediata sería la implantación del sistema en el proyecto de coche autónomo del Centre de Visió per Computadors[10].
Otra linea de futuro que discierne de este proyecto podría ser la utilización de esta tecnología para comunicar el estado del tráfico, el paso de vehículos de emergencia o accidentes en la calzada.

7 Agradecimientos

Los agradecimientos de este proyecto son principalmente para mi tutor de trabajo de final de grado Sergi Robles por sus consejos y correcciones. Me gustaría hacer especial mención a Carlos Borrego, ya que sin su ayuda no podría haber realizado las simulaciones con The One. También agradecer a los miembros del Centre de Visió per Computadors, en especial a David Vazquez por su impulso al inició del proyecto. Para acabar dar las gracias a la familia, amigos y compañeros por el soporte dado durante todo el proyecto.

Referencias

[1] Michael Solomon Desta, Ari Keränen, Teemu Kärkkäinen, Esa Hyytiä, Jörg Ott: Evaluating (Geo) Content Sharing with the ONE Simulator. Invited demo paper. Proceedings of the 14th ACM Symposium Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), November 2013.

[2] Michael S. Desta, Esa Hyytiä, Ari Keränen, Teemu Kärkkäinen, Jörg Ott, Evaluating (Geo) Content Sharing with the ONE Simulator (2013)

[3] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, Howard Weiss, RFC 4838, delay-tolerant networking architecture. RFC 4838 (Informational), 2007.

[4] Carlos Borrego, Sergi Robles, Angela Fabregues, Adrián Sánchez-Carmona, A mobile code bundle extension for application-defined routing in delay and disruption tolerant networking. Computer Networks 87 (2015) 59–77.

[5] PrivHab+: A secure geographic routing protocol for DTN A Sánchez-Carmona, S Robles, C Borrego Computer Communications 78, 56-73

[6] Delay and Disruption Tolerant Networks (DTNs), 23/07/2012, version 2.0. Forrest Warthman; [accessed 16-4-2016]. http://ipnsig.org/wp-content/uploads/2012/07/DTN_Tutorial_v2.05.pdf

[7] Security of Networks and Distributed Applications - SENDA(http://senda.uab.es/Active-DTN)

[8] Encoded Polyline Algorithm Format, 15/02/2016; [accessed 2016 April 16]. https://developers.google.com/maps/documentation/utilities/polylinealgorithm?hl=es#example

[9] The One Javadoc. [Accessed 21-05-2016]. https://www.netlab.tkk.fi/tutkimus/dtn/theone/javadoc_v141/

[10] Advanced Driver Assistance Systems research group. Computer Vision Center. [Accessed 26-06-2016] http://adas.cvc.uab.es/site/