PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

GEOPPS–N: ENRUTAMIENTO OPORTUNISTA PARA REDES VANET, EN UN ESCENARIO DE LOCALIZACIÓN GEOGRÁFICA DE LOS BUSES DE UN SISTEMA PTS

FRANCISCO JOSÉ VALDÉS VIDAL

Tesis para optar al grado de
Magíster en Ciencias de la Ingeniería

Profesor Supervisor:
MIGUEL RÍOS O.

Santiago de Chile, Enero, 2012

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

GEOPPS–N: ENRUTAMIENTO OPORTUNISTA PARA REDES VANET, EN UN ESCENARIO DE LOCALIZACIÓN GEOGRÁFICA DE LOS BUSES DE UN SISTEMA PTS

FRANCISCO JOSÉ VALDÉS VIDAL

Tesis presentada la Comisión integrada por los profesores:

MIGUEL RÍOS O.

VLADIMIR MARIANOV K.

HERNÁN SANTA MARÍA

REINALDO VALLEJOS C.

Para completar las exigencias del grado de
Magister en Ciencias de la Ingeniería

Santiago de Chile, Enero, 2012

A mis padres Julio y Marcia por su apoyo emocional e intelectual, y mis amigos por estar siempre ahí.

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi profesor Don Miguel Ríos, por guiar este proyecto con su constante ayuda y apoyo a lo largo de los meses de dedicación, fue un gusto trabajar con él.

A mi amigo Hernán Monsalve, por dedicarse a leer líneas y líneas de códigos conmigo para editar o encontrar errores de programación, también a mi ex – compañero de oficina Holger Kuprian, por guiarme en la edición de los escenarios de simulación de tráfico vehicular y de comunicación, al profesor de la Universidad de la Frontera Juan Ignacio Huircán por dedicarse a revisar las simulaciones y ayudarme a entender los problemas de una forma global.

Me gustaría agradecer a mis colegas y compañeros de oficina Roberto Pérez y Melisa Pérez por crear un ambiente de amistad y buena convivencia, compartiendo tasas de cafés y conversaciones, también les agradezco por tolerarme durante los días de trabajo en que nada me salía bien. Dentro de este grupo, también quisiera agradecer a Heinz Müller, mi amigo desde mi ingreso a la Universidad, quien estuvo todos los días fomentando mi motivación de trabajo, conversando acerca de temas particulares que lograban aclarar las dudas, y más que nada intercambiando opiniones.

A mis padres, por nunca rendirse conmigo y darme la libertad de realizarme como quisiera, aunque no siempre estuviéramos de acuerdo con la forma de hacerlo, también a mis hermanos Julio, Gonzalo y Andrés por acompañarme en este viaje.

Por último, y no menos importante, dar las gracias a la música por existir y crear un ambiente de relajo y concentración, durante todo el período de dedicación a este proyecto.

INDICE GENERAL

Pág.

INDICE DE TABLAS

INDICE DE FIGURAS

Pág.

RESUMEN

Los protocolos de enrutamiento basados en la topología de la red, han demostrado ser mucho más eficientes en las redes inalámbricas del tipo ad hoc de baja densidad, en comparación con los protocolos de otro tipo, tales como los de posición geográfica o de inundación. Este concepto se puede ocupar en las redes VANET (*Vehicular Ad hoc NETworks*) las que, debido a su naturaleza dinámica, consisten en grupos de nodos o vehículos desconectados unos de otros. En estas redes, en lugar de buscar el destino dentro del grupo en análisis, se ubica el mejor candidato para transportar el mensaje al destino. Esta tesis propone GeOpps-N, un nuevo protocolo de enrutamiento hibrido. GeOpps-N reune aspectos del enrutamiento reactivo basado en la topología de la red y del enrutamiento del tipo geográfico para redes DTN *(Delay Tolerant Networks)* y entrega resultados que superan a los obtenidos con protocolos tales como DYMO-UM, DYMO-FAU y GeOpps.

El escenario en el cual se evalúan los distintos protocolos, incluye la presencia de RSUs *(Road Side Units)* distribuidas a lo largo de un recorrido de buses de un sistema de transporte público (PTS), a las cuales cada bus del sistema de transporte debe reportar su posición a intervalos regulares. Para la evaluación de los protocolos de enrutamiento, se sensibilizan la cantidad de RSUs y densidad de buses dados por el tipo de horario. Los recorridos, los horarios de salida y la frecuencia de los buses son datos reales, obtenidos de la operación del sistema PTS, que se insertan al modelo.

Los resultados obtenidos por las simulaciones muestran una mejora en la razón promedio de entrega de paquetes hasta en un 159% por sobre los protocolos reactivos, tales como DYMO-FAU y DYMO-UM, y hasta en un 59% por sobre GeOpps. GeOpps-N también supera el retardo promedio de principio a fin en un 36% en comparación con GeOpps.

Palabras Claves: RSU, V2V, V2I, VANET, Ad hoc routing.

ABSTRACT

The network topology-based routing protocols has proven to be well suited in wireless ad hoc networks with low density, compared with protocols of other kind like geographic-based routing or flooding-based routing, this concept can be used in VANET (Vehicular Ad hoc NETworks) wish, due to its dynamic nature, consist in clusters of nodes or vehicles disconnected one from another. In these networks, instead of looking for the destiny inside the cluster in analysis, the best candidate to transport the message to the destiny is located. This thesis proposes GeOpps-N, a novel hibrid routing protocol. GeOpps-N brings together aspects from reactive network topology-based routing and geographic-based routing for DTN (Delay Tolerant Networks) and show results that outperform protocols like DYMO-UM, DYMO-FAU and GeOpps.

The scenario where the different protocols are evaluated, include the presence of RSUs (Road Side Units) distributed through the route of buses from a Public Transist System (PTS), to wich each bus has to report its position at regular intervals. To evaluate the routing protocols, the amount of RSU and density of buses are sensitized. The routes, time of departure and the frecuency of the buses are real data, obtained from the PTS operation, that are inserted in the model.

In the simulations, it improves the packet delivery ratio by up to 159% over the results of reactive protocols, such as DYMO-UM and DYMO-FAU and up to 59% over GeOpps. GeOpps-N also improves the mean end-to-end delay by up to 36% as compared to GeOpps.

Keywords: RSU, V2V, V2I, VANET, Ad hoc routing.

1 INTRODUCCIÓN

1.1 Motivación

Luego de más de 4 años de operación, el sistema de transporte público (PTS) de Santiago, conocido como Transantiago, todavía presenta algunos problemas para satisfacer la demanda de pasajeros en los distintos puntos de la ciudad. Estos problemas se deben tanto a la mala distribución de los buses disponibles en los diversos recorridos, debido a errores de diseño, como también a la variabilidad en los tiempos de espera de los usuarios de estos buses en los paraderos. Para mejorar la situación en este último aspecto, es necesario que los buses de un recorrido se mantengan, en lo posible, a una distancia de tiempo constante. Esta característica se puede lograr mediante el monitoreo en tiempo real de los buses, desde el Centro de Monitoreo de Buses (CMB), seguido de indicaciones a los buses sobre la disminución o aumento de su velocidad.

Para este propósito, los buses del Transantiago cuentan con varios enlaces de comunicación, como se observa en la Figura 1-1. Particularmente, los datos de la ubicación geográfica de los buses, son obtenidos desde un módulo GPS, y enviados a la CMB a través de dos sistemas de comunicación redundantes (un modem Mobitex y un módulo GSM/GPRS de respaldo).

El problema que presenta el depender de los datos de posición entregados por el sistema GPS, es que en aquellos lugares donde existen algunas obstrucciones (como edificios, túneles, árboles, etc.), el bus puede no tener conexión a los satélites GPS, o bien, los dos enlaces de comunicación con el CMB no funcionar adecuadamente, lo que hace que no se sepa con precisión la localización del bus. Este problema ocurre alrededor del 20% del tiempo de un recorrido.

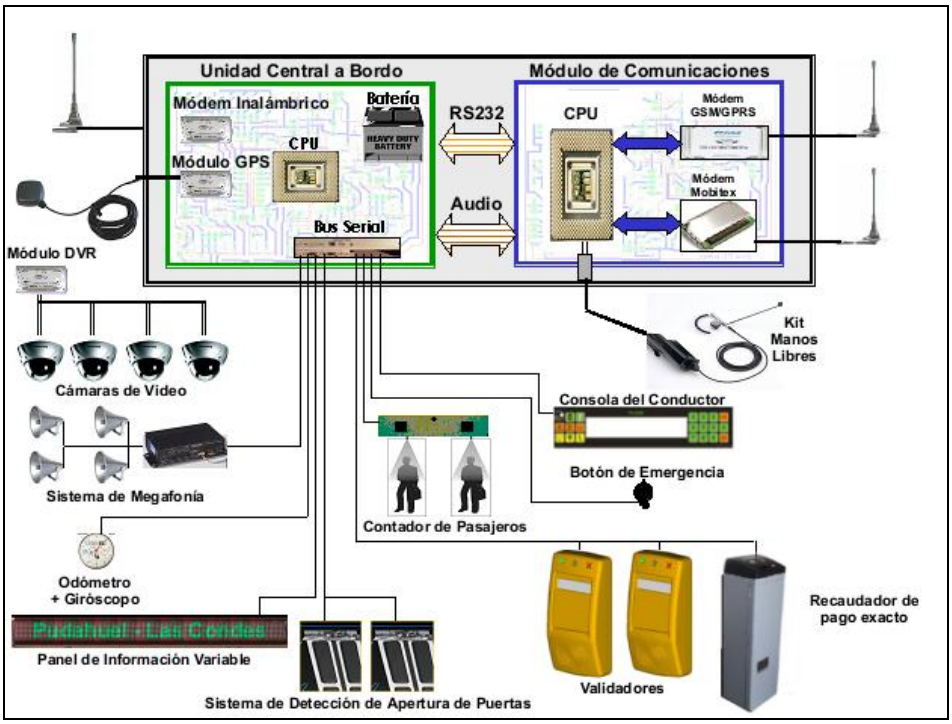

Figura 1-1: Plataforma tecnológica activa en los buses (Anexo G: Sistemas de Comunicaciones)

Para mejorar la confiabilidad y poder llenar estos vacíos de información del sistema actual, se propone complementarlo con el uso de redes VANETs (*Vehicular Ad hoc NETworks*), las cuales son redes de comunicación inalámbrica, que permiten las comunicaciones entre vehículos (V2V: *Vehicle to Vehicle*) y las comunicaciones entre vehículos y las instalaciones fijas (V2I: *Vehicle to Infrastructure*), conocidas como RSUs (*Road Side Units*).

1.2 **Antecedentes generales**

El desarrollo de los sistemas de transporte inteligente (ITS) ha conducido a mejoras significativas en los sistemas de transporte, posibilitando la interacción inteligente entre la infraestructura fija y los vehículos para el control eficiente y seguro del tráfico.

La naturaleza espontánea del tráfico vehicular y la movilidad en las aplicaciones

de ITS, han dado un impulso al desarrollo de las redes VANETs. Se ha estandarizado a DSRC (*Dedicated Short Range Communications*) como el espectro asignado para VANETs, el que permite la comunicación, en casos de movilidad, a alta velocidad (hasta 200 [km/h]) tanto para aplicaciones V2V como V2I, dentro de un rango de cobertura de hasta 1000 [m]. El espectro asignado se encuentra en la banda de 5.9 [GHz] en Estados Unidos (Figura 1-2) y de 5.8 [GHz] en Europa, Korea y Japón.

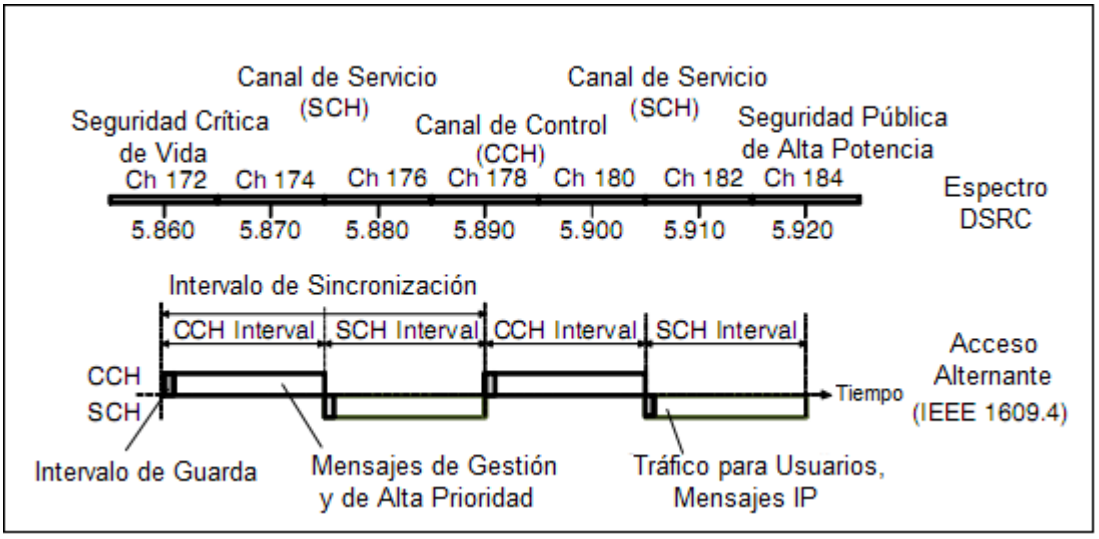

Figura 1-2: Espectro DSRC con canales alternantes de WAVE, según Chung et al. (2011)

Las aplicaciones para ITS, incluyen la seguridad y el control del tráfico vehicular, la gestión de flotas de transporte público y privado, la información para el viajero, etc.

La tendencia creciente en la disponibilidad geográfica de las redes IP (*Internet Protocol*), ha hecho que la necesidad de envío de contenido multimedia y la conectividad a internet sea también esencial en los ambientes vehiculares.

Satisfacer los requerimientos de comunicación para las aplicaciones en ITS, se ha convertido en un tema desafiante debido al alto dinamismo en el movimiento del

tráfico vehicular. En general estas aplicaciones ITS no pueden tolerar grandes retardos en el acceso, mientras también ellas requieren tener una alta confiabilidad.

En el año 2004, el grupo IEEE802.11 comenzó a trabajar en el sistema WAVE (*Wireless Access in Vehicular Environment*), un modo de operación que puede ser usado por los dispositivos IEEE802.11 WLAN (*Wireless Local Area Network*) que operan en la banda DSRC, y cuya finalidad es proveer compatibilidad, servicios inter-operacionales y conectividad para aplicaciones de comunicación inalámbrica dentro de vehículos. Las definiciones del estándar WAVE y las mejoras al respecto, se están desarrollando por los grupos de trabajo IEEE802.11p y IEEE1609, los cuales se distribuyen la tarea en las diversas capas del modelo OSI (*Open System Interconnect*) según la Figura 1-3.

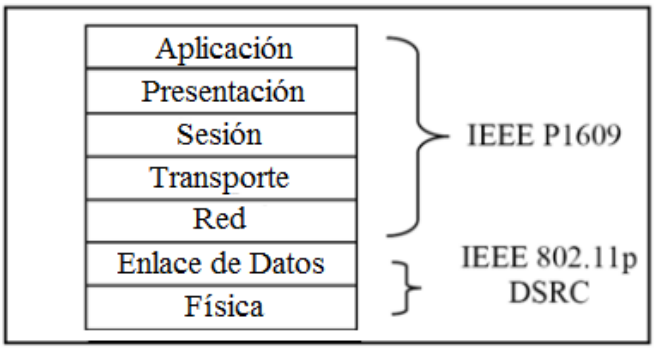

Figura 1-3: Grupos de trabajo de WAVE, separados por capas del modelo OSI, según Gukhool y Cherkaoui (2008).

Los parámetros a configurar en cada una de las capas van a depender de las características de la red en la cual se desea incorporar este servicio.

El enfoque de la tesis apunta a establecer un protocolo de enrutamiento óptimo para la capa de red del modelo OSI. Actualmente existe una amplia gama de protocolos creados para redes VANETs, los cuales se pueden categorizar según se muestra en la

Figura 1-4. Se observa que existen dos categorías principales: a) protocolos de enrutamiento basado en la topología y b) protocolos de enrutamiento geográfico. El enrutamiento basado en la topología, utiliza los enlaces activos de la red para generar un mapa local en cada nodo y luego enrutar los datos mediante el uso de tablas de enrutamiento. El enrutamiento geográfico, sólo utiliza información local acerca de la posición de los nodos vecinos para tomar las decisiones de reenvío de los paquetes. Debido al alto dinamismo de estas redes móviles, los protocolos basados en la topología sufren continuamente de fallas en los enlaces, establecidos en los mapas locales.

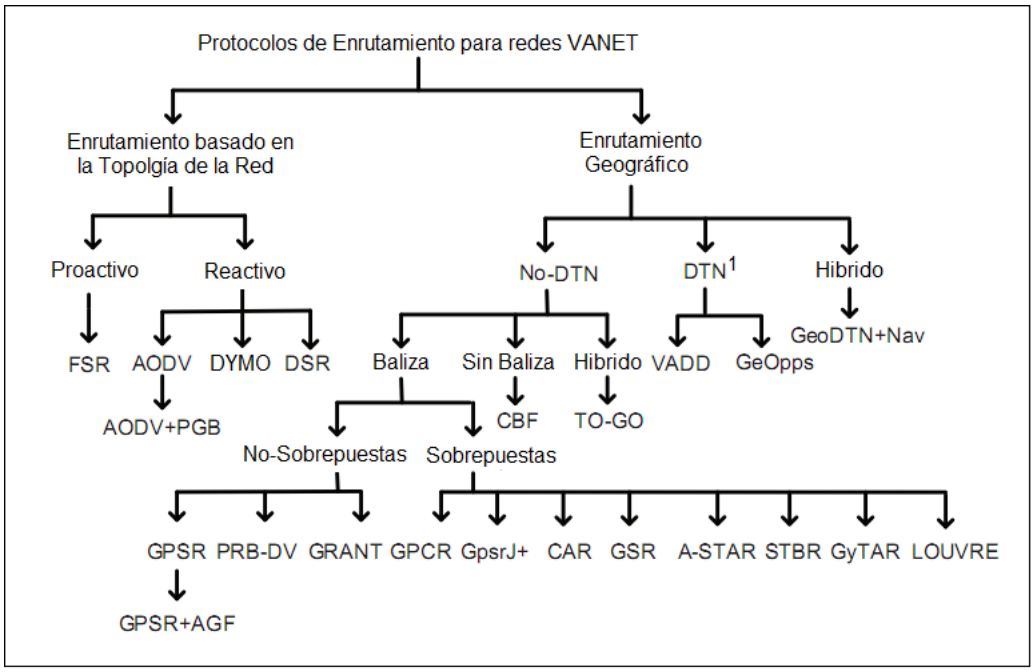

Figura 1-4: Protocolos de enrutamiento para VANETs, según Lee y Gerla (2010).

Los protocolos de enrutamiento geográfico, a su vez, se dividen en aquellos tolerantes al retardo (DTN: *Delay Tolerant Networks*) y aquellos no tolerantes al retardo (*None*-DTN). Las redes VANET tolerantes al retardo son aquellas que utilizan el sistema de transporte de información mediante la ayuda de otros vehículos que se mueven en la dirección del destino de la información. Este sistema es bastante eficiente para redes de

bajas densidades y en que, la mayor parte del tiempo, no hay conectividad directa entre la fuente y el destino, lo cual sucede con frecuencia en las redes vehiculares, debido a la formación de grupos separados de vehículos, generados en la cola de espera en los semáforos. Lamentablemente, el uso de este tipo de protocolos genera grandes retardos en el envío de datos, un ejemplo de este tipo de protocolo es GeOpps (*Geographical Opportunistic Routing for Vehicular Networks*) de Leontiadis y Mascolo (2007).

Los protocolos de enrutamiento más vistos en las publicaciones recientes son del tipo no tolerantes al retardo, de los cuales el más conocido es GPSR (*Greedy Perimeter Stateless Routing*) creado por Karp y Kung (2000), el que utiliza el principio básico de enviar los datos a aquel nodo que se encuentre más cerca del destino. GPSR tiende a caer en mínimos locales cuando se llega a un nodo i que no corresponde al nodo destino, y que se encuentra más cerca al destino que sus nodos vecinos a un salto de distancia, esto ocurre cuando existe un vacío entre el nodo i y el nodo destino. Para solucionar esto GPSR utiliza un modo perimetral en el que se rodea el vacío enviando la información hacia la derecha mediante un método llamado "regla de la mano derecha". También se encuentra en esta clase de protocolos el protocolo GSR, el cual en vez de utilizar la distancia euclidiana para elegir al siguiente nodo para enviar los datos, utiliza la distancia real a través de las calles, según un mapa que cada nodo conoce. Otros protocolos como A-STAR (*Anchor-based Street and Traffic Aware Routing*) de Seet et al. (2004), suponen que las calles por donde pasan una mayor cantidad de recorridos de buses, tienen una mayor conectividad para el envío de datos que aquellas con menor tráfico. También el protocolo GyTAR (*Greedy Traffic Aware Routing*) de Jerbi, Senouci, Meraihi y Ghamri-Doudane (2007), utiliza la información actualizada de la congestión entre intersecciones para enrutar los paquetes.

Las diferencia esenciales del funcionamiento de los diversos protocolos de enrutamiento se puede apreciar en la Figura 1-5.

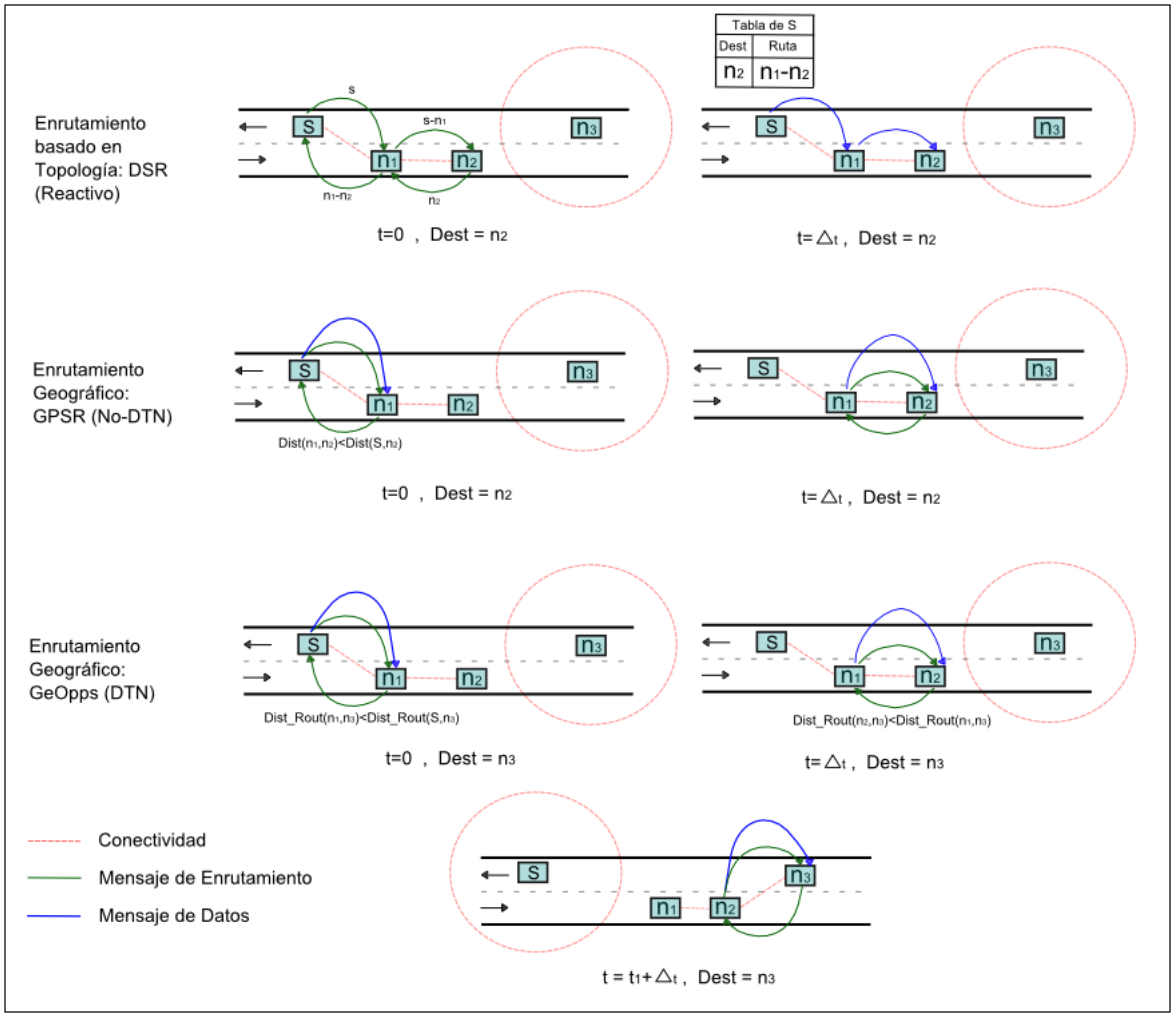

Figura 1-5: Funcionamiento de los protocolos de enrutamiento según categoría (Elaboración propia).

1.3 **Objetivos**

El objetivo general de esta tesis es desarrollar un protocolo de enrutamiento que cumpla lo mejor posible con los requerimientos de un sistema de transporte público, en el sentido de mejorar el desempeño en el monitoreo en tiempo real de la localización de los buses, suponiendo conocida la localización de las RSUs y el uso de redes VANET de comunicación inalámbrica V2V y V2I. Dentro de los objetivos específicos se encuentran: 1) recopilar información de la cobertura geográfica de los buses del

sistema PTS, para elegir la zona más adecuada para realizar las simulaciones; 2) generar el ambiente de simulación de tráfico vehicular en la zona elegida, utilizando el software SUMO 0.12.3; 3) diseñar un nuevo algoritmo de enrutamiento, basado en los existentes, que mejore la tasa de entrega de los paquetes, posiblemente sacrificando un retardo aceptable dentro de los requerimientos del proyecto; 4) aprender a utilizar el *software* de simulación de redes OMNET++ 4.2b para poder simular y evaluar el algoritmo diseñado; 5) comparar el protocolo diseñado con aquellos de mejor desempeño, que vienen en el *framework* INETMANET del programa y así analizar los resultados obtenidos.

1.4 **Aportes de la tesis**

Los aportes de la tesis se pueden resumir como siguen:

- Se crea un nuevo protocolo de enrutamiento, llamado GeOpps-N, que supera a otros protocolos como DYMO-UM, DYMO-FAU y GeOpps, en el escenario especificado de localización RSU-BUS para el PTS de Santiago, en el que los buses reportan su posición al CMB cada 30 segundos.
- Se logra generar un ambiente de simulación de tráfico de buses cercano a la realidad, utilizando datos estadísticos de los buses del transantiago y mapas reales de la ciudad.

1.5 **Organización de la tesis**

La organización de la tesis es como sigue: En el capítulo 2 se presentan los detalles del Protocolo de enrutamiento para un sistema de localización RSU-BUS. En el capítulo 3 se presenta la simulación realizada tanto para el modelo de tráfico vehicular como para el tráfico de datos, en los cuales se obtuvieron resultados comparativos para los distintos ambientes y protocolos. Finalmente, en el capítulo 4, se entregan las reflexiones y conclusiones del trabajo realizado.

2 PROTOCOLO DE ENRUTAMIENTO PARA UN SISTEMA DE LOCALIZACIÓN RSU-BUS

2.1 Características de las redes VANET

Para diseñar un protocolo de enrutamiento para redes VANET, hay que tener en consideración las características del flujo vehicular, en particular aquel de los buses de un sistema PTS. Los buses en este escenario PTS son los únicos vehículos que contienen los módulos para la comunicación inalámbrica.

Según Li y Wang (2007), las redesVANET se diferencian de las redes ad-hoc comunes (como WSN) en las siguientes características:

- Las redes de comunicación V2V y V2I se encuentran confinadas a las rutas vehiculares (recorridos).
- La topología de la red de comunicaciones es altamente dinámica, ya que los enlaces de comunicación V2V, entre vehículos sólo duran algunos segundos.
- La red, frecuentemente, se encuentra desconectada, debido al patrón de movimiento de los vehículos en grupos o *clusters.*
- Los dispositivos inalámbricos cuentan con suficiente capacidad energética y de almacenamiento ya que, a diferencia de las redes ad-hoc comunes, en el caso de las redes VANET los nodos son vehículos que cuentan con una fuente energética de alta disponibilidad (gracias a las baterías y el motor del vehículo).
- El modelamiento y predicción de los móviles es relativamente sencillo. Debido a las características del ambiente operativo de un sistema PTS, en que los vehículos se desplazan por calles, en las cuales se encuentran elementos como semáforos, sentidos de transito e información del tráfico, es fácil predecir la nueva posición de un vehículo basándose en la ubicación y velocidad de aquel.

- Existen dos tipos de ambientes de comunicación: a) ambiente de autopistas, en donde los vehículos se mueven en sólo dos sentidos y no hay obstáculos importantes para la comunicación inalámbrica, y b) ambiente de zona urbana, donde los vehículos se mueven por calles en diversos sentidos y donde existen diversos obstáculos para la comunicación inalámbrica, tales como edificios, túneles, árboles, etc.
- Las restricciones de retardo máximo de las comunicaciones son exigentes. Existen casos, por ejemplo, de comunicaciones de emergencia, en las que se requiere un retardo acotado en la comunicación. En estos casos importa, como medida de desempeño, el máximo retardo entre dos vehículos que se comunican, en lugar de considerar el retardo promedio.
- Hay interacciones con los sistemas de sensores a bordo de los vehículos. Por ejemplo, se utilizan GPS y mapas virtuales del estado del tráfico.

Estas características serán fundamentales en el diseño del algoritmo de enrutamiento a definir.

2.2 **Algoritmo de enrutamiento propuesto**

El protocolo de enrutamiento a implementar, debe depender completamente del escenario en que se encuentra el sistema de transporte público, el cual se supone que va a contar con RSUs distribuidas en la zona y en que cada bus va a contar con un recorrido propio a seguir, su posición (uso de GPS) y su velocidad instantánea. También las especificaciones para el monitoreo del sistema, requieren que cada bus entregue su posición a cada una de estas RSUs cada 30 segundos.

Debido a la tendencia de los vehículos de moverse en *Clusters*, muchas veces no va a existir conectividad con alguna RSU, como es el caso para n_1, si este debiera enviar un mensaje a la RSU, indicada en la Figura 2-1. El caso opuesto se da cuando la fuente de emisión es n_2, el cual se encuentra dentro del *Cluster* de la RSU. Luego, para

maximizar el tiempo en que los buses van a estar siendo monitoreados, es necesario maximizar la tasa de envío de paquetes desde los mismos a cada RSU. Una forma eficiente de hacer esto, es considerar los dos escenarios de conectividad mencionados para el diseño del protocolo. Para esto se hace necesario el uso de *Piggybacking* con los mensajes en los mismos buses, en base al protocolo, del tipo DTN, llamado GeOpps de Leontiadis y Mascolo (2007).

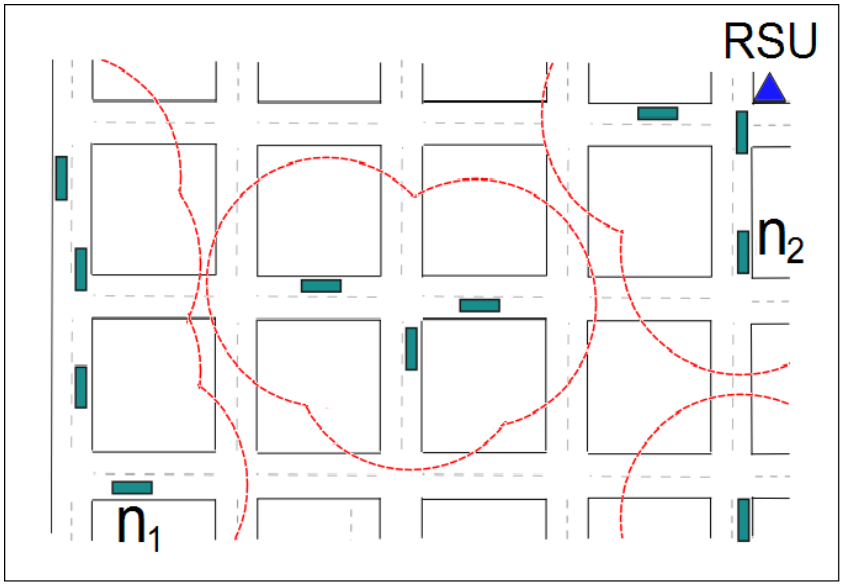

Figura 2-1: *Clusters* de conectividad en el tráfico vehicular (Elaboración propia).

GeOpps se basa en el cálculo del METD (*Minimum Estimated Time of Delivery)* en base a la ruta que sigue el vehículo que quiere enviar un paquete y la velocidad que lleva el mismo. Como los buses del PTS de Santiago tienen un recorrido definido, las rutas a seguir por los vehículos son determinísticas.

Geopps considera el tiempo en llegar al NP (*Nearest Point*: punto que geométricamente se encuentra más cerca del destino) y el tiempo estimado en ir del NP

al destino (este término supone que va a haber un vehículo que llevara el paquete desde NP al destino D) con la siguiente fórmula:

$$METD = ETA\ to\ NP + ETA\ from\ NP\ to\ D$$

En donde el ETA (*Estimated Time of Arrival*) se calcula con la velocidad instantánea del vehículo en el primer término y con una velocidad mucho menor en el segundo término.

Figura 2-2: Ejemplo del cálculo del punto más cercano NP del destino del paquete, según Leontiadis y Mascolo (2007).

En el caso de la Figura 2-2, en p_1 se requiere enviar un paquete a D, luego los vehículos "a" y "b" calculan sus respectivos METD, en donde el vehículo "b" tiene un menor valor de METD que "a". Luego, el vehículo "b" avanza hasta p_2 en donde se encuentra con "c" que tiene un menor valor de METD, debido a que su ruta pasa más cerca de D que la de "b". Finalmente, "c" transporta el paquete, esperando que en, NP_C o antes, encuentre a algún vehículo con menor METD, o bien esté dentro del área de cobertura de D.

El problema de este protocolo, es que tiende a caer en mínimos locales, al no considerar la conectividad entre el vehículo fuente y el destino, lo que genera grandes retardos debido al transporte. Por ejemplo, en la Figura 2-3, el vehículo "a" tiene que enviar un paquete a D. Luego, como el vehículo "b" tiene un mayor METD, debido a que NP_b se encuentra más lejos de D que NP_a, "a" decide transportar el paquete hasta D, incurriendo en un gran retardo.

Si, en lugar del procedimiento descrito, el vehículo "b" hubiera preguntado el valor del METD a "c" y este, a su vez, a su vecino y así sucesivamente, se hubiera encontrado el vehículo "e" con un valor del METD menor al que correspondía al nodo "a" en un principio. Esta es la esencia del protocolo que se propone en esta tesis.

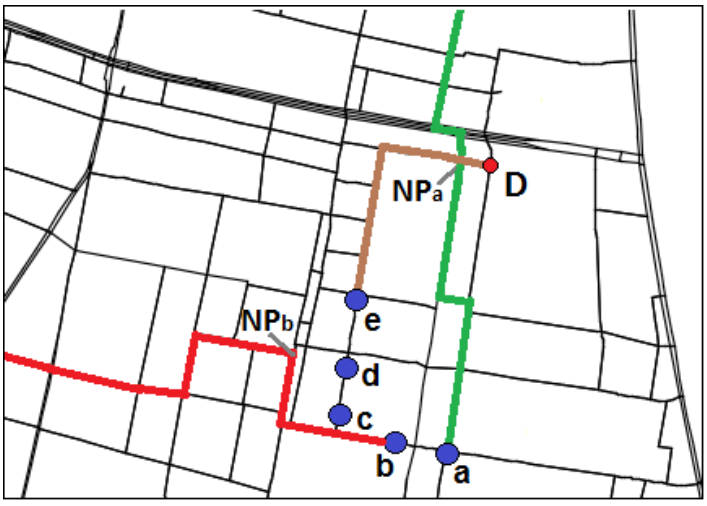

Figura 2-3: Mínimo Local para el protocolo GeOpps (Elaboración propia).

El nuevo protocolo se denomina GeOpps-N, en donde el nombre GeOpps es en honor al protocolo de Leontiadis y Mascolo (2007), en el que se basa el algoritmo, y el valor de N representa la cantidad de saltos mayor a 1 que requiere un vehículo fuente para encontrar otro vehículo con menor valor del METD.

Luego, específicamente, cuando un bus tiene que enviar un paquete a una RSU:

- Genera una llamada *broadcast* a todos los buses en su vecindad con un paquete RREQ (*Routing Request*) en el cual adjunta el valor de su propio METD y su dirección IP.
- Si algún bus vecino tiene un valor de METD menor al valor del bus fuente, genera un paquete RREP (*Routing Reply*) que se envía a través de la ruta almacenada en el mensaje original. Si ningún bus vecino tiene un METD menor al del bus fuente, cada bus vecino reenvía el paquete RREQ a todos los buses en su vecindad mediante *broadcast*, incluyendo en el paquete la ruta hasta su dirección IP.
- Una vez que el paquete RREP llega al bus fuente, este transmite los datos con su posición al bus que emitió el paquete RREP.

Una vez que se envían estos mensajes de ruteo, un bus esperará el paquete RREP por 3 segundos, antes de volver a buscar un bus con menor valor de METD. En casos en los que haya conectividad directa de un vehículo con la RSU, los vehículos más cercanos a esta tienen un valor menor del METD, por lo que naturalmente los paquetes tienden a dirigirse hacia las RSU. De esta forma, el algoritmo propuesto abarca todos los escenarios posibles para llegar al destino.

2.3 **Comparación de protocolos**

Los índices de desempeño de los protocolos de enrutamiento, generalmente consideran una terna de medidas del desempeño. Existen protocolos que pueden mejorar bastante el desempeño en un valor de la terna, pero tienden a empeorar el desempeño de otros valores. La terna considerada en este trabajo, está compuesta por las medidas de desempeño siguientes: a) tasa de entrega de paquetes (*Packet Delivery Ratio* o PDR); b) el retardo de principio a fin (*End to End Delay o E2E Delay*) y c) la sobrecarga de la red (*Overload*). Normalmente, los protocolos más deseables deberían maximizar el valor del PDR, minimizar el valor de *E2E Delay* y minimizar el valor de *Overload*.

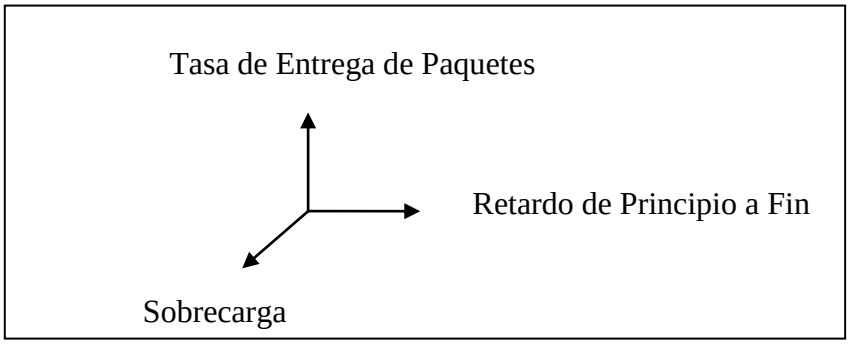

Figura 2-4. Terna de índices de desempeño.

Para comparar todos los tipos de protocolos de enrutamiento, habría que fijar escenarios comunes para todos ellos, variando la densidad de nodos de la red, la velocidad de los móviles, el tipo de movilidad, la cantidad de destinos, etc. Sin embargo, se puede hacer un análisis general en una red VANET con respecto a los 5 tipos mencionados en la sección de Antecedentes generales utilizando la terna de índices de desempeño indicada y basándose en las publicaciones del desarrollo de los mismos. En la Figura 2-5, se muestra la clasificación comparativa. El valor del *overload* de los protocolos basados en la topología es alto, debido a que utilizan tablas de enrutamiento que se obtienen a través de procesos de inundación en la red. El valor del *overload* del protocolo reactivo es menor al valor del protocolo de tipo proactivo, ya que sólo solicita una tabla de enrutamiento cuando la requiere. Los protocolos del tipo geográfico tienen valores mayores del *E2E Delay* que los protocolos basados en topología ya que, al usar sólo la información local, no aseguran un envío de datos exitoso y a tiempo, como en los otros tipos de algoritmos. El mayor de estos retardos se observa en los protocolos del tipo DTN, debido al uso de transporte por *piggybacking*. Por último, los protocolos con los mayores valores de PDR son aquellos del tipo DTN, ya que al transportar los datos, los mantiene mayor tiempo dentro de la red, haciendo que tengan mayores oportunidades de llegar a su destino. En cambio los otros protocolos, como los basados en topología o No-DTN, descartan los paquetes que no encuentran su destino, ya que cuentan con un temporizador TTL (*Time To Live*).

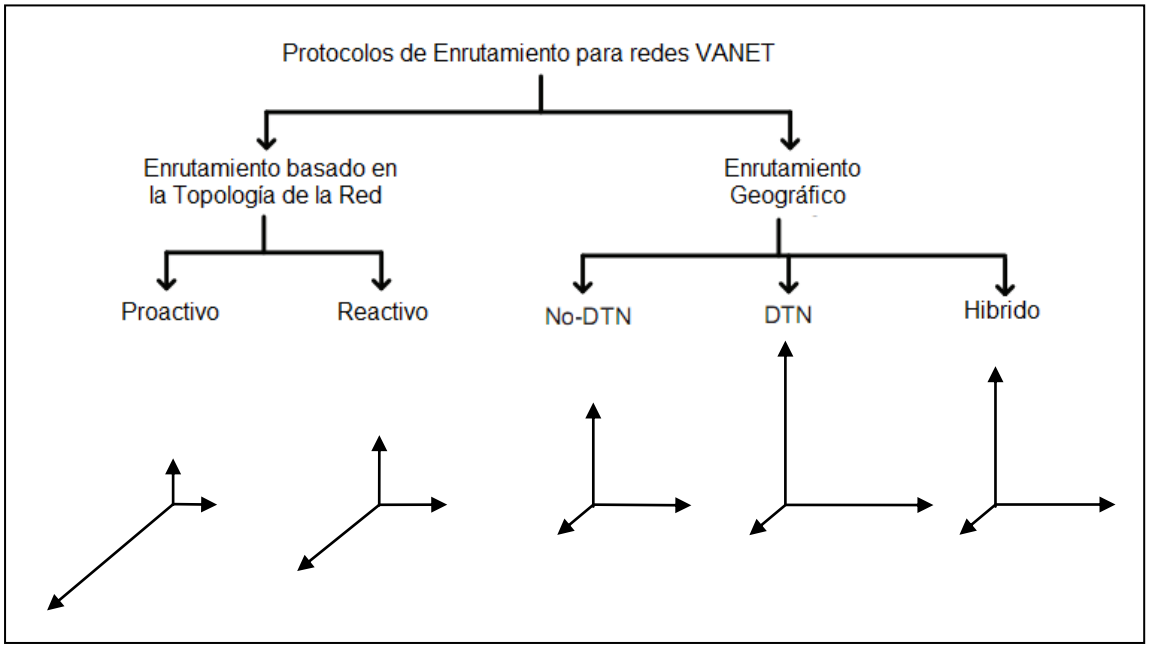

Figura 2-5. Desempeño de los tipos de protocolos de enrutamiento (Elaboración propia).

La idea central del nuevo protocolo GeOpps-N, es combinar el alto PDR de los protocolos DTN, tales como GeOpps y el bajo retardo de los protocolos reactivos, tales como DYMO. De este modo, se crea un protocolo de enrutamiento con retardos aceptables, dentro de las restricciones del sistema PTS de Santiago, y con una alta tasa de entrega de datos.

2.4 **Probabilidad de entrega de paquetes de GeOpps-N**

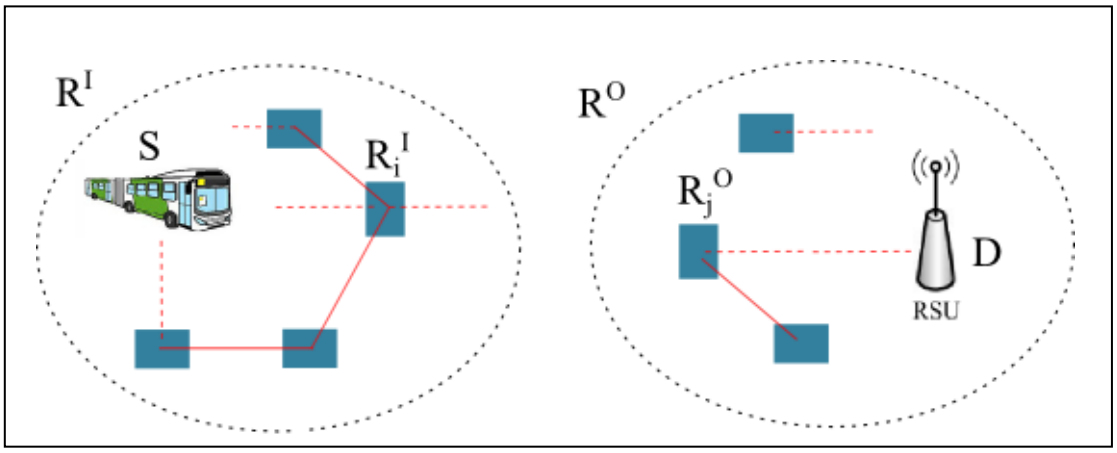

Figura 2-6. Esquema general de un protocolo de enrutamiento (Elaboración propia).

Considerando una red VANET, como se observa en la figura 2-6, existe un nodo origen, (nodo S), un nodo destino (nodo D) y un conjunto R de nodos que pueden retransmitir un mensaje. Cuando el nodo origen selecciona nodos para retransmitir el mensaje hacia el nodo destino, asumimos que existen M nodos con conectividad directa con el nodo origen, los cuales son agrupados en el conjunto $R^I = \{R_1^I, R_2^I, ..., R_M^I\}$, de la misma forma, se agrupan otros N nodos que no están dentro del rango de comunicaciones del nodo origen, en el conjunto $R^O = \{R_1^O, R_2^O, ..., R_N^O\}$. Así se tiene que: $R = R^I \cup R^O$ y $R^I \cap R^O = \varnothing$.

Siguiendo el modelo de Li et al. (2011), la ocurrencia de que dos nodos se encuentren dentro de sus respectivos rangos de comunicación, sigue una distribución Poisson. Así se define λ_{ij} como la tasa de encuentro entre dos nodos R_i y R_j, con $i \neq j$. Se define de igual forma a T como el tiempo de vida de un mensaje, y la probabilidad de entrega de un mensaje como la probabilidad de que un mensaje pueda ser entregado al destino D dentro del tiempo de vida T.

En primer lugar se consideran los nodos dentro del grupo R^I. Cuando un nodo dentro de este grupo es elegido para retransmitir un mensaje (a través de uno o más saltos), este recibe el mensaje del nodo origen sin un retardo considerable, y luego el paquete debe ser enviado al nodo destino D. Como el tiempo entre eventos de una distribución Poisson sigue una distribución exponencial, la probabilidad de que un nodo R_i^I encuentre al nodo D, antes de un período de tiempo T, se puede expresar como:

$$p_i = 1 - e^{-\lambda_{i,D}^I \cdot T} \tag{2.1}$$

Donde $\lambda_{i,D}^I$ es la tasa de encuentro entre el nodo R_i^I y el nodo D.

Si, por otro lado, el nodo origen S elige un nodo $R_j^O \in R^O$ para retransmitir, el nodo R_j^O debe entrar dentro del rango de comunicaciones del nodo origen S y luego contactar al nodo destino D. Para encontrar la probabilidad de entrega para este caso, se usa el resultado de Li et al. (2011), quienes encuentran la función de densidad de probabilidad para la entrega exitosa de un paquete, desde el nodo S, a través de un nodo $R_j^O \in R^O$, hacia un nodo D, en un tiempo y, como:

$$f_j(y) = \frac{\lambda_{j,D}^O \lambda_{j,S}^O (e^{-\lambda_{j,S}^O y} - e^{-\lambda_{j,D}^O y})}{\lambda_{j,D}^O - \lambda_{j,S}^O} \tag{2.2}$$

Dónde $\lambda_{j,S}^O$ es la tasa de contacto entre R_j^O y S, y $\lambda_{j,D}^O$ es la tasa de contacto entre R_j^O y D.

Se denota $q_j = \Pr ob(y \le T)$ como la probabilidad de que, un mensaje emitido por un nodo S, pueda ser transmitido a un nodo destino D antes de un tiempo de medición T, mediante la retransmisión por medio de un nodo R_j^O, y se calcula como:

$$q_j = \int_0^T f_i(y)dy = 1 - \int_T^\infty f_i(y)dy = 1 - \frac{\lambda_{j,D}^O \lambda_{j,S}^O}{\lambda_{j,D}^O - \lambda_{j,S}^O}\left(\int_T^\infty e^{-\lambda_{j,S}^O y}dy - \int_T^\infty e^{-\lambda_{j,D}^O y}dy\right)$$

$$q_j = 1 - \frac{\lambda_{j,D}^O e^{-\lambda_{j,S}^O T} - \lambda_{j,S}^O e^{-\lambda_{j,D}^O T}}{\lambda_{j,D}^O - \lambda_{j,S}^O} \qquad (2.3)$$

Muchos nodos dentro de R pueden estar involucrados en una comunicación entre un nodo origen y un nodo destino. Así se define la variable $x_i \in \{0,1\}$ que toma el valor 1 si es que el nodo R_i^I es usado para retransmitir un mensaje y toma el valor 0 en otro caso. Se define también la variable $y_j \in \{0,1\}$, la cual toma el valor 1 si es que el nodo R_j^O es elegido para retransmitir el mensaje y el valor 0 en otro caso.

Luego, se obtiene la siguiente probabilidad de transmisión exitosa para cualquier protocolo:

$$P = 1 - \prod_{i=1}^{M}(1-p_i)^{x_i}\prod_{j=1}^{N}(1-q_j)^{y_j} \qquad (2.4)$$

El objetivo de cada protocolo de enrutamiento es seleccionar los nodos apropiados para retransmitir dentro del conjunto R, en un tiempo de vida T. Protocolos del tipo No-DTN (basados en la topología de la red) no utilizan nodos fuera del rango de comunicación del nodo origen S, en consecuencia, para este tipo de protocolos $y_j = 0$. Así la probabilidad de transmisión exitosa de un mensaje para protocolos de enrutamiento del tipo No-DTN es:

$$P_{No-DTN} = 1 - \prod_{i=1}^{M} (1 - p_i)^{x_i} \prod_{j=1}^{N} (1 - q_j)^0 = 1 - \prod_{i=1}^{M} (1 - p_i)^{x_i} \quad (2.5)$$

Protocolos del tipo DTN, como GeOpps, sólo considera los nodos vecinos a un salto para elegir cuál retransmitirá el mensaje. Por lo tanto, existe un conjunto de nodos dentro de R^I que se encuentran a dos saltos o mas separados del nodo origen S que no son considerados para enrutar los datos. Supongamos que existen $K \leq M - 1$ nodos que pertenecen a R^I y no son considerados para las decisiones de enrutamiento.

Luego, la probabilidad de entrega exitosa de un paquete para este tipo de protocolos es:

$$P_{DTN} = 1 - \prod_{i=1}^{M-K} (1 - p_i)^{x_i} \prod_{j=1}^{N} (1 - q_j)^{y_j} \quad (2.6)$$

La probabilidad de entrega del protocolo propuesto GeOpps-N es P, ya que utiliza nodos para retransmitir dentro y fuera del rango de cobertura del nodo origen S.

Es fácil ver que el protocolo propuesto GeOpps-N supera a los protocolos DTN y No-DTN ya que:

$$P \geq P_{No-DTN} \quad \wedge \quad P \geq P_{DTN}$$

Existen dos aproximaciones en el análisis realizado. En primer lugar, la probabilidad de entrega exitosa se estimó sólo para un paquete. Cuando exiten más de un paquete en la transmisión, podrían existir colisiones, las cuales incrementan el tiempo de transmisión. En segundo lugar, cuando dos o más nodos en el conjunto R^I son usados para retransmitir, la expresión para q_j cambia. Sin embargo, el hecho de que GeOpps-N tiene una mejor probabilidad de entrega se mantiene.

3 EXPERIENCIA COMPUTACIONAL

3.1 Introducción

Para evaluar el desempeño del algoritmo de enrutamiento GeOpps-N, es necesario crear un modelo que permita simular dicho algoritmo y compararlo con otros. En este trabajo, se requiere crear un escenario de tráfico vehicular (se usará como caso en estudio el PTS de Santiago) e incorporar el algoritmo de enrutamiento en estudio al módulo de comunicación inalámbrica, presente en los buses del Transantiago, y medir su desempeño. Crear un modelo que abarque toda la ciudad de Santiago es demasiado complejo, debido a la gran cantidad de vehículos en la red, lo que requeriría demasiada capacidad de cálculo para efectuar la comunicación inalámbrica, así como la compleja adaptación de la realidad de tráfico a una zona tan extensa. Por estas razones se decidió restringir el análisis, seleccionando la zona del Transantiago que tenga la peor cobertura (menor densidad) de comunicaciones por parte de los buses.

Las siguientes secciones muestran, en primer lugar, una evaluación real (usando los datos de operación del Transantiago) de la cobertura de los buses por zonas. Luego se presenta la creación del modelo de tráfico vehicular para esta zona, seguido de la evaluación del algoritmo de enrutamiento planteado en la sección anterior y la comparación con otros algoritmos conocidos y, por último, se realiza un análisis de estos resultados.

3.2 Simulación del tráfico vehicular

Para simular el movimiento de los buses, es necesario seleccionar una zona del Transantiago. En la selección de esta zona hay que considerar la cantidad de buses que se encuentran en ella en cada intervalo de tiempo, estos intervalos se separan de acuerdo a la Tabla 1.

Tipo Periodos	Horario Periodos
01 - Pre Nocturno	00:00 - 01:00
02 - Nocturno	01:00 - 05:30
03 - Transición Nocturno	05:30 - 06:30
04 - Punta Mañana	06:30 - 08:30
05 - Transición Punta Mañana	08:30 - 09:30
06 - Fuera de Punta Mañana	09:30 - 12:30
07 - Punta Mediodía	12:30 - 14:00
08 - Fuera de Punta Tarde	14:00 - 17:30
09 - Punta Tarde	17:30 - 20:30
10 - Transición Punta Tarde	20:30 - 21:30
11 - Fuera de Punta Nocturno	21:30 - 23:00
12 - Pre Nocturno	23:30 - 00:00

Tabla 1: Horas del día, separadas por intervalos de demanda de buses.

Los datos operativos del Transantiago no incluyen la posición de cada uno de los buses en cada instante de tiempo, sino que sólo incluyen la hora de salida y llegada de cada bus. Por ello no es sencillo determinar la zona en que se encuentran los buses de los recorridos troncales en cada momento, ya que estos recorridos circulan por varias zonas. Por esta razón, se supuso que un bus troncal, que recorre una cierta cantidad de zonas en un periodo dado, corresponde a un bus troncal que se encuentra en cada una de esas zonas en todo ese período. Es claro que esto significa que se está sobreestimando la cantidad total de buses por cada zona. En cuanto a los buses alimentadores, se asumió que todos ellos se encuentran en su zona correspondiente en todo instante, lo cual se da con la mayoría de los recorridos. Como base de análisis, se consideraron los datos de un día de semana (correspondientes al día Lunes 23 de Marzo del 2009) y se estimó la cantidad de buses por zona, obteniendo los resultados mostrados en la Figura 3-1. Si bien estos datos sobreestiman el número de buses, debido a la suposición mencionada anteriormente sobre los buses troncales, sin embargo dan una buena referencia para comparar las distintas zonas.

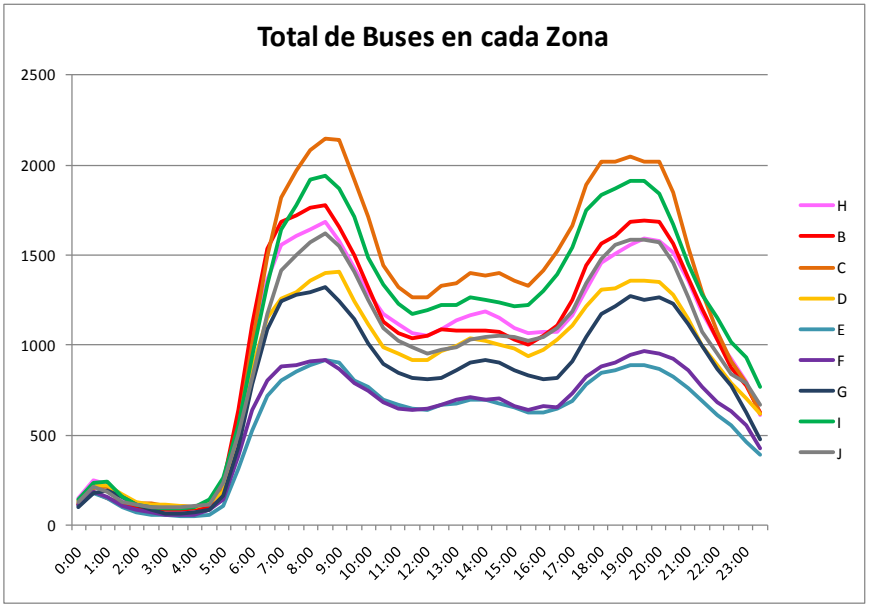

Figura 3-1: Total de buses (troncales y alimentadores), separados por horario y zona (Subsecretaría de Transporte).

La métrica a considerar, para elegir una zona para la simulación, corresponde a la conectividad de los nodos (Buses) de la red (Calles), la cual se puede obtener a través del cálculo de la densidad de buses en cada zona. Para ello, se necesita calcular la superficie urbana de cada zona del PTS de Santiago, lo cual se realizó tomando como base las calles que recorren los buses del Transantiago y despreciando las zonas no urbanas, como cerros y campos en la periferia de la ciudad, tal como se observa en la Figura 3-2. Los datos relevantes se muestran en la Tabla 2.

Con los datos de la cantidad de buses y la superficie por zonas, es posible calcular la densidad de buses por zona, la que se muestra en la Figura 3-3. En la figura se observa que las zonas con peor conectividad son la G y F. Tomando como referencia que la peor situación corresponde a la zona valle o nocturna de estas curvas, se deduce que la mejor elección de zona corresponde a la Zona G.

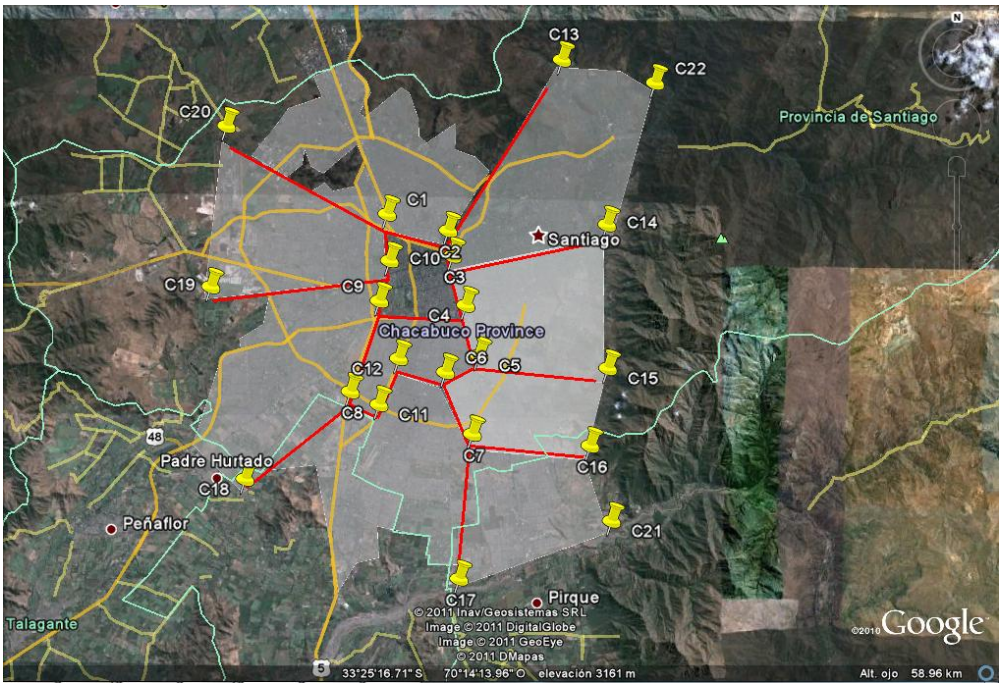

Figura 3-2: Zonas urbanas del Transantiago (Google Earth).

		Sup Real(Km^2)	Sup Urb(Km^2)
B	Independencia	7,4	96
	Huechuraba	44,8	
	Quilicura	57,5	
	Recoleta	16,2	
	Conchalí	10,7	
	Renca	24,2	
	TOTAL Zona B:	160,8	
C	Lo Barnechea	1023,7	120
	Providencia	14,4	
	Las Condes	99,4	
	Vitacura	28,3	
	TOTAL Zona C:	1165,8	
D	Peñalolén	54,2	80,8
	La Reina	23,4	
	Macul	12,9	
	Ñuñoa	16,9	
	TOTAL Zona D:	107,4	
E	La Florida	70,8	51,2
	La Granja	10,1	
	TOTAL Zona E:	80,9	
F	Puente Alto	88,2	65,8
	TOTAL Zona F:	88,2	
G	San Bernardo	155,1	104
	La Cisterna	10	
	San Ramón	6,5	
	La Pintana	30,6	
	El Bosque	14,1	
	TOTAL Zona G:	216,3	
I	Estación Central	14,1	106
	Cerrillos	21	
	Maipú	133	
	TOTAL Zona I:	168,1	
J	Quinta Normal	12,4	83,6
	Cerro Navia	11,1	
	Pudahuel	197,4	
	Lo Prado	6,7	
	TOTAL Zona J:	227,6	
H	Pedro Aguirre Ce	9,7	34
	San Joaquín	9,7	
	San Miguel	9,5	
	Lo Espejo	7,2	
	TOTAL Zona H:	36,1	

Tabla 2: Comunas, superficie total y superficie urbana de cada zona de Santiago

(Instituto Nacional de Estadísticas y Google Earth).

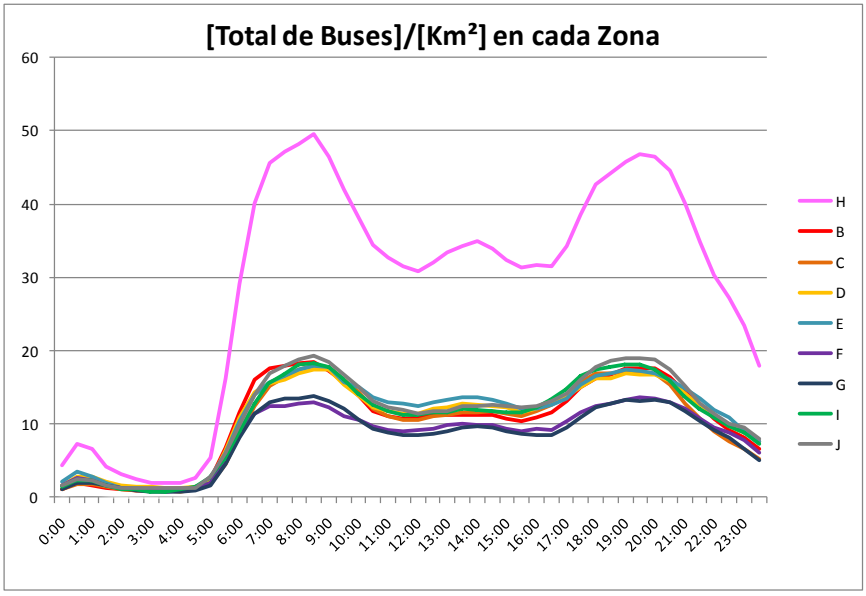

Figura 3-3: Densidad de buses en cada zona urbana (Elaboración propia).

Para crear un modelo de comunicación inalámbrica que funcione con vehículos en movimiento, como lo hacen las redes VANETs, es necesario utilizar un modelo para el desplazamiento físico de los vehículos y otro para el intercambio de paquetes de datos entre ellos y las RSUs. En el primer caso se utilizó el simulador SUMO (Behrisch, Bieker, Erdmann, y Krajzewicz, 2011) y para el segundo caso OMNET++ (Varga, 2010).

Para realizar una simulación en SUMO (Behrisch et al. 2011), se necesitan dos archivos: uno que contenga la descripción de la red (*.net.xml) y uno con la descripción de las rutas (*.rou.xml), junto con las características que cada vehículo usa en tales rutas, tales como el tiempo de salida, la aceleración, la desaceleración y la velocidad máxima. El proceso de creación de la simulación de movimiento de vehículos con SUMO (Behrisch et al. 2011) se observa en la Figura 3-4.

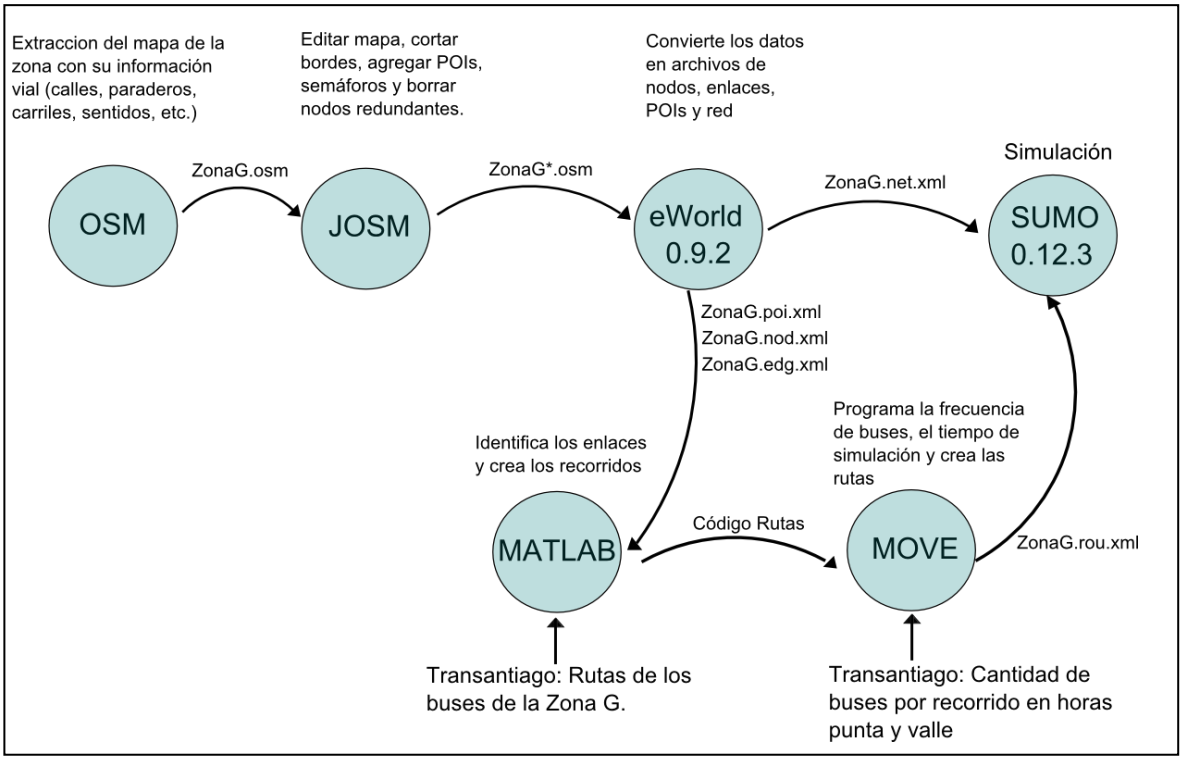

Figura 3-4: Edición y creación de una simulación de tráfico vehicular en SUMO

(Elaboración propia).

En primer lugar, se extrae toda la información disponible acerca de la vialidad y características de una zona geográfica, en particular la disponible en OSM (© Colaboradores de OpenStreetMap, CC-BY-SA). Ese proyecto, al igual que Wikipedia, es una base de datos abierta y editable por el público. Una parte de la Zona G se observa en la Figura 3-5. La información se extrae y se edita usando JOSM (Edgewall Software), eliminando las calles que no corresponden a la Zona G y agregando POIs (*Point Of Interest*) como paraderos e intersecciones relevantes. También se agrega la ubicación de los semáforos y, por último, se eliminan las calles por donde no pasan recorridos de los buses de esta zona. Una vez finalizado este proceso, el mapa queda como el mostrado en la Figura 3-6.

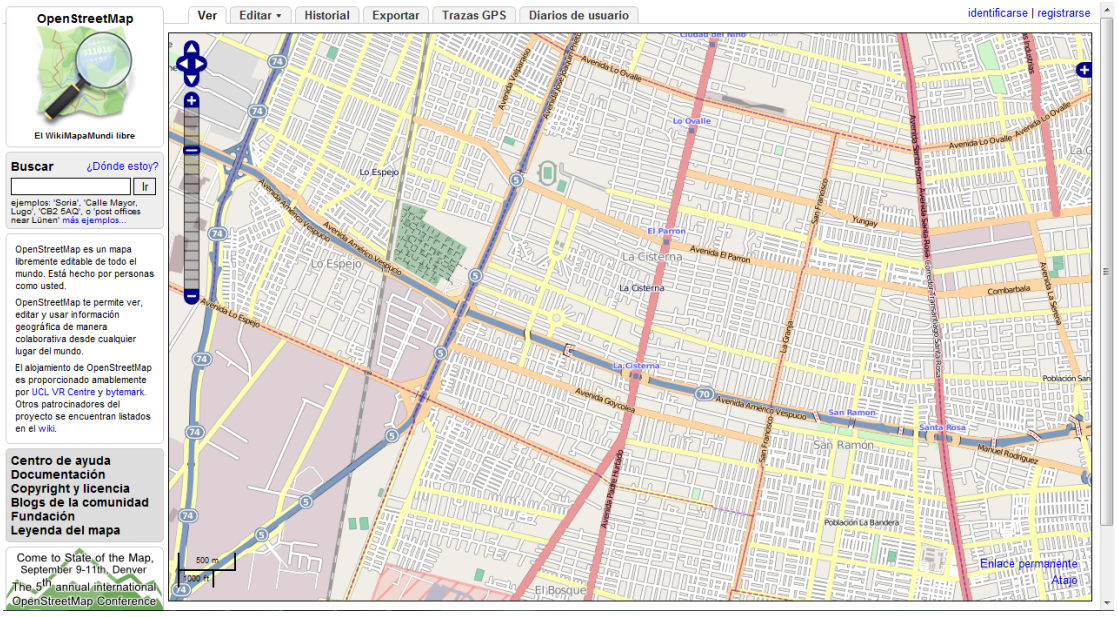

Figura 3-5: Parte de la Zona G en OSM (© Colaboradores de OpenStreetMap, CC-BY-SA).

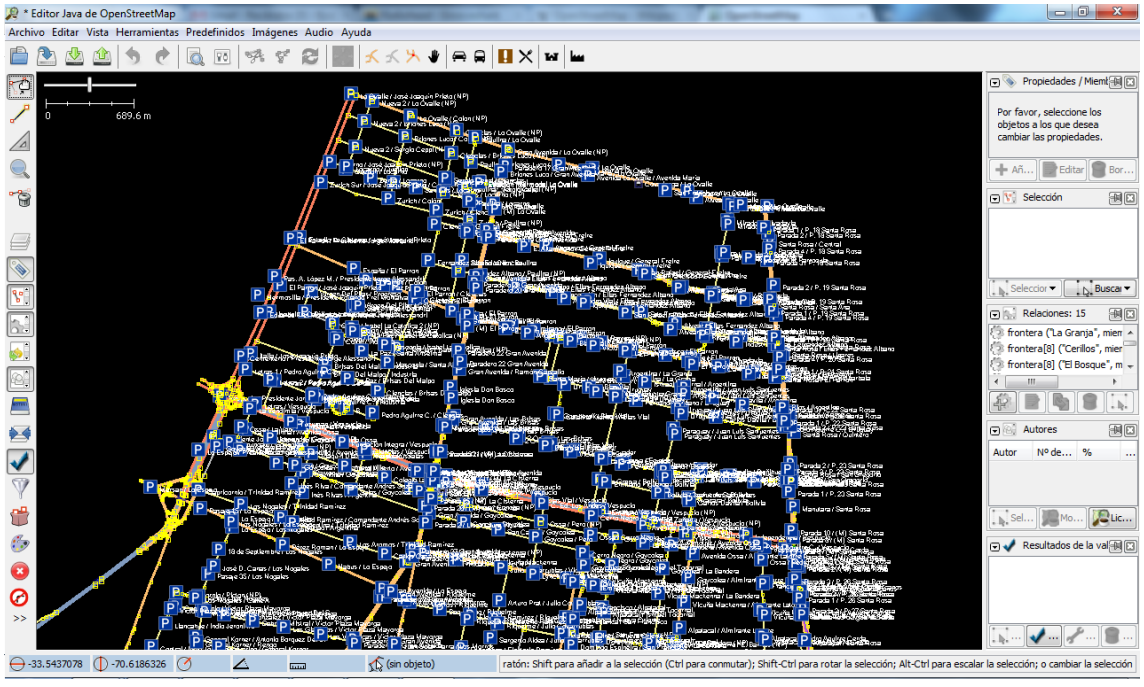

Figura 3-6: Parte de la Zona G en JOSM (Edgewall Software).

Una vez concluida la edición del mapa de recorridos, se exporta el archivo ZonaG.osm al programa eWorld 0.9.2 (Meinel y Schünemann, 2011), el cual toma toda la red y la convierte en archivos de nodos, enlaces, POIs y red. En este punto, se encuentra lista la red de recorridos donde se va a simular. Sin embargo, para definir las rutas, se necesita conocer el nombre de cada enlace. Lamentablemente, el software eWorld 0.9.2 (Meinel y Schünemann, 2011) no asigna el nombre de los enlaces como se asigna en JOSM (Edgewall Software) con el nombre de las calles, sino que utiliza códigos de 36 números en hexadecimal. Para conocer los nombres de los enlaces, por los que se mueve cada bus de esta zona, fue necesario entonces crear un código en MATLAB que leyera los archivos de nodos, enlaces y POIs. Como estos últimos contienen el nombre real de las intersecciones (por ejemplo Santa María/ Américo Vespucio), asignadas al código de un nodo, es posible interpretar el archivo de enlaces, los cuales se encuentran en el formato: Edge: (Código), From Node: (Código), To Node: (Código). El código en MATLAB se muestra en el Anexo A.

Una vez identificados los enlaces del recorrido, se calcula la ruta de enlaces en código para cada recorrido de ida y regreso en la Zona G. Una parte del código y el gráfico de uno de los recorridos creados en MATLAB se observa en la Figura 3-7.

Luego de tener todas las rutas calculadas, estas se ingresan a un programa generador de movilidad, llamado MOVE (*MObility model generator for VEhicular networks*) creado por Karnadi, Mo y Lan (2007). Para este fin se crean los vehículos Bus_chico y Bus_grande que corresponden a buses Alimentadores y Troncales respectivamente, los cuales tienen una aceleración de $1,2[m/s^2]$, una desaceleración de $1,5[m/s^2]$, una velocidad promedio de $8,33[m/s]$ y una longitud de $12[m]$ para el caso del bus troncal y $18[m]$ para el caso del bus alimentador, según información recopilada del Transantiago. Luego se ingresan los códigos de las rutas de ida y regreso, calculadas en MATLAB y, por último, la cantidad de buses utilizados en cada ruta y el tiempo de espera entre la salida de cada uno de ellos. Esto se observa en la Figura 3-8.

Por último, se ingresan las rutas y la red al programa de simulación SUMO (Behrisch et al. 2011) y se obtiene la simulación del movimiento de los buses, como se observa en la Figura 3-9.

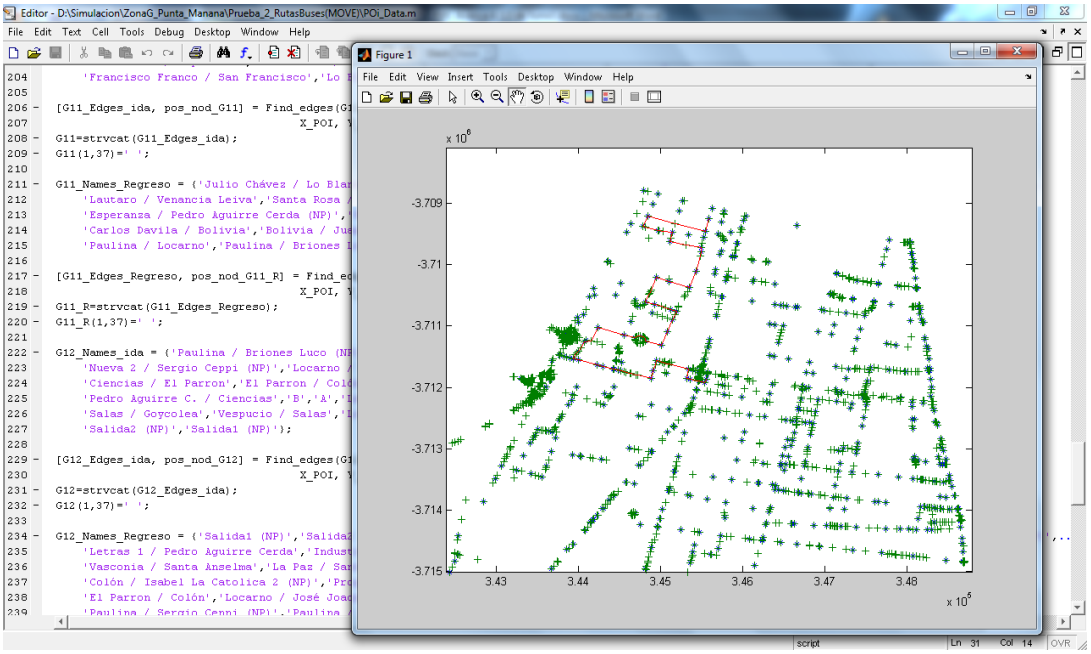

Figura 3-7: Recorrido G12 en MATLAB.

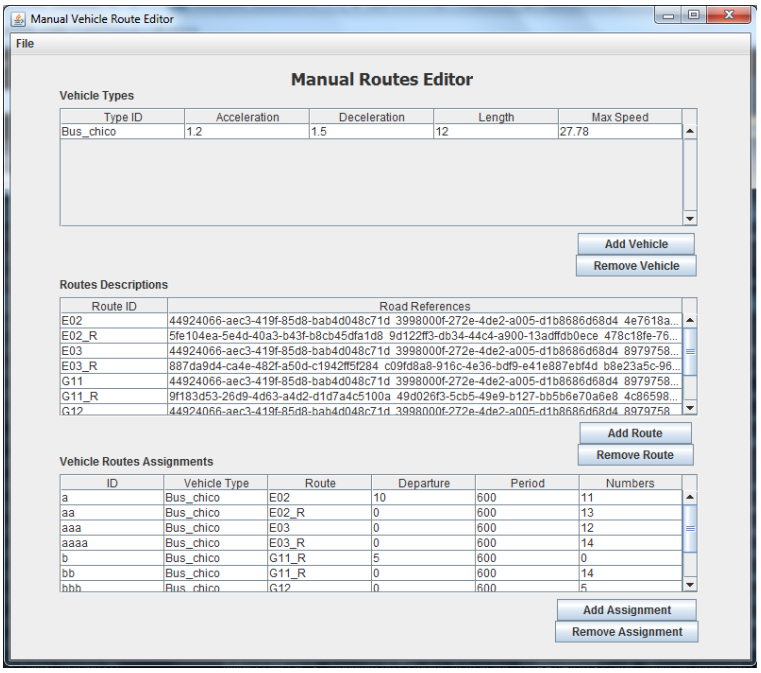

Figura 3-8: Editor manual de rutas del programa MOVE (Karnadi et al. 2007).

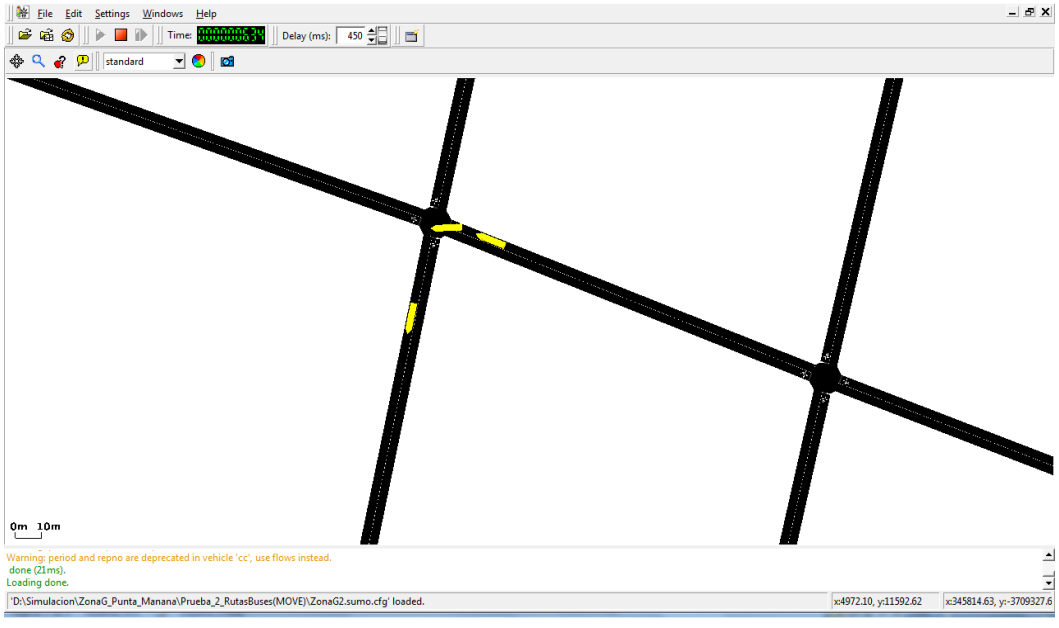

Figura 3-9: Simulación en SUMO (Behrisch et al. 2011).

Las rutas de buses alimentadores que se simularon fueron: E02, E03, G01, G02, G03, G04, G05, G07, G08, G08v, F09, G10, G11, G12, G13, G14, G15, G16, G17, G18, G19, G22, F03, F05, F06, F20, H03, H07, H08, H10, H11, H11c y H18. Los recorridos de buses troncales que se simularon fueron: T113c, T118, T120, T201, T201c, T201e, T201ec, T203, T203e, T205, T205c, T205e, T206, T206e, T206e, T207, T207c, T207e, T209, T209e, T211, T211c, T211e, T212, T214, T216, T219s, T223, T226, T227, T228, T229, T301, T301c, T301c2, T302, T302e, T428, T428c y T428e.

Se utilizaron dos escenarios como base para la simulación, los cuales pertenecen al horario punta mañana y al horario nocturno, que corresponden a las zonas punta y valle, respectivamente.

El comportamiento de cada bus sigue el modelo de seguimiento de vehículos que utiliza SUMO (Behrisch et al. 2011) por defecto, llamado *"Safe Distance Model"* de Stefan Krauss, el cual define una distancia segura entre cada vehículo para casos de frenados imprevistos y accidentes.

3.3 **Simulación de la comunicación inalámbrica**

Para simular la interacción de la simulación de tráfico vehicular y la comunicación inalámbrica, se utilizó el tutorial del proyecto de simulación VEINS (*VEhicles In Network Simulation*) de Sommer (2011), el cual tiene la opción de importar la plataforma INETMANET para el software de simulación de redes OMNET++ (Varga, 2010), que contiene todas las librerías, ejemplos y *stacks* de módulos para inicializar. El programa OMNET++ (Varga, 2010) ejecuta simulaciones basadas en eventos discretos, de nodos comunicándose en una gran variedad de plataformas y es popular en la comunidad de comunicaciones. Los escenarios en OMNET++ (Varga, 2010) son representados por una jerarquía de módulos escritos en C++. La relación y comunicación entre los módulos se almacenan en archivos de Descripción de Red

(NED) y se pueden modelar gráficamente. Las simulaciones en ejecución pueden aparecer en la consola como líneas de comandos o de forma interactiva y gráfica como se observa en la Figura 3-12.

Una vez que OMNET++ (Varga, 2010) reconoce la simulación en paralela en SUMO (Behrisch et al. 2011), es posible programar los módulos de comunicación para cada Bus. Este proceso consta de una capa de aplicación, un *stack* UDP y TCP en la capa de transporte, una capa de red que se conecta directamente con los protocolos de enrutamiento y una capa de enlace como se muestra en la Figura 3-10.

En primer lugar, es necesario establecer el escenario de la simulación, el cual además de tener los buses en movimiento a través del modulo TraCI (*Traffic Control Interface*), requiere las RSUs localizadas, un control del canal inalámbrico y un recopilador de estadísticas. La cantidad de RSUs a simular se varió entre 1, 10, 20, 30, 40, 50, 60, 70 y 80. Las RSUs se localizaron en las calles por donde pasan la mayor cantidad de rutas de buses, tal como se muestra en la Figura 3-11 y cuyo código se expone en el Anexo B.

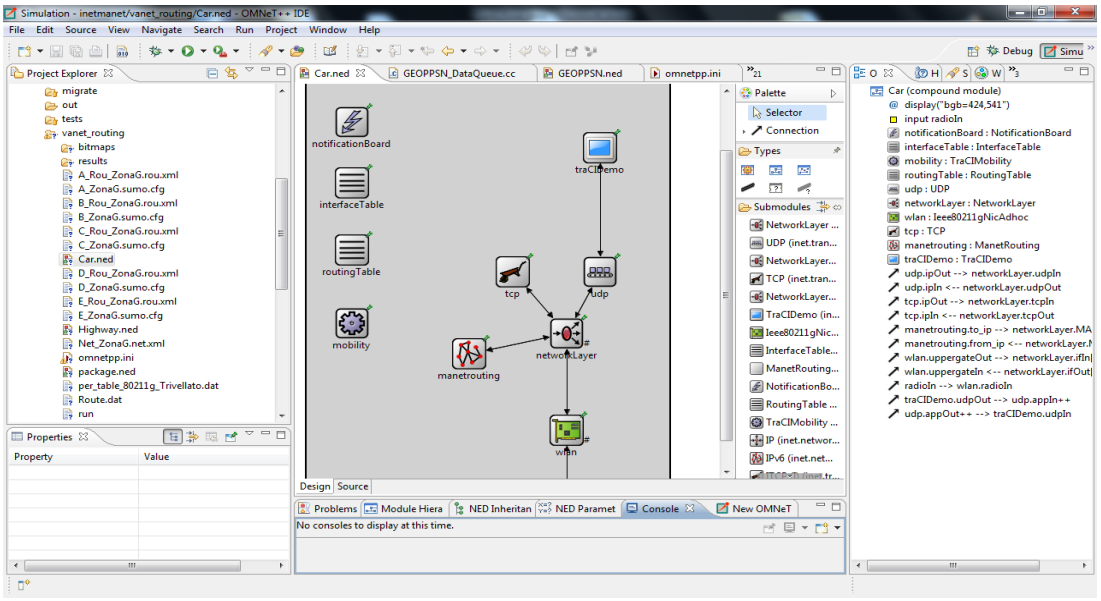

Figura 3-10: Módulos de comunicación de cada bus en OMNET++.

Figura 3-11: 20 RSUs localizadas (Izquierda) y 80 RSUs localizadas (Derecha).

Una vez establecido cada escenario, se modifica la capa de aplicación, tanto para las RSUs (*Fix Manet Routing Host*) como para los buses. A estos últimos se les indica que deben enviar mensajes de posición, cada 30 segundos, a la RSU más cercana que tenga a su alcance. El código para los módulos la capa de aplicación tanto para los buses como las RSUs es un aporte de Kuprian (2010).

Para programar el protocolo de enrutamiento GeOpps-N, se tomó como base el código creado para DYMO-FAU por Sommer y Dressler (2007), que viene dentro del *stack* de INETMANET. Este protocolo es un protocolo reactivo, donde cada vez que un nodo origen tiene que enviar un paquete a un destino, el nodo origen inunda la red con paquetes RREQ en busca del destino, y cada nodo que recibe estos paquetes RREQ, almacena en su tabla de ruteo la información sobre ruta hacia el nodo origen y las rutas hacia el resto de los nodos intermedios por los que el paquete RREQ ha sido transmitido, y reenvía el paquete RREQ hacia otros nodos, hasta encontrar el destino. El nodo destino, tras recibir el paquete RREQ, envía un paquete RREP a través de la primera ruta que recibe. Cuando ocurre el rompimiento de un enlace establecido en la tabla de ruteo, los nodos involucrados inundan la red con paquetes RERR (*Route ERRor message*), invalidando todas las rutas que vinculan a estos nodos. Este protocolo es similar al protocolo GeOpps-N propuesto en el modo local, pero GeOpps-N, en lugar de buscar el nodo destino, busca aquellos nodos con un menor valor METD que el nodo origen. Esto obligó a incluir nuevos métodos y atributos en las clases originales del código de DYMO-FAU, escrito en C++, para crear el módulo de GeOpps-N. Las modificaciones son las siguientes:

- Se creó un método en el *main* que calcula el valor METD (Anexo C), para este propósito fue necesario incluir las coordenadas de la RSU destino del bus origen, como atributo a los mensajes de ruteo. En esencia el método lee el archivo con las coordenadas de las rutas "*BusRoutes*", y busca la ruta que corresponde al recorrido del bus que tiene el mensaje de ruteo. Luego busca entre qué coordenadas de la ruta

se encuentra el bus en análisis, para determinar la distancia hasta su NP (*Nearest Point*). Por último calcula el valor del METD utilizando los valores de la velocidad instantánea del bus en análisis, la distancia del bus hasta su NP, la métrica de velocidad entre el NP del bus en análisis y la RSU destino, y la distancia euclidiana entre las coordenadas de NP y las coordenadas de la RSU destino.

- Se agregó a la clase "*DataQueue*", que administra las colas de datos (datagramas) a enviar a los destinos que no se encuentran en la tabla de ruteo, un método (Anexo D) que cambia el destino original de los datos programados (RSU destino) por un destino temporal (bus con menor valor METD que el valor METD del bus origen).
- A los datagramas se le incorporaron los atributos: Coordenadas de la RSU destino y dirección IP del destino temporal (bus con menor valor METD). Esta información necesariamente debe adjuntarse a los datos, para calcular el valor METD en cada nodo y notificar a un nodo si es que es un nodo temporal o un nodo destino.
- Se agregó, como atributo, el valor METD del bus origen (SrcMETD) a los mensajes de ruteo, además de las coordenadas de la RSU destino. El valor METD del bus origen es necesario para ser comparado con el valor METD de los buses donde llega el mensaje de ruteo, y así, enviar un paquete RREP en caso de que este valor METD sea menor al valor METD del bus origen.

En la Tabla 3 se muestran los parámetros usados en cada capa del modelo OSI para la simulación. En la capa de red se incluyó un archivo llamado "route.dat", con las coordenadas por donde pasan los recorridos, para que el protocolo de enrutamiento pueda realizar los cálculos del METD. También se estableció un tiempo de espera de 3 [s] para cada inundación de los paquetes RREQ, y un número de intentos de envío de 8, para ocupar todo el intervalo de tiempo de espera de 30 [s], entre el reporte de cada posición y los límites de alcance de estos RREQ en 20 saltos.

Parameter	Description	Value
Network Layer		
**.manetrouting.manetmanager.AUTOASSIGN_ADDRESS	Autoassign IP Address	true
**.BusRoutes	Coords of the routes from the buses	"Route.dat"
**.RREQ_WAIT_TIME	Time to wait to repeat RREQ	3 [s]
**.RREQ_TRIES	Times to retry a RREQ message	8
**.MIN_HOPLIMIT	Times of HOPS of the first sending RREQ	20
**.MAX_HOPLIMIT	Times of HOPS of the last sending RREQ	20
Wifi Link Layer		
**.wlan.mac.address	Setting of MAC address	"auto"
**.wlan.mac.maxQueueSize	Limit size of the queue of the MAC layer	14
**.wlan.mac.rtsCts	Use of the mechanism RTS-CTS	false
**.wlan.mac.bitrate	Bit rate for the MAC module	12 [Mbps]
**.wlan.mac.broadcastBackoff	Broadcast Backoff	31
**.wlan.decider.snirThreshold	Minimum level for the SNIR	10 [dB]
**.wlan.decider.bitrate	Bit rate	12 [Mbps]
**.wlan.snrEval.bitrate	Bit rate	12 [Mbps]
**.wlan.snrEval.headerLength	Header Length	192 [bytes]
**.wlan.snrEval.thermalNoise	Base noise level	-110 [dBm]
**.wlan.snrEval.sensitivity	Signals below this level are ignored	-95 [dBm]
**.wlan.snrEval.pathLossAlpha	Path Loss Alpha	2,7
**.wlan.snrEval.carrierFrequency	Carrier Frecuency	5,89 [GHz]
**.wlan.snrEval.channelNumber	Channel identifier	0
*.channelcontrol.carrierFrequency	Carrier Frecuency	5,89 [GHz]
*.channelcontrol.pMax	Maximum Transmitted Power	372,93 [mW]
*.channelcontrol.sat	Saturation from the receiver	-95 [dBm]
*.channelcontrol.alpha	Path Loss Alpha	2,7
*.channelcontrol.numChannels	Number of Channels	1
Physical Layer		
**.wlan.radio.transmitterPower	Transmitted Power	372,93 [mW]
**.wlan.radio.bitrate	Bit rate	12 [Mbps]
**.wlan.radio.sensitivity	Sensitivity	-95 [dBm]
**.wlan.radio.carrierFrequency	Carrier Frecuency	5,89 [GHz]
**.wlan.radio.pathLossAlpha	Path Loss Alpha	2,7
**.wlan.radio.TransmiterAntennaHigh	Transmiter Antenna Height	4 [m]
**.wlan.radio.ReceiverAntennaHigh	Receiver Antenna Height	4 [m]

Tabla 3: Parámetros de las capas de red, enlace y física de la simulación

Dentro de los parámetros de la capa física, la tasa de datos corresponde a la más ocupada dentro de las opciones de WAVE, la frecuencia portadora se extrae de la banda DSRC para el estándar en EEUU, la cual en Chile también se encuentra disponible para comunicación móvil según la SUBTEL. La sensibilidad en la recepción (P_r) se obtuvo de pruebas reales con equipos WSU (*Wireless Safety Unit*) incorporados en vehículos, pruebas realizadas por Miucic, Popovic, y Mahmud (2009).

La componente α, de pérdida por desvanecimiento de sombra, corresponde a ambientes urbanos como es el del PTS de Santiago. Las alturas de las antenas transmisoras y receptoras, suponen su ubicación en el techo de los buses. Por último, se calcula la potencia a transmitir (P_t) para tener una cobertura de $d = 500[m]$ de alcance de acuerdo a los parámetros mencionados y utilizando la Ecuación (3.1) del modelo de Friis.

$$P_r = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi)^2 \cdot d^{\alpha}} \qquad (3.1)$$

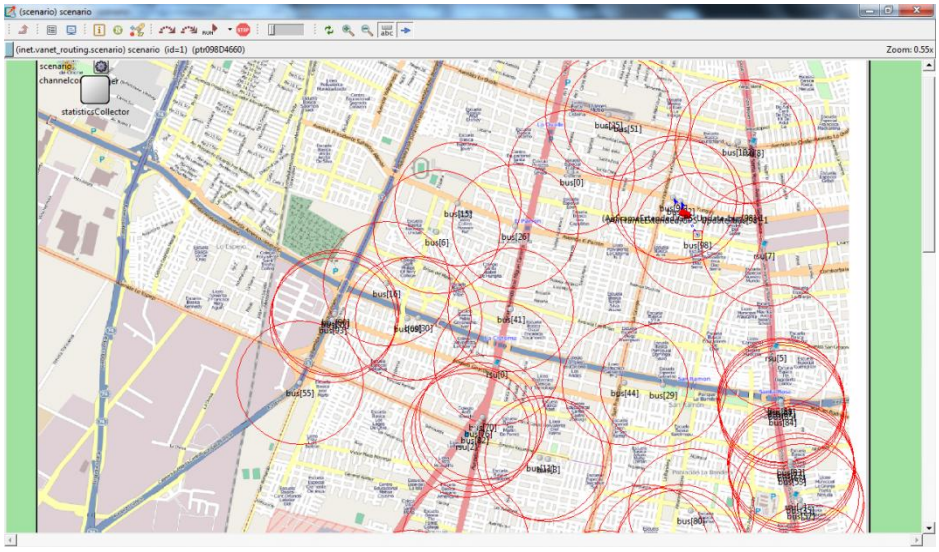

Figura 3-12. Interfaz gráfica de la simulación final en OMNET++.

3.4 **Análisis de los resultados**

La superficie donde ocurre la simulación de GeOpps-N, se configuró como un rectángulo de 8.350 m por 12.180 m, y se compararon los resultados con aquellos de los protocolos DYMO (DYMO-UM y DYMO-FAU) y GeOpps.

Para generar los escenarios de simulación, se tomó como base la información operativa real de horarios y recorridos de los buses del PTS en la zona valle (Nocturna entre las

01:00 y 05:30) y en la zona punta (Mañana entre las 06:30 y 08:30), las que corresponden a los escenarios A y D de la Tabla 4, respectivamente. Los escenarios B, C y E corresponden a zonas de transición, generados a partir del escenario D, cuyas densidades son 1/3, 2/3 y 4/3 de la del escenario D, respectivamente.

Para disminuir el tiempo de computación, sólo se consideraron 30 minutos (1800 segundos), entre los segundos 2000 y 3800 mostrados en la Figura 3-13, ya que desde el segundo 2000 en adelante, la densidad de buses en todos los casos se estabiliza en un valor aproximadamente constante.

Escenario	Buses promedio
A	40
B	104
C	216
D	303
E	405

Tabla 4. Escenarios de simulación para cada densidad de buses

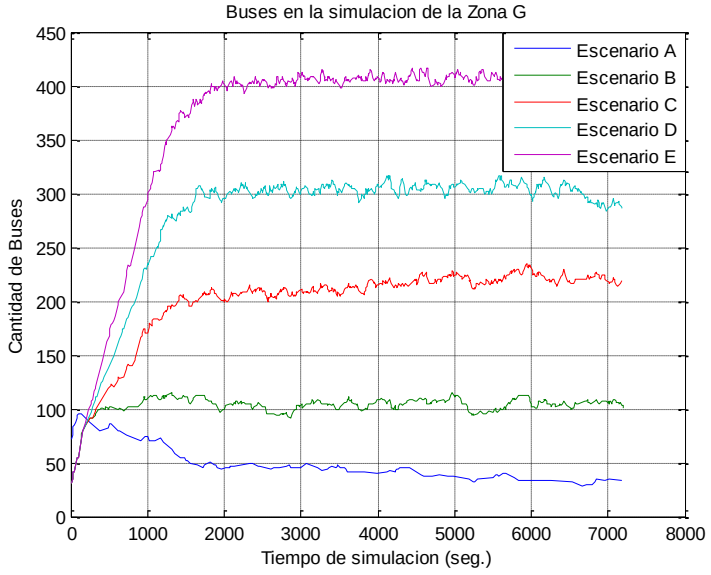

Figura 3-13: Cantidad de buses en la simulación, en cada instante de tiempo.

La Figura 3-14 muestra los resultados para el retardo promedio, con las distintas densidades, tanto de busescomo de RSUs. Se observa que GeOpps-N alcanza un retardo promedio menor al de el protocolo GeOpps, siendo este en promedio un 36% más bajo y llegando en algunos casos a requerir 38 segundos menos como se observa en la Figura 3-15 para el escenario C. Las mayores diferencias se dan en los escenarios de baja densidad de RSUs, debido a que al tener pocas RSUs se tiende a usar mucho más el transporte de los datos mediante *piggybacking* y, como GeOpps-N considera la conectividad, GeOpps-N utiliza muchas veces a los mejores candidatos para el transporte, a diferencia del protocolo GeOpps que sólo considera a los nodos vecinos para transportar los datos. Los retardos promedios de las versiones de DYMO son muchos menores y se encuentran bajo los dos segundos en todo momento. Este resultado es esperable, dada su naturaleza de protocolos basados en la topología.

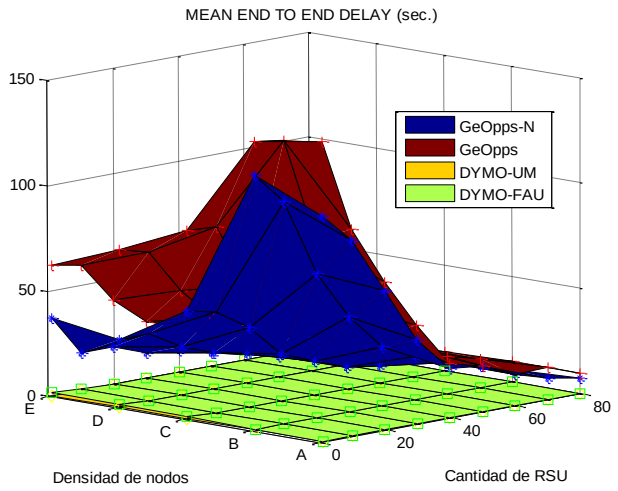

Figura 3-14. *Mean End to End Delay* para cada escenario de simulación.

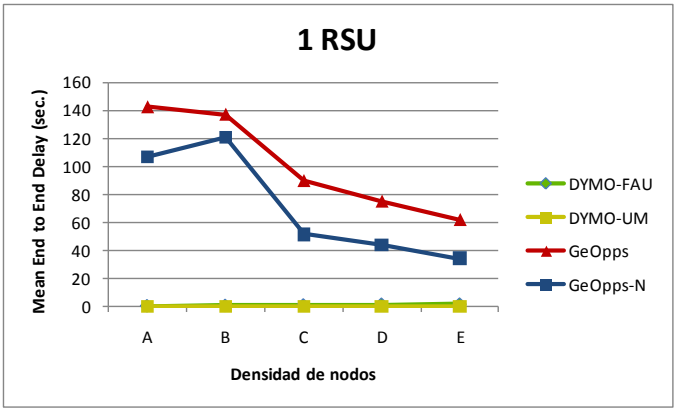

Figura 3-15. *Mean End to End Delay* para escenarios con 1 RSU.

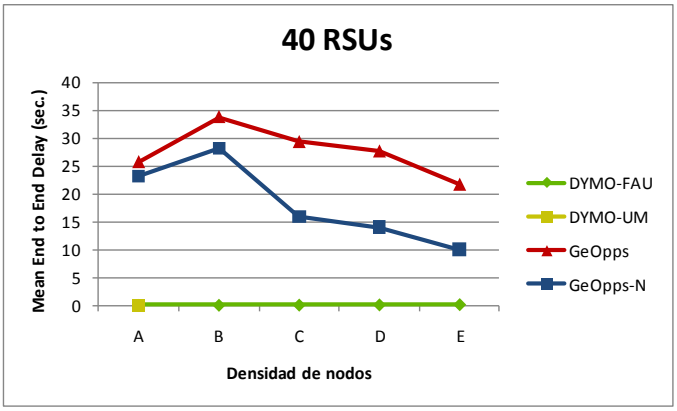

Figura 3-16. *Mean End to End Delay* para escenarios con 80 RSUs.

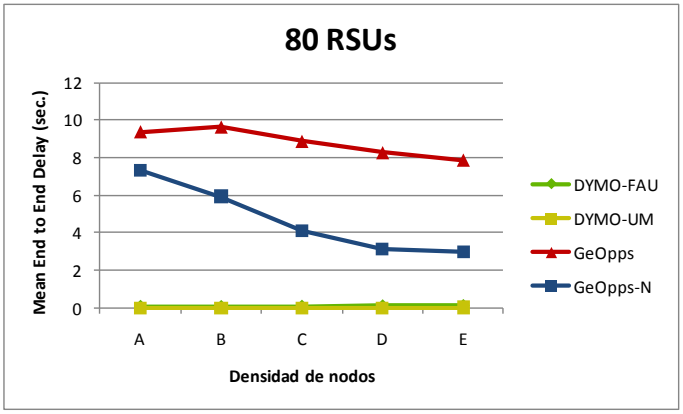

Figura 3-17. *Mean End to End Delay* para escenarios con 80 RSUs.

En aquellos casos de una baja densidad de RSUs, el protocolo GeOpps-N supera a todos los demás protocolos, en la tasa de paquetes enviados exitosamente. GeOpps-N es un 59% mejor que GeOpps, un 128% mejor que DYMO-UM y un 159% mejor que DYMO-FAU, como se observa en la Figura 3-19 en el escenario E. A medida que aumenta la cantidad de RSUs, GeOpps-N, tiende a tener una tasa similar a GeOpps, ya que el transporte de datos mediante *piggybacking* es menos frecuente, y los paquetes se enrutan directamente debido a la conectividad con las RSUs. La diferencia principal entre los protocolos del tipo DTN y aquellos reactivos, se refleja en la Figura 3-18, en donde se observa que la tasa de paquetes enviados exitosamente por los protocolos GeOpps y GeOpps-N, se mantiene sobre el 50% para más de 1 RSU y cualquier densidad. En cambio los protocolos reactivos necesitan por lo menos 30 RSUs para estar sobre esta tasa.

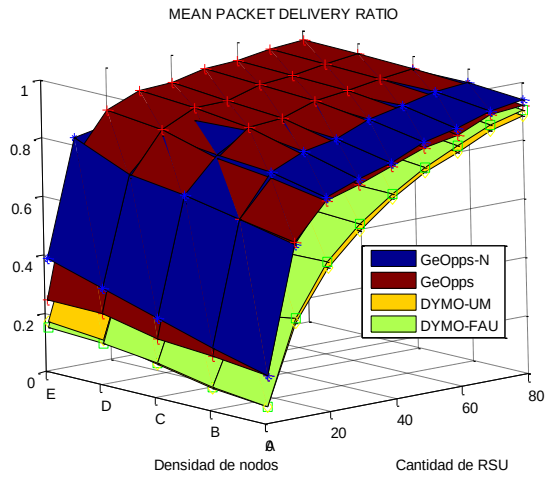

Figura 3-18. *Mean Packet Delivery Ratio* para cada escenario de simulación

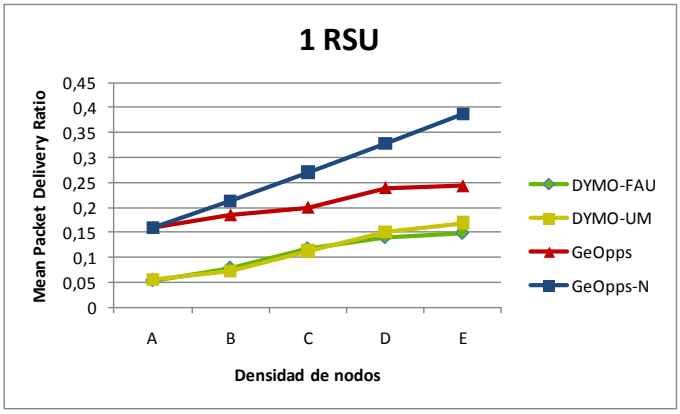

Figura 3-19. *Mean Packet Delivery Ratio* para escenarios con 1 RSU.

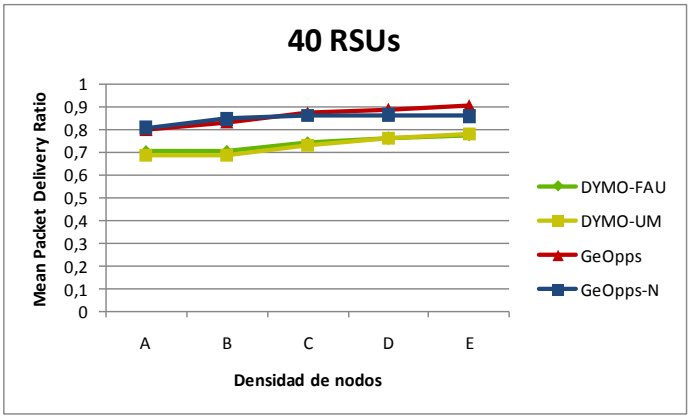

Figura 3-20. *Mean Packet Delivery Ratio* para escenarios con 40 RSUs.

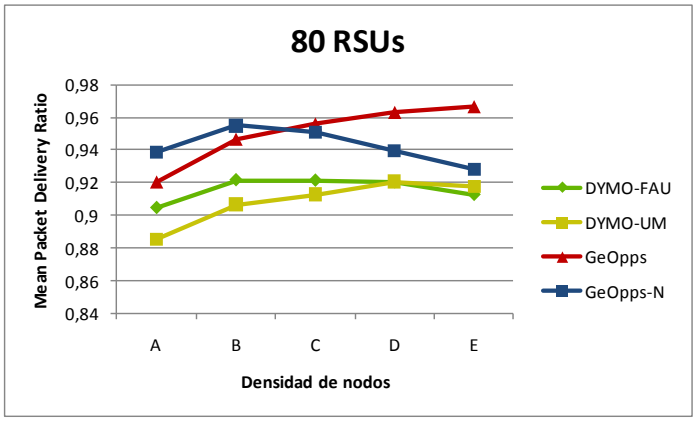

Figura 3-21. *Mean Packet Delivery Ratio* para escenarios con 80 RSUs.

El costo asociado con la mejora en las medidas de desempeño de retardo promedio y tasa exitosa de entrega de datos, es un aumento en la sobrecarga de la red, que representa la cantidad de paquetes totales que se necesita envíar por cada paquete de dato que llega finalmente a su destino. Para el caso de GeOpps-N, como se aprecia en la Figura 3-23, la Figura 3-24 y la Figura 3-25, esta sobrecarga decae a medida que se aumenta la cobertura de RSUs, pareciéndose al resto de los protocolos. La sobrecarga es mayor que para el caso de GeOpps debido a que GeOpps-N esta continuamente buscando mejores candidatos para el acarreo, inundando parte de la red con RREQ.

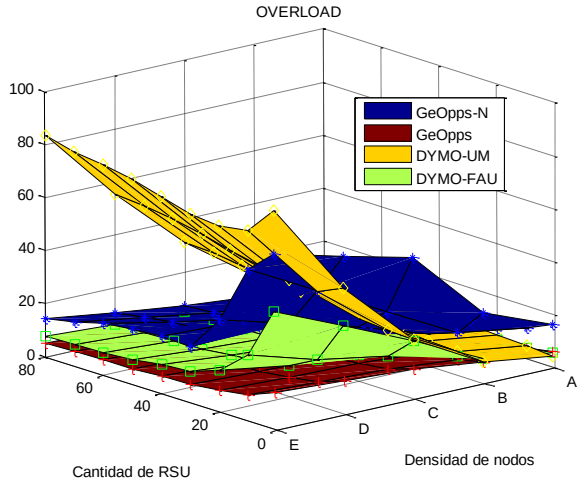

Figura 3-22. *Overload* para cada escenario de simulación.

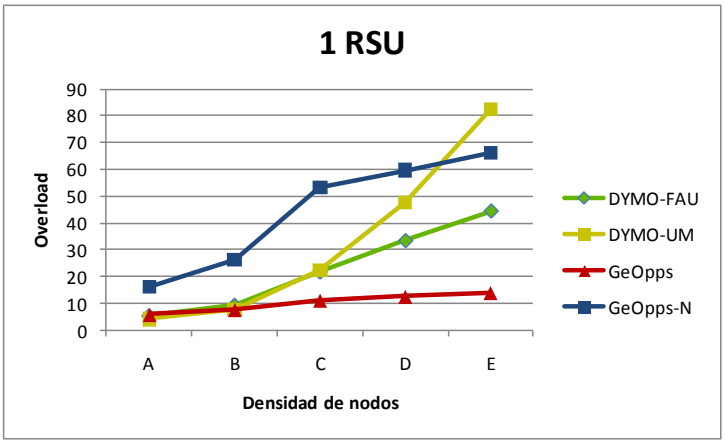

Figura 3-23. *Overload* para escenarios con 1 RSU.

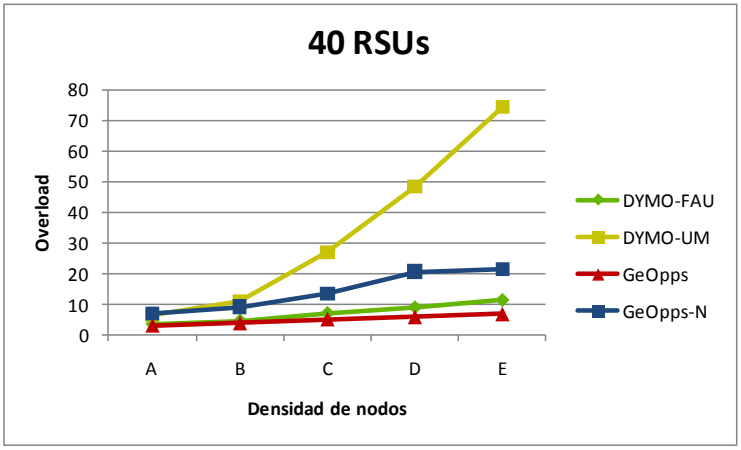

Figura 3-24. *Overload* para escenarios con 40 RSUs.

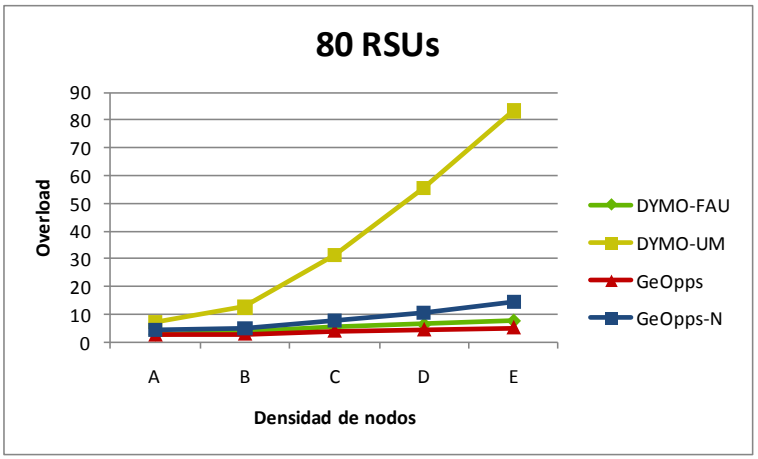

Figura 3-25. *Overload* para escenarios con 80 RSUs.

4 CONCLUSIONES

En esta tesis, se evalúa la factibilidad del uso de tecnologías de comunicación inalámbrica para redes VANETs en el PTS de Santiago. El PTS Transantiago requiere aumentar el tiempo de monitoreo de los buses, los cuales se encuentran inubicables el 20% del tiempo, por lo que, se modela un sistema de localización de buses con RSUs distribuidas en la red del Transantiago. En este contexto se diseña el protocolo de enrutamiento GeOpps-N, cuyo objetivo es cumplir de la forma más eficiente los requerimientos del PTS de Santiago.

El protocolo de enrutamiento propuesto GeOpps-N, permite el transporte de paquetes de datos en redes ad-hoc dinámicas y de alta mobilidad las que, por lo general, se encuentran conectadas de una manera intermitente, como es el caso de las redes VANETs. El protocolo GeOpps-N tiene la ventaja inherente que un mensaje es transportado de la manera más rápida y lejos posible, vía el enrutamiento a través de *multihops*. Si el nodo de destino tiene conectividad con el nodo origen, el mensaje es enviado de manera inmediata. En el caso contrario, el paquete lo transportará aquel nodo, que tiene conectividad con el nodo origen, y una mayor probabilidad de entregar el mensaje al nodo destino. De esta forma, el protocolo abarca todos los casos de conectividad posibles, maximizando así la tasa de entrega de datos en forma exitosa.

En los escenarios evaluados de localización de buses, con el uso de RSUs, como el modelado en la zona G del PTS de Santiago, se obtienen mejoras para todos los protocolos de enrutamiento, en métricas de desempeño como la tasa promedio de entrega de datos y el retardo promedio de principio a fin, a medida que se aumenta la cantidad de RSUs distribuidas en la red. Esto se debe al aumento en la cobertura de las RSUs, claramente la cantidad de RSUs a localizar se restringe con los costos de implementación y mantención de cada RSU.

El aumento de la densidad de los buses en la red, dados por las horas punta del PTS, mejora la conectividad entre los buses y las RSUs, por lo tanto, aumenta la tasa promedio de entrega de datos y disminuye el retardo promedio de principio a fin, para todos los protocolos de enrutamiento. Aunque las mejoras obtenidas, al aumentar la densidad de buses, ocurren en un menor grado que al aumentar la cantidad de las RSUs.

El protocolo propuesto GeOpps-N supera en términos de la tasa de entrega de datos en forma exitosa hasta en un 159% a aquellos protocolos reactivos, tales como DYMO-UM y DYMO-FAU. También el protocolo propuesto GeOpps-N, supera en esta misma medida de desempeño, hasta en un 59% a el protocolo DTN GeOpps. Los mejores resultados en la tasa de entrega de datos para el protocolo propuesto GeOpps-N, en comparación con el resto de los protocolos, ocurren en escenarios de alta densidad de buses y con pocas RSUs. Al tener una alta conectividad y pocos nodos destinos, se potencia el uso de la propiedad de alcance como protocolo reactivo y el transporte de datos mediante *piggybacking* con el bus que sea mejor candidato, respectivamente, para el protocolo propuesto GeOpps-N.

También, el protocolo propuesto GeOpps-N muestra una mejora significativa, en términos del retardo promedio, de un 36% con respecto a GeOpps. La disminución en esta métrica de desempeño era de esperarse, ya que el diseño del protocolo propuesto GeOpps-N tiene como objetivo, disminuir el retardo del protocolo GeOpps, mediante la información de conectividad de la red.

BIBLIOGRAFIA

Chung, J., Kim M., Park Y., Choi M., Lee S. y Oh H. (2011). Time Coordinated V2I Communications and Handover for WAVE Networks. *IEEE Journal on Selected Areas in Communications, VOL. 29, No. 3.*

Gukhool, B.S. y Cherkaoui, S. (2008). IEEE 802.11p modelling in NS-2. *33rd IEEE Conference on Local Computer Networks, 622-626.*

Lee, K. C. y Gerla, M. (2010). Opportunistic Vehicular Routing. *European Wireless Conference (EW), 873-880*.

Leontiadis, I. y Mascolo, C. (2007). GeOpps: Geographical Opportunistic Routing for Vehicular Networks. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM),* 1-6.

Karp, B. y Kung H. T. (2000). GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. *Proceedings of the 6th annual international conference on Mobile computing and networking, 243-258.*

Jerbi, M., Senouci, S., Meraihi, R. y Ghamri-Doudane, Y. (2007). An Improved Vehicular Ad Hoc Routing Protocol for City Environments. *IEEE International Conference on Communications*, *3972-3979*.

Seet, B., Liu, G., Lee, B., Foh, C., Wong, K. y Lee, K. (2004). A-STAR: A Mobile Ad Hoc Routing Strategy for Metropolis Vehicular Communications. *Networking 2004. Networking Technologies, services, and protocols; Performance of Computer and communications networks; Mobile and Wireless Communications, VOL. 3042, 989-999.*

Li, F. y Wang, Y. (2007). Routing in vehicular ad hoc networks: A survey. *Vehicular Technology Magazine, IEEE, VOL. 2, No. 2, 12-22.*

Miucic, R., Popovic, Z. y Mahmud, S.M. (2009). Experimental characterization of DSRC signal strength drops. *12th International IEEE Conference on Intelligent Transportation System (ITSC), 1-5.*

Sommer, C. y Dressler, F.(2007). The DYMO Routing Protocol in VANET Scenarios. *Vehicular Technology Conference, 2007, VTC-2007 Fall. 2007 IEEE 66th , 16-20.*

Kuprian, H. (2010). A VANET for Santiago: Evaluation of Information Dissemination Characteristics on the Basis of different Traffic Situations and Infrastructures. *Diplomarbeit KOM-D-0423, Fachbereich Elektrotechnik und Informationstechnik, Technische Universität Darmstadt, Germany.*

Karnadi, F.K., Mo, Z.H y Lan, K. (2007). Rapid Generation of Realistic Mobility Models for VANET, *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE, 2506 -2511.*

Li, Y., Wang, Z., Jin. D., Su, L., Zeng, L. y Chen, S. (2011). Optimal Relaying in Heterogeneous Delay Tolerant Networks, *IEEE International Conference on Communications (ICC), 2011, 1 -5.*

Transantiago Official Documentation, Anex G: Communications Systems, page 20.

INE, Mapa interactivo del Instituto Nacional de Estadísticas. Recuperado de http://www.ine.cl/canales/chile_estadistico/censos_poblacion_vivienda/censo2002/mapa_interactivo/inicio.swf

Sommer, C. (2011), VEINS (*VEhicles In Network Simulations*) Simulation Framework. Disponible en the Computer Networks and Communication Systems, University of Erlangen: http://veins.car2x.org/tutorial/

Behrisch, M., Bieker, L., Erdmann, J. y Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility: An Overview. *SIMUL 2011, The Third International Conference on Advances in System Simulation*, 63-68. SUMO (Versión 0.12.3) [Software]. Disponible en: http://sumo.sourceforge.net/

Varga, A. (2010), OMNET++ (Versión 4.2b2) [Software] Disponible en: http://www.omnetpp.org/

© Colaboradores de OpenStreetMap, CC-BY-SA. Mapa virtual de Santiago. Disponible en: http://www.openstreetmap.org/

Edgewall Software. JOSM (Versión 4550) [Software] Disponible en: http://josm.openstreetmap.de/

Meinel, C. y Schünemann, B. (2011). eWorld (Versión 0.9.2) [Software] Disponible en the Hasso Plattner Institute in Potsdam, Germany: http://eworld.sourceforge.net/

A N E X O S

ANEXO A: EXTRACCIÓN DE CÓDIGOS DE EDGES EN MATLAB

Para procesar los archivos del tipo *.edg.xml, *.nod.xml y *.poi.xml generados por eWorld 0.9.2 y extraer los códigos correspondientes a los enlaces por los que pasa cada ruta del transantiago en la Zona G, se crearon los siguientes códigos en MATLAB de extracción, se muestra sólo el caso para la ruta E02_ida.

```
%-------------------------- POi_Data.m -----------------------------
clc
clear all
Nod_fid = fopen('ZonaG_POI2_V4.nod.xml');%abrir el archivo
Nod_File = fscanf(Nod_fid,'%c');

POI_fid = fopen('ZonaG_POI2_V4.poi.xml');%abrir el archivo
POI_File = fscanf(POI_fid,'%c');
i=1;
resto=POI_File;
while length(resto)~=0
     [palabra,resto] = strtok(resto,'"');
     Node_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     NodeID_inicial{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     x_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     X_inicial{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     y_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Y_inicial{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Type_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Poi_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
      desc_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     POI_ID{i}=palabra;
     i=i+1;
end
X_ini = str2double(X_inicial);
Y_ini = str2double(Y_inicial);

i=1;
resto=Nod_File;
while length(resto)~=0
     [palabra,resto] = strtok(resto,'"');
     Node_bla{i}=palabra;
```

```
     [palabra,resto] = strtok(resto,'"');
     NodeID{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     x_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     X_nodos{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     y_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Y_nodos{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Type_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Priority_bla{i}=palabra;
     i=i+1;
end
X_nod = str2double(X_nodos);
Y_nod = str2double(Y_nodos);

L = length(X_ini);
i=1;
for i=1:L
    [aux,aux2] = min(sqrt((X_ini(i)-X_nod).^2+(Y_ini(i)-Y_nod).^2));
    Posicion_Nodo_Real(i) = aux2;
end

POI_ID_Real = NodeID(Posicion_Nodo_Real);
 Edge_fid = fopen('ZonaG_POI2_V4.edg.xml');%abrir el archivo
 Edge_File = fscanf(Edge_fid,'%c');

i=1;
resto=Edge_File;
while length(resto)~=0
     [palabra,resto] = strtok(resto,'"');
     Edge_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     EdgeID{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     fromNode_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     From_Node{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     toNode_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     To_Node{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Priority_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Priority{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Nolanes_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Nolanes{i}=palabra;
```

```
     [palabra,resto] = strtok(resto,'"');
     Speed_bla{i}=palabra;
     [palabra,resto] = strtok(resto,'"');
     Speed{i}=palabra;
     i=i+1;
end

% Vectores importantes: POI_ID, POI_ID_Real, From_Node, To_Node, EdgeID

i=1;
L = length(NodeID);
for i=1:L
    aux = strcmp(NodeID(i), From_Node);
    aux2 = find(aux==1);
    for j=1:length(aux2)
        aux3 = To_Node(aux2);
        aux4 = find(strcmp(aux3(j),NodeID));
        Matriz_NOD2NOD(i,j) = aux4 ;
    end
    NOD_Matriz_To_Nodes(i,1:length(aux2)) = To_Node(aux2);
    NOD_Matriz_Edges(i,1:length(aux2)) = EdgeID(aux2);
end

i=1;
L = length(POI_ID_Real);
for i=1:L
    aux = strcmp(POI_ID_Real(i), From_Node);
    aux2 = find(aux==1);
    for j=1:length(aux2)
        aux3 = To_Node(aux2);
        aux4 = find(strcmp(aux3(j),NodeID));
        Matriz_NOD2POI(i,j) = aux4 ;
    end
    POI_Matriz_To_Nodes(i,1:length(aux2)) = To_Node(aux2);
    POI_Matriz_Edges(i,1:length(aux2)) = EdgeID(aux2);
end
    POI_Matriz_To_Nodes = [POI_ID_Real',POI_Matriz_To_Nodes];

for i=1:L
    aux2 = strcmp(POI_Matriz_To_Nodes(i,2),POI_ID_Real);
    aux3 = strcmp(POI_Matriz_To_Nodes(i,3),POI_ID_Real);
    aux4 = strcmp(POI_Matriz_To_Nodes(i,4),POI_ID_Real);
    aux5 = strcmp(POI_Matriz_To_Nodes(i,5),POI_ID_Real);
    aux22 = find(aux2);
    aux33 = find(aux3);
    aux44 = find(aux4);
    aux55 = find(aux5);
    if (length(aux22)~= 0)
    POI_Matriz_Names(i,1)= POI_ID(aux22(1));
    end
    if (length(aux33)~= 0)
    POI_Matriz_Names(i,2)= POI_ID(aux33(1));
    end
```

```
    if (length(aux44)~= 0)
    POI_Matriz_Names(i,3)= POI_ID(aux44(1));
    end
    if (length(aux55)~= 0)
    POI_Matriz_Names(i,4)= POI_ID(aux55(1));
    end
    aux22=0;
    aux33=0;
    aux44=0;
    aux55=0;
    i=i+1;
end
r=length(POI_Matriz_Names(1:end,1));
POI_Matriz_Names = [POI_ID(1:r)',POI_Matriz_Names];

X_POI = X_nod(Posicion_Nodo_Real);
Y_POI = Y_nod(Posicion_Nodo_Real);

%----------------------- ALIMENTADORES ZONA E -------------------------
-
E02_Names_ida = {'Paulina / Briones Luco (NP)',...
'Ciencias / Briones Luco (NP)','Ciencias / Sergio Ceppi',...
'Ciencias / Locarno (NP)','Ciencias / General Freire',...
'Santa Rosa / Santa Ana','Parada 8 / (M) Santa Rosa',...
'Parada 3 / P. 26 Santa Rosa'};

[E02_Edges_ida, pos_nod_E02] = Find_edges(E02_Names_ida, POI_ID,...
POI_Matriz_Names, POI_Matriz_Edges, Matriz_NOD2POI,...
X_POI, Y_POI, X_nod, Y_nod, Matriz_NOD2NOD, NOD_Matriz_Edges);
E02=strvcat(E02_Edges_ida);
E02(1,37)=' ';

%-------------------------- Find_edges.m -------------------------------
function   [E02_Edges_ida,   pos_nod]   =   Find_edges(E02_Names_ida,
POI_ID,... POI_Matriz_Names, POI_Matriz_Edges, Matriz_NOD2POI,...
X_POI, Y_POI, X_nod, Y_nod, Matriz_NOD2NOD, NOD_Matriz_Edges)

L=length(E02_Names_ida);
E02_Edges_ida = {};
pos_nod = 0;
j=1;
for i=1:L-1
k1=strcmp(E02_Names_ida(i),POI_ID);
aux1 = find(k1); aux1 = aux1(1);
k2=strcmp(E02_Names_ida(i+1),POI_ID);
aux2 = find(k2); aux2 = aux2(1);
aux_Name = POI_Matriz_Names(aux1,:)';
k3=strcmp(POI_ID(aux2),aux_Name);

if(sum(k3)~=0)
   aux3 = find(k3)-1;
   E02_Edges_ida(j)= POI_Matriz_Edges(aux1,aux3);
   pos_nod(j)= Matriz_NOD2POI(aux1,aux3);
```

```
   j=j+1;
end

if(sum(k3)==0)
   ind_nod_ady = Matriz_NOD2POI(aux1,find(Matriz_NOD2POI(aux1,:)));
%  Guarda  los  valores  distintos  de  cero  del  vector  de  indices
%Matriz_NOD2POI(aux1,:)
   dist = sqrt((X_nod(ind_nod_ady)-X_POI(aux2)).^2 + ...
(Y_nod(ind_nod_ady)-Y_POI(aux2)).^2);
% Calcula las distancias entre los adyacentes de i, e i+1
   ind_elec = find(dist==min(dist));
% Elige aquel nodo adyacente de i que este más cerca de i+1
   E02_Edges_ida(j) = POI_Matriz_Edges(aux1,ind_elec);
% Guarda este "Edge" en la ruta
   pos_nod(j)= Matriz_NOD2POI(aux1,ind_elec);
   j=j+1;
   while dist~=0
     ind_nod_elegido = ind_nod_ady(ind_elec);
     ind_nod_ady                                                    =
Matriz_NOD2NOD(ind_nod_elegido,find(Matriz_NOD2NOD(ind_nod_elegido,:)))
;
     dist      =      sqrt((X_nod(ind_nod_ady)-X_POI(aux2)).^2      +...
(Y_nod(ind_nod_ady)-Y_POI(aux2)).^2);
% Calcula las distancias entre nodos adyacentes de i, e i+1
     ind_elec = find(dist==min(dist));
% Elige aquel nodo adyacente de i que este más cerca de i+1
     E02_Edges_ida(j) = NOD_Matriz_Edges(ind_nod_elegido,ind_elec);
% Guarda este "Edge" en la ruta
     pos_nod(j)= Matriz_NOD2NOD(ind_nod_elegido,ind_elec);
     j=j+1;
   end
end
end
```

ANEXO B: LOCALIZACIÓN DE LAS RSUS

Para enlistar los nodos donde pasaban los recorridos de más a menor recurrente para posteriormente localizar las RSUs se creó el siguiente código, tomando como datos base los nodos y enlaces por los que pasan los recorridos en un archivo guardado anteriormente del tipo *.mat en MATLAB.

```
% -------------------------- RSU.m ---------------------------------
clc
aa_IDA                                                              =
[pos_nod_E02,pos_nod_E03,pos_nod_F03,pos_nod_F05,pos_nod_F06,...
pos_nod_F20,pos_nod_G01,pos_nod_G02,pos_nod_G03,pos_nod_G04,...
pos_nod_G05,pos_nod_G07,pos_nod_G08,pos_nod_G08v,pos_nod_G09,...
pos_nod_G10,pos_nod_G11,pos_nod_G12,pos_nod_G13,pos_nod_G14,...
pos_nod_G15,pos_nod_G16,pos_nod_G17,pos_nod_G18,pos_nod_G19,...
pos_nod_G22,pos_nod_H03,pos_nod_H03,pos_nod_H03,pos_nod_H03,...
pos_nod_H03,pos_nod_H03,pos_nod_H10,pos_nod_T113e,pos_nod_T118,...
pos_nod_T120,pos_nod_T201,pos_nod_T201c,pos_nod_T201e,...
pos_nod_T201e,pos_nod_T203,pos_nod_T203,pos_nod_T205,pos_nod_T205,...
pos_nod_T205,pos_nod_T206,pos_nod_T206,pos_nod_T207,...
pos_nod_T207,pos_nod_T207,pos_nod_T207,pos_nod_T207,pos_nod_T211,...
pos_nod_T211c,pos_nod_T211c,pos_nod_T212,pos_nod_T214e,...
pos_nod_T216,pos_nod_T219s,pos_nod_T223,pos_nod_T226,pos_nod_T227,...
pos_nod_T228,pos_nod_T229,pos_nod_T301,pos_nod_T301c,...
pos_nod_T301c2,pos_nod_T302,pos_nod_T302,pos_nod_T428,pos_nod_T428,...
pos_nod_T428];

aa_RET = [pos_nod_E02_R,pos_nod_E03_R,pos_nod_F03_R,pos_nod_F05_R,...
pos_nod_F06_R,pos_nod_F20_R,pos_nod_G01_R,pos_nod_G02_R,pos_nod_G03_R,.
..
pos_nod_G04_R,pos_nod_G05_R,pos_nod_G07_R,pos_nod_G08_R,...
pos_nod_G08v_R,pos_nod_G09_R,pos_nod_G10_R,pos_nod_G11_R,...
pos_nod_G12_R,pos_nod_G13_R,pos_nod_G14_R,pos_nod_G15_R,...
pos_nod_G16_R,pos_nod_G17_R,pos_nod_G18_R,pos_nod_G19_R,...
pos_nod_G22_R,pos_nod_H03_R,pos_nod_H03_R,pos_nod_H03_R,pos_nod_H03_R,.
..
pos_nod_H03_R,pos_nod_H03_R,pos_nod_H10_R,pos_nod_T113e_R,...
pos_nod_T118_R,pos_nod_T120_R,pos_nod_T201_R,pos_nod_T201c_R,...
pos_nod_T201e_R,pos_nod_T201e_R,pos_nod_T203_R,pos_nod_T203_R,...
pos_nod_T205_R,pos_nod_T205_R,pos_nod_T205_R,pos_nod_T206_R,...
pos_nod_T206_R,pos_nod_T207_R,pos_nod_T207_R,pos_nod_T207_R,...
pos_nod_T207_R,pos_nod_T207_R,pos_nod_T211_R,pos_nod_T211c_R,...
pos_nod_T211c_R,pos_nod_T212_R,pos_nod_T214e_R,pos_nod_T216_R,...
pos_nod_T219s_R,pos_nod_T223_R,pos_nod_T226_R,pos_nod_T227_R,...
pos_nod_T228_R,pos_nod_T229_R,pos_nod_T301_R,pos_nod_T301c_R,...
pos_nod_T301c2_R,pos_nod_T302_R,pos_nod_T302_R,pos_nod_T428_R,...
pos_nod_T428_R,pos_nod_T428_R];
```

```
IDA_largo = length(aa_IDA);
RET_largo = length(aa_RET);
for i=1:IDA_largo
    aa_conteo_IDA(i) = length(find(aa_IDA==aa_IDA(i)));
% Se cuentan cuantas veces cada recorrido pasa por un mismo nodo
end
for i=1:RET_largo
    aa_conteo_RET(i) = length(find(aa_RET==aa_RET(i)));
% Se cuentan cuantas veces cada recorrido pasa por un mismo nodo
end

[aa_IDA_cont_ord,aa_Pos_ida]=sort(aa_conteo_IDA);
% se ordena de menor a mayor veces
[aa_RET_cont_ord,aa_Pos_ret]=sort(aa_conteo_RET);

aa_IDA_ORD = aa_IDA(aa_Pos_ida);
% ahora se tienen ordenados de menor a mayor los valores originales
aa_RET_ORD = aa_RET(aa_Pos_ret);

aa_IDA_Matriz = [aa_IDA_cont_ord; aa_IDA_ORD];
% se guardan las veces que pasa por un nodo y la posicion de ese nodo
aa_RET_Matriz = [aa_RET_cont_ord; aa_RET_ORD];

l=IDA_largo;
for j=1:l
if (j>=l)
    j=l;
end
   aux = find(aa_IDA_Matriz(2,:)==aa_IDA_Matriz(2,j));
% se borran las posiciones repetidas
   l_aux = length(aux);
   if (l_aux>1)
       for k=2:l_aux
           aa_IDA_Matriz = [aa_IDA_Matriz(:,(1:aux(k)-1...
-k+2)),aa_IDA_Matriz(:,(aux(k)+1-k+2:end))];
       end
   end
   aux=0;
   l=length(aa_IDA_Matriz(1,:));
end

l=RET_largo;
for j=1:l
if (j>=l)
    j=l;
end
   aux = find(aa_RET_Matriz(2,:)==aa_RET_Matriz(2,j));
% se borran las posiciones repetidas
   l_aux = length(aux);
   if (l_aux>1)
       for k=2:l_aux
           aa_RET_Matriz = [aa_RET_Matriz(:,(1:aux(k)-1...
-k+2)),aa_RET_Matriz(:,(aux(k)+1-k+2:end))];
```

```
       end
   end
   aux=0;
   l=length(aa_RET_Matriz(1,:));
end
r=500; % Rango cobertura RSU
aa_IDA_Matriz = fliplr(aa_IDA_Matriz);
aa_RET_Matriz = fliplr(aa_RET_Matriz);
l_ida = length(aa_IDA_Matriz);
l_ret = length(aa_RET_Matriz);
for i=1:l_ida-1
    j=i+1;
    while (j<=l_ida)
    d = sqrt((X_nod(aa_IDA_Matriz(2,i))...
 - X_nod(aa_IDA_Matriz(2,j)))^2+(Y_nod(aa_IDA_Matriz(2,i))...
 - Y_nod(aa_IDA_Matriz(2,j)))^2);
        if(d<=r)
           if (j~=l_ida)
           aa_IDA_Matriz = [aa_IDA_Matriz(:,(1:j...
-1)),aa_IDA_Matriz(:,(j+1:end))];
           end
           if (j==l_ida)
           aa_IDA_Matriz = [aa_IDA_Matriz(:,(1:j-1))];
           end
           j = j-1;
           l_ida = l_ida-1;
        end
        j=j+1;
    end
end

for i=1:l_ret-1
    j=i+1;
    while (j<=l_ret)
    d = sqrt((X_nod(aa_RET_Matriz(2,i))...
 - X_nod(aa_RET_Matriz(2,j)))^2+(Y_nod(aa_RET_Matriz(2,i))...
 - Y_nod(aa_RET_Matriz(2,j)))^2);
        if(d<=r)
           if (j~=l_ret)
           aa_RET_Matriz = [aa_RET_Matriz(:,(1:j...
-1)),aa_RET_Matriz(:,(j+1:end))];
           end
           if (j==l_ret)
           aa_RET_Matriz = [aa_RET_Matriz(:,(1:j-1))];
           end
           j = j-1;
           l_ret = l_ret-1;
        end
        j=j+1;
    end
end

l_ida = length(aa_IDA_Matriz);
```

```
l_ret = length(aa_RET_Matriz);

for i=1:l_ida
    j=1;
    while (j<=l_ret)
        d = sqrt((X_nod(aa_IDA_Matriz(2,i))...
 - X_nod(aa_RET_Matriz(2,j)))^2+(Y_nod(aa_IDA_Matriz(2,i))...
 - Y_nod(aa_RET_Matriz(2,j)))^2);
        if (d<=r)
% Se agregan los nodos de la vuelta cercanos a los de ida segun r =
%cobertura
            aa_IDA_Matriz(1,i) = aa_IDA_Matriz(1,i)+aa_RET_Matriz(1,j);
            if (j~=1 && j~=l_ret)
            aa_RET_Matriz = [aa_RET_Matriz(:,(1:j...
-1)),aa_RET_Matriz(:,(j+1:end))];
            end
            if (j==1)
            aa_RET_Matriz = [aa_RET_Matriz(:,(j+1:end))];
            end
            if (j==l_ret)
            aa_RET_Matriz = [aa_RET_Matriz(:,(1:j-1))];
            end
            j=j-1;
            l_ret = l_ret-1;
        end
        j=j+1;
    end
end

aa_NODOS = [aa_IDA_Matriz , aa_RET_Matriz];
[aa_cont_ord,aa_Pos]=sort(aa_NODOS(1,:));
aa_NODOS =  fliplr(aa_NODOS(:,aa_Pos));

X_RSU = X_nod(aa_NODOS(2,:))- min(X_nod);
Y_RSU = max(Y_nod) - Y_nod(aa_NODOS(2,:));

plot(X_nod,Y_nod,'+',X_nod(aa_NODOS(2,:)),Y_nod(aa_NODOS(2,:)),'*');
```

ANEXO C: MÉTODO: CALCULATEMETD( ) DE GEOPPSN.CC EN OMNET++

```
double GEOPPSN::calculateMETD(GEOPPSN_RM *routingMsg)
{
   double SpeedMetric = 2.78;   //10 km/h
   double BusSpeed = getSpeed();
   double RsuX = routingMsg->getDestPosX();
   double RsuY = routingMsg->getDestPosY();
   double BusX = getXPos();
   double BusY = getYPos();

   if(BusSpeed > 0 && BusSpeed < 42)
   {
      mobility = TraCIMobilityAccess().get();
      std::string BusId = mobility->getExternalId();

      const char *filename = par("BusRoutes");
      std::ifstream in(filename, std::ios::in);

      if (in.fail())
          opp_error("Cannot open file '%s'",filename);
      std::string line;
      std::string subline;
      std::string Route_Bus = " ";
      std::string substr = " ";
      std::string RealBusId = " ";
      std::string::size_type pos0 = BusId.find(".");


      while (std::getline(in, line))
        {
          subline = line;

          std::string::size_type pos1 = subline.find("id=");
          std::string::size_type pos2 = subline.find(" X=");

          substr = subline.substr(pos1+3,pos2-(pos1+3));
          RealBusId = BusId.substr(0,pos0);

          if (strcmp(RealBusId.c_str(), substr.c_str())==0)
             {
            Route_Bus = subline;
            break;
             }
        }
      ev << "Real Bus ID: " << RealBusId << "substr: " << substr <<
endl;
      std::string::size_type pos3 = Route_Bus.find("X=");
      std::string::size_type pos4 = Route_Bus.find("Y=");
      std::string X_route_str = Route_Bus.substr(pos3+2,pos4-(pos3+2));
```

```
        std::string Y_route_str = Route_Bus.substr(pos4+2);

        std::stringstream X_route(X_route_str);
        std::stringstream Y_route(Y_route_str);

        std::vector< double > X_rout;
        std::vector< double > Y_rout;
        double dx = 0.0;
        double dy = 0.0;

        while (X_route >> dx)
           {
               X_rout.push_back (dx);
           }
        while (Y_route >> dy)
           {
               Y_rout.push_back (dy);
           }

        in.close();
        // Identify where is the Bus
        int indice = -1.0;
        double cos_theta = 1.0;
        double Best_cos = 1.0;
        double* X_rou; double* Y_rou;
        X_rou = new double [X_rout.size()];
        Y_rou = new double [Y_rout.size()];
        copy( X_rout.begin(), X_rout.end(), X_rou);
        copy( Y_rout.begin(), Y_rout.end(), Y_rou);
        int sz = X_rout.size();

        for (int i=0; i<sz-1; i++)
        {
           cos_theta = ((BusX-X_rou[i])*(BusX-X_rou[i+1])+(BusY-
Y_rou[i])*(BusY-Y_rou[i+1]))/(pow(pow((BusX-X_rou[i]),2)+pow((BusY-
Y_rou[i]),2),0.5)*pow(pow((BusX-X_rou[i+1]),2)+pow((BusY-
Y_rou[i+1]),2),0.5));

           if (cos_theta < Best_cos)
              {
                    Best_cos = cos_theta;
                    indice = i;
              }
        }
        // Calculate NP: Nearest Point
        double Best_dist = 100000.0;
        double NPx = -1.0;
        double NPy = -1.0;
        double indice_NP = -1.0;
        double alpha;
        double dist;

        for (int i=indice; i<sz-1; i++)
```

```
      {
           if (i == indice)
              {
                 // Alpha = p.u/abs(u)^2, where p is the vector to the
Rsu and u the segment, proyection of p in u
                  alpha = ((RsuX-BusX)*(X_rou[i+1]-BusX)+(RsuY-
BusY)*(Y_rou[i+1]-BusY))/(pow(X_rou[i+1]-BusX,2)+pow(Y_rou[i+1]-
BusY,2));
            if (alpha<0)
              {
                 dist = pow(pow(RsuX-BusX,2)+pow(RsuY-BusY,2),0.5);
// The point is to the left of the segment
                 if (dist<Best_dist)
                    {
                          Best_dist = dist; NPx = BusX; NPy = BusY;
indice_NP = -1;
                    }
              }
            else if (alpha>1)
              {
                 dist = pow(pow(RsuX-X_rou[i+1],2)+pow(RsuY-
Y_rou[i+1],2),0.5);  // The point is to the right of the segment
                 if (dist<Best_dist)
                    {
                          Best_dist = dist; NPx = X_rou[i+1]; NPy =
Y_rou[i+1]; indice_NP = i+1;
                    }
              }
            else
              {
                 dist = abs((X_rou[i+1]-BusX)*(RsuY-BusY)-(Y_rou[i+1]-
BusY)*(RsuX-BusX))/pow((pow(X_rou[i+1]-BusX,2)+pow(Y_rou[i+1]-
BusY,2)),0.5);  // The point is between the segment, dist = u X
p/abs(u)
                 if (dist<Best_dist)
                    {
                          Best_dist = dist;
                          double aux_X = X_rou[i+1]-BusX;
                          double aux_Y = Y_rou[i+1]-BusY;
                          double aux_u = ((RsuX-BusX)*aux_X + (RsuY-
BusY)*aux_Y)/(pow(aux_X,2)+pow(aux_Y,2)); // p.u/abs(u)^2
                          NPx = BusX + aux_u*aux_X;
                          NPy = BusY + aux_u*aux_Y;
                          indice_NP = -1;
                    }
              }
            }
          else
              {
                 // Alpha = p.u/abs(u)^2, where p is the vector to the
Rsu and u the segment
```

```
                alpha = ((RsuX-X_rou[i])*(X_rou[i+1]-X_rou[i])+(RsuY-
Y_rou[i])*(Y_rou[i+1]-Y_rou[i]))/(pow(X_rou[i+1]-
X_rou[i],2)+pow(Y_rou[i+1]-Y_rou[i],2));
            if (alpha<0)
               {
               dist = pow(pow(RsuX-X_rou[i],2)+pow(RsuY-
Y_rou[i],2),0.5);  // The point is to the left of the segment
               if (dist<Best_dist)
                  {
                        Best_dist = dist; NPx = X_rou[i]; NPy =
Y_rou[i]; indice_NP = i;
                  }
               }
            else if (alpha>1)
               {
               dist = pow(pow(RsuX-X_rou[i+1],2)+pow(RsuY-
Y_rou[i+1],2),0.5);  // The point is to the right of the segment
               if (dist<Best_dist)
                  {
                        Best_dist = dist; NPx = X_rou[i+1]; NPy =
Y_rou[i+1]; indice_NP = i+1;
                  }
               }
            else
               {
               dist = abs((X_rou[i+1]-X_rou[i])*(RsuY-Y_rou[i])-
(Y_rou[i+1]-Y_rou[i])*(RsuX-X_rou[i]))/pow((pow(X_rou[i+1]-
X_rou[i],2)+pow(Y_rou[i+1]-Y_rou[i],2)),0.5);  // The point is between
the segment, dist = u X p/abs(u)
               if (dist<Best_dist)
                  {
                        Best_dist = dist;
                        double aux_X = X_rou[i+1]-X_rou[i];
                        double aux_Y = Y_rou[i+1]-Y_rou[i];
                        double aux_u = ((RsuX-X_rou[i])*aux_X + (RsuY-
Y_rou[i])*aux_Y)/(pow(aux_X,2)+pow(aux_Y,2)); // p.u/ abs(u)^2
                        NPx = X_rou[i] + aux_u*aux_X;
                        NPy = Y_rou[i] + aux_u*aux_Y;
                        indice_NP = i;
                  }
               }
            }
      }

      // Calculate Distance to NP: Nearest Point
      dist =  pow(pow(BusX-X_rou[indice+1],2)+pow(BusY-
Y_rou[indice+1],2),0.5);
      if (indice_NP == -1)
         {
               dist = pow(pow(BusX-NPx,2)+pow(BusY-NPy,2),0.5);
         }
      else
         {
```

```
            for (int i=indice+1; i<=indice_NP; i++)
                {
                   if (i == indice_NP)
                      {
                           dist = dist + pow(pow(X_rou[i]-
NPx,2)+pow(Y_rou[i]-NPy,2),0.5);
                      }
                   else
                      {
                           dist = dist + pow(pow(X_rou[i]-
X_rou[i+1],2)+pow(Y_rou[i]-Y_rou[i+1],2),0.5);
                      }
                }
         }

      //              Distance to NP                 Distance NP to D
      //   METD = ------------------------   +   --------------------
      //                BusSpeed                        SpeedMetric

      double METD = dist/BusSpeed + Best_dist/SpeedMetric;

      ev << " dist:" << dist << " Best_dist:" << Best_dist <<  " NPx:"
<< NPx <<  " NPy:" << NPy << " indice_NP:" << indice_NP << " indice:"
<< indice << " alpha:" << alpha << endl;
      double dis_tot = pow(pow(RsuY-BusY,2)+pow(RsuX-BusX,2),0.5) ;
      in.close();
      ev << " BusId:" << BusId << " Distancia:" << dis_tot << " METD:"
<< METD <<  " Velocity:" << BusSpeed <<  " BusX:" << BusX << " BusY:"
<< BusY << " RsuX:" << RsuX << " RsuY:" << RsuY << endl;
      return METD;
   }
   else
   {
        double dis_tot = pow(pow(RsuY-BusY,2)+pow(RsuX-BusX,2),0.5) ;
        double METD = dis_tot/SpeedMetric;
        return METD;
   }
}
```

ANEXO D: MÉTODO: CHANGEDESTADDRTO( ) DE GEOPPSN_DATAQUEUE.CC EN OMNET++

```
void GEOPPSN_DataQueue::changeDestAddrTo(unsigned int myAddr, IPAddress
realDestAddr, IPAddress interDestAddr, int prefix)
{
    ev << "Changing destiny address in Queue from " << realDestAddr <<
" to " << interDestAddr << endl;
    bool tryAgain = true;
    while (tryAgain)
    {
      tryAgain = false;
        for (std::list<GEOPPSN_QueuedData>::iterator iter =
dataQueue.begin(); iter != dataQueue.end(); iter++)
        {
            GEOPPSN_QueuedData qd = *iter;
            if (qd.destAddr.prefixMatches(realDestAddr, prefix))
            {
                qd.datagram->setDestAddress(interDestAddr);
                qd.datagram->setInterDestAddress(realDestAddr);

dataQueue.push_back(GEOPPSN_QueuedData(const_cast<IPDatagram *>
(qd.datagram), interDestAddr));
                    dataQueue.erase(iter);
                tryAgain = true;
                break;
            }
        }
     }
}
```