ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“DISEÑO E IMPLEMENTACIÓN DE UN MECANISMO PARA PROTEGER LA EQUIDAD DEL MERCADO EN UN SISTEMA DE AUTOMÓVIL COMPARTIDO OCULTANDO OFERTAS Y MANTENIENDO LA CAPACIDAD DE DETERMINAR LA COTIZACIÓN MÁS BAJA USANDO CIFRADO HOMOMÓRFICO EN UNA RED OPORTUNISTA"

TRABAJO DE TITULACIÓN

Previo a la obtención del Título de:

MAGISTER EN SEGURIDAD INFORMÁTICA APLICADA

Presentado por

Ing. Miguel Eduardo Carpio Miranda

Guayaquil – Ecuador

2017

AGRADECIMIENTO

A mi tutor, PhD. Sergi Robles, a su grupo de investigación SeNDA y a mi profesor de criptografía PhD. Jordi Herrera, por su valiosa colaboración y predisposición para transmitir sus conocimientos y experiencia, además de darme la oportunidad de relacionarme y aportar con este proyecto, en las áreas de investigación de la seguridad informática correspondientes a la criptografía homomórfica y Redes oportunistas.

DEDICATORIA

A mis abuelos, Rosa Rada y Vicente Carpio por el apoyo y la confianza incondicional brindada en todas las etapas de mi vida que compartí con ellos, siempre fueron y serán el motivo de mi superación personal y profesional. A mis padres por el esfuerzo y sacrificio realizado para que logre mi superación académica.

A mi amada esposa Alejandra Cárdenas y a mi hija Arianna Carpio quienes llenan mis días de felicidad, y que con amor y paciencia apoyan y resguardan mis proyectos académicos y profesionales.

TRIBUNAL DE SUSTENTACIÓN

_____
Ing. Lenin Freire

DIRECTOR MSIA

_____
Ing. Lenin Freire

CO-TUTOR DEL TRABAJO DE TITULACIÓN

_____
Mgs. Karina Astudillo

MIEMBRO DEL TRIBUNAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

.....

Miguel Eduardo Carpio Miranda

RESUMEN

La criptografía homomórfica y las Redes tolerantes a retardos oportunistas son dos áreas de investigación totalmente distintas que se pueden complementar para diseñar e implementar sistemas seguros y dinámicos. En el capítulo 1 presentamos antecedentes históricos y actuales sobre los sistemas de automóvil compartido, así como el cifrado homomórfico y las Redes oportunistas. Para luego exponer el objetivo de implementar estas técnicas de seguridad al posible problema del aumento de precios de viajes por consenso de los conductores en los sistemas de automóvil compartido. Por lo tanto en este capítulo muestro la metodología para el diseño e implementación de un mecanismo de protección equidad de mercado, MPEM, para sistemas de automóvil compartido basado en criptografía homomórfica sobre una Red oportunista.

En el capítulo 2 se revisa el estado de arte del proyecto donde se expone documentación de técnicas implementadas y relacionadas con las Redes

tolerantes a retardos y el cifrado homomórfico. Se estudia a las Redes tolerantes a retardos y a los criptosistemas homomórficos, así como las actuales y posibles aplicaciones de ambas técnicas de seguridad de información.

En el capítulo 3 se obtiene información de los sistemas de automóviles compartidos como su historia en el mundo y actual uso en Ecuador. Luego se revisan los mapas, rutas y cotizaciones que nos servirán para el diseño y simulación del MPEM, así como las herramientas para la simulación de Redes oportunistas y librerías para el desarrollo de aplicaciones con funciones que soporten criptografía homomórfica.

En el capítulo 4 se analizan los protocolos de Redes tolerantes a retardos oportunistas con los cuales se simula el esquema de intercambio de mensajes del MPEM. También se somete a análisis el esquema de cifrado homomórfico para determinar las librerías que son óptimas para el desarrollo e

implementación del MPEM. En base a los análisis mencionados se procede con el diseño del esquema distribuido del MPEM basado en una Red oportunista y el diseño de las aplicaciones del MPEM basadas en criptografía homomórfica, casos de uso y flujo de procesos.

En el capítulo 5 se realiza las configuraciones de las simulaciones basadas en el esquema de intercambio de mensajes del MPEM, protocolos de enrutamiento para Redes tolerantes a retardos y por el tamaño de comunidades de usuarios. Para luego simular y obtener datos estadísticos de entregas satisfactorias de mensajes y rendimiento de la Red oportunista. Por otro lado se desarrolla las aplicaciones para el conductor y pasajero en base a funciones con criptografía homomórfica, casos de uso, diagramas de flujo de procesos y el esquema de intercambio de mensajes del MPEM. Finalmente ejecutamos las aplicaciones del MPEM y comprobamos su funcionamiento.

En el capítulo 6 se analizan los resultados obtenidos de la simulación de la Red oportunista y se determina el protocolo óptimo y recomendado para la Red distribuida en base a los análisis de las estadísticas obtenidas al simular el esquema de intercambio de mensajes del MPEM y sus casos de usos en diferentes tamaños de comunidades de usuarios. Luego se analiza la ejecución de las aplicaciones del MPEM en busca de comprobar la precisión de los resultados de operaciones realizadas con cifrado homomórfico y la factibilidad de uso de estos criptosistemas para aplicaciones reales.

ÍNDICE GENERAL

ABREVIATURAS Y SIMBOLOGÍA

BGV : Brakerski, Gentry y Vaikuntanathan

CCSDS : Comité Consultivo de Sistemas de Datos Espaciales

DARPA : Agencia de Proyectos de Investigación Avanzada de Defensa

DINET : Red de impacto profundo

DTN : Red tolerante a retardos

DTNRG : Grupo de investigación de Redes tolerantes a retardos

ESP : Agencia Espacial Europea

FHE : Cifrado totalmente homomórfico

GIS : Sistema de Información Geográfica

GNU GPL : Licencia Pública General de GNU

HEAT : Aplicaciones y tecnología de cifrado homomórfico

IPN : Internet Interplanetaria

IRTF : Grupo de Trabajo de Investigación de la Internet

MANET : Redes Ad-Hoc móviles

MMLAB : Laboratorio de Multimedia y Comunicaciones Móviles

MPEM : Mecanismo de protección de la equidad del mercado

N4C : North Central Camera Club Council

NASA : Administración Nacional de la Aeronáutica y del Espacio

NS2 : El simulador de Red

OppNets : Redes oportunistas

RFC : Solicitud de comentarios

RSA : Rivest, Shamir y Adleman

RTT : Tiempo de ida y vuelta

SeNDA : Seguridad de las Redes y aplicaciones distribuidas

SNC : Conectividad de Red Sámi

SPINDLE : Sobrevivencia de políticas influenciadas en Redes disruptivas y tolerantes a través del aprendizaje y evolución.

The ONE : Ambiente de Redes oportunistas

THEP : El Proyecto de Cifrado Homomórfico

TIER : Tecnología e infraestructura para regiones en desarrollo

UAV : Vehículo aéreo no tripulado

WEB : World Wide Web

WKT : Texto conocido

WMN : Redes inalámbricas en malla

WSN : Sensores de Redes inalámbricas

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCIÓN

En las tecnologías de información y comunicación, la seguridad informática comprende varios campos de estudio, experimentación y mejoras, que cubren desde el desarrollo de sistemas y aplicaciones seguras hasta la protección de la integridad, confiabilidad y disponibilidad de la Red de datos. La necesidad de comunicación y privacidad se torna esencial para mantener el estilo de vida y relaciones sociales de los internautas, instituciones y gobiernos. Por lo que la demanda de nuevos diseños y esquemas seguros alientan e incitan a la investigación de nuevas técnicas de seguridad de información y comunicación.

Este proyecto tiene la finalidad de aplicar técnicas de seguridad informática que se encuentran en investigación, en un sistema sencillo y útil para la sociedad como lo es un mecanismo de protección de equidad de mercado para sistemas de automóvil compartido basado en criptografía homomórfica sobre Redes oportunistas. Estas técnicas mencionadas son producto de las áreas de investigación de la criptografía homomórfica en el desarrollo de sistemas y

operaciones totalmente cifradas; y las Redes tolerantes a retardos oportunistas las cuales permiten la privacidad de la ubicación de los usuarios y la comunicación en ambientes hostiles, intermitentes y de enlaces dinámicos.

CAPÍTULO 1

1. GENERALIDADES

El trabajo de titulación tiene la finalidad de usar criptografía homomórfica y Redes oportunistas para el diseño e implementación de un mecanismo de protección de la equidad del mercado (MPEM) para un sistema de automóvil compartido. Manteniendo la privacidad en las operaciones que calculan los valores de los viajes más económicos y mejorando la seguridad de los usuarios protegiendo su ubicación.

1.1. Antecedentes

Ecuador es un país con una vasta variedad de culturas y tradiciones e inigualable riqueza natural. Posee cuatro regiones Costa, Sierra, Oriente e Insular las cuales ofrecen un sin número de atracciones

turísticas las cuales son visitadas por nacionales y extranjeros. Durante los feriados como carnaval, semana santa y fin de año los terminales terrestres se saturan con turistas que buscan diversión en las playas o diversidad cultural en la Sierra y Amazonía. En el feriado del 24 de Mayo del 2016 se realizaron 505.000 viajes a diferentes destinos del Ecuador, incluidas la región Andina, Amazónica e Insular. La Sierra recibió a 216.000 visitantes, la Amazonía 69.000 y Galápagos 2.000 personas [1], y solo en el Feriado de Carnaval del 2015 el ministerio de turismo cuantificó a 1.278.056 viajes realizados por turistas y excursionistas nacionales y extranjeros [2].

Con un sistema de automóvil compartido los turistas o personas en general que pretendan viajar a un punto en común pueden reservar un puesto en un automóvil cuyo conductor viaje al mismo punto o pase por él y tenga la disponibilidad de compartir el viaje con el fin de reducir gastos, de esta forma tanto los pasajeros como el conductor se benefician ahorrando costos de viaje o evitando largas filas y mal tiempo en terminales saturados en el caso de los pasajeros.

Este sistema no es solo aplicable para viajes interprovinciales, también es aplicable para recorridos dentro de una ciudad, como en el caso del

Distrito Metropolitano de Quito el cual luego de la implementación de la medida de restricción a la circulación vehicular “pico y placa” en Mayo del 2010 (que regula el tránsito prohibiendo la circulación vehicular por unas horas cada día, de acuerdo con el último número de su placa, en un área determinada de Quito) se levantó un requerimiento de análisis hacia el Municipio de Quito en Agosto del 2014 para promocionar el auto compartido (Carpooling), con la finalidad de optimizar el uso de los vehículos privados y de la Red vial, exonerando esta práctica de la medida pico y placa a los autos compartidos. Con lo cual la secretaría de movilidad en base a un informe técnico plantea como objetivo ser más eficiente en la utilización de la Red vial al movilizar más personas en los vehículos livianos al elevar su índice de ocupación, disminuir la utilización innecesaria de vehículos particulares, incrementar y mejorar las relaciones sociales y elevar el nivel de productivo de la ciudad [3].

Según las estadísticas de la Secretaría de Movilidad del Municipio de Quito de cada 10 carros que se trasladan de Cumbayá a Quito, solo dos autos llevan más de dos personas. En base a este contexto nació Socialcar en Noviembre del 2013, una aplicación virtual en donde los cibernautas ponen a disposición de otros usuarios sus rutas de traslado con el fin de compartir su vehículo [4]. La Red en línea para

compartir vehículos actualmente no se encuentra disponible, su plataforma web fue www.socialcar.ec y poseía disponibilidad para sistemas Android e IOS.

El uso del sistema de automóvil compartido data de la Segunda Guerra Mundial en Estados Unidos donde el automóvil compartido fue una técnica para el racionamiento de recursos [5]. En Europa, el auto compartido se ha vuelto cada vez más popular en los últimos años, gracias a la alemana Mitfahrgelegenheit (carpooling.com), y la francesa BlaBlaCar [6]. De acuerdo con sus respectivos sitios web, a principios de 2015 estas plataformas en línea cuentan con más de 6 millones de dólares y 10 millones de usuarios [7], respectivamente, en toda Europa y más allá.

Ninguno de estos sistemas mencionados posee un mecanismo de protección de equidad de mercado que prive a los conductores elevar precios de los viajes debido a la influencia de los precios de otros conductores. Para mantener la privacidad de las cotizaciones planteamos el uso de criptografía homomórfica que mantendrá los precios de viajes cifrados, privados y confidenciales en todo el proceso de determinar el precio más bajo. Además proponemos el uso

de Redes oportunistas para obtener un sistema descentralizado donde los nodos se puedan comunicar directamente entre ellos siendo tolerantes a retardos, posibilitando la comunicación incluso cuando no existan rutas de conexión entre ellos, y garantizar que los usuarios no puedan ser localizados por otros usuarios, preservando así la privacidad de su posicionamiento.

Con la criptografía homomórfica es posible analizar datos sin descifrarlos. La clave está en cifrarlos de tal manera que la realización de una operación matemática con la información cifrada y el posterior descifrado del resultado produzca la misma respuesta que se obtendría tras una operación similar con los datos sin cifrar. La correspondencia entre las operaciones con los datos sin cifrar y las operaciones que se realizarán con los datos cifrados se conoce como homomorfismo [8].

En 2009, Riggio y Sicari dieron a conocer una aplicación práctica de cifrado homomórfico para una Red de comunicaciones inalámbrica de arquitectura híbrida basada en sensores de Redes inalámbricas (wireless sensor network, WSN) y Redes inalámbricas en malla (wireless mesh network, WMN). El sistema permite la autentificación

de los nodos y múltiples envíos de datos cifrados entre nodos, con la capacidad de realizar análisis estadísticos de diferentes parámetros (temperatura, humedad, etc.) procedentes de una WSN. [9]

En la última sesión de Eurocrypt 2010, Craig Gentry y Shai Halevi presentaron una implementación funcional de un sistema de cifrado completamente homomórfico junto con las estadísticas de rendimiento [10]. En abril de 2013, IBM liberó en GitHub la primera versión de HElib [11], una librería que implementa el sistema de cifrado homomórfico de Brakerski, Gentry y Vaikuntanathan (BGV) [12], escrita en lenguaje C++ y bajo licencia GNU GPL [13].

Por otro lado, una Red oportunista es un tipo de Red que resuelve desafíos de conexión, donde los contactos de la Red son intermitentes o cuando el rendimiento del enlace es muy variable o extremo. En una Red de este tipo, no existe una ruta completa desde el origen al destino la mayor parte del tiempo. Además, el camino puede ser muy inestable y puede cambiar o romper rápidamente. Por lo tanto, con el fin de hacer posible la comunicación en una Red Oportunista, los nodos intermedios pueden tomar la custodia de los datos durante el apagón para luego transmitirlos cuando la conectividad vuelve a habilitarse

[14], así sucesivamente a la espera de un futuro contacto oportunista con otro nodo para reenviar la información hasta llegar a su destino.

La aplicación de las Redes oportunistas usualmente se enfoca en áreas rurales donde la población se encuentra separada por grandes distancias y poseen ausencia de Redes de datos. Las Redes tolerantes a retardos (Delay Tolerant Networking, DTN) están diseñadas para trabajar sobre estos desafíos de conectividad, y poseen varios protocolos de enrutamiento aplicables. PrivHab+ es un protocolo de geo-enrutamiento seguro para DTN que trabaja sobre Redes oportunistas y utiliza cifrado homomórfico para determinar la mejor ruta hacia el siguiente nodo. Este protocolo de geo-enrutamiento seguro compara la ubicación geográfica de los nodos candidatos y determina la mejor opción para llegar a destino, todo este proceso de elección lo realiza de manera cifrada utilizando criptografía homomórfica de Paillier, manteniendo así la privacidad de localización de los nodos [15]. PrivHab+ es un buen ejemplo de uso de cifrado homomórfico sobre una Red oportunista donde se asegura la privacidad de la ubicación de los nodos.

1.2. Descripción del problema

En un esquema de automóvil compartido, como blablacar.com, los pasajeros y conductores intercambian solicitudes y precios. Un pasajero puede reservar fácilmente su plaza en Internet y viajar más barato con toda confianza, debe indicar la ciudad de salida y de llegada, la fecha en la que va a viajar y escoger entre todos los conductores que han publicado sus viajes, luego reserva su plaza con tarjeta mediante un pago seguro y recibe un código de reserva que posteriormente durante el trayecto debe entregar al conductor para que pueda recibir su aportación por el viaje realizado. El objetivo del esquema de automóvil compartido es de carácter económico y social sin ánimo de lucro, reduciendo los gastos del conductor compartiendo el viaje con pasajeros [16].

Desafortunadamente, las cotizaciones de los conductores son generalmente influenciados por los precios de otro conductor para solicitudes de viajes similares que resultan en una tendencia de mercado que favorece a los conductores con cotizaciones costosas y perjudiciales para los pasajeros.

Por lo tanto, la falta de equidad de precios se debe a que las cotizaciones de viajes de los conductores pueden ser vistas por todos los usuarios del sistema incluyendo a otros conductores y los métodos de cifrado como criptografía de llave pública y simétrica no poseen la capacidad de mantener la privacidad de los datos en todos los procesos del esquema los cuales implican operaciones matemáticas que demandan privacidad de entradas y salidas. Además de que en un sistema de automóvil compartido por ejemplo Blablacar se utiliza un esquema centralizado donde la escalabilidad de los usuarios se ve comprometida por la disponibilidad de los dispositivos centrales y enrutamiento sin retardos. De ahí que los consumidores se ven perjudicados por la falta de equidad de precios y disponibilidad del sistema.

1.3. Solución propuesta

Blablacar es un esquema centralizado de un sistema de automóvil compartido basado en Web. Aquí proponemos una alternativa distribuida basada en una Red oportunista que permitirá la conexión directa entre dispositivos y utilizará un esquema basado en criptografía homomórfica para garantizar la equidad del mercado en un escenario de sistema de automóvil compartido. En este proyecto se diseñará un

mecanismo de protección de la equidad del mercado (MPEM) basado en criptografía homomórfica sobre una Red oportunista para ocultar ofertas manteniendo la capacidad de determinar qué cotización es la más baja.

Se usará un esquema distribuido como una Red oportunista para implementar un mecanismo descentralizado donde los dispositivos se conectarán entre ellos permitiendo el envío de mensaje entre nodos usando algoritmos de enrutamiento tolerantes a retardos (Delay Tolerant Network, DTN). Podremos comprobar la escalabilidad de nuestro mecanismo en comunidades grandes de usuarios y trabajar con un protocolo de geo-enrutamiento seguro para proteger la privacidad de la localización de los usuarios.

Para comprobar nuestra propuesta se simulará nuestro esquema distribuido con el simulador The ONE, capaz de generar movimientos de nodos utilizando diferentes modelos de movimientos, emulando una Red oportunista donde utilizaremos algoritmos de enrutamiento DTN (Delay Tolerant Network) para el enrutamiento de mensajes entre nodos [17], protocolos de geo-enrutamiento seguro que nos permitirá la protección de la privacidad de la localización de los usuarios y un

protocolo criptográfico que implica el cifrado homomórfico para poder ocultar ofertas y determinar la más baja.

En una Red oportunista los nodos son móviles y fijos, la comunicación es posible incluso cuando no existan rutas de conexión entre los nodos. Las rutas son levantadas dinámicamente, no existe una ruta predeterminada, para el siguiente salto un nodo será oportunistamente escogido solo si es capaz enviar el mensaje a su destino final, cuando no exista un envío oportunista, el nodo guarda el mensaje y espera para un futuro contacto oportunista con otro nodo para reenviar la información [18].

Se usará un protocolo criptográfico que implica cifrado homomórfico para proporcionar equidad de precio utilizando y beneficiándonos de la capacidad de la criptografía homomórfica de cifrar las ofertas de los conductores y realizar cálculos sobre datos cifrados para determinar el precio más bajo permitiendo la confidencialidad y privacidad de las cotizaciones, operaciones y resultados en todos los procesos del mecanismo de protección de equidad del mercado, entregando solo a los consumidores el valor más bajo de forma descifrada [19]. Por lo tanto, como beneficio, los conductores no estarán en la capacidad de

saber a través del sistema las cotizaciones de los viajes de otros conductores para así elevar los precios.

Nuestro esquema de cifrado pertenece a una aplicación de cifrado homomórfico denominado cómputo multipartidista, esquema donde las partes interesadas en cómputo común utilizan funciones públicas manteniendo la privacidad de las entradas, por lo tanto la función que debe ser calculada es de conocimiento público, mientras que las entradas son datos cifrados y privados [20].

Para la implementación de criptografía homomórfica utilizaremos el cripto-sistema parcialmente homomórfico de Paillier. En un cripto-sistema de Paillier, si la llave pública es el módulo m y la base g, entonces el cifrado de un mensaje x se expresa como [21]:

$$\mathcal{E}(x) = g^x r^m \bmod m^2 \tag{1.1}$$

Para algún r aleatorio,

$$r \in \{0, \ldots, m-1\} \tag{1.2}$$

Las propiedades homomórficas son las siguientes:

Suma: E(x1)*E(x2) = Eh(x1+x2) (1.3)

Multiplicación: $E(x1)\ ^{x2} = E(x1*x2)$ (1.4)

La implementación del cripto-sistema de Paillier se la realizará usando librerías en JAVA con licencia GPL o de código abierto de tal forma que podamos utilizarlo, modificarlo y adaptarlo a lo que demande nuestro mecanismo de protección de equidad del mercado. Estas librerías las podemos encontrar en los proyectos “The Homomorphic Encryption Project - thep” dentro del repositorio de Google Code [22], el proyecto “NICTA/javallier” dentro del repositorio de GitHub [23] y otros proyectos de los cuales se analizarán las librerías más eficientes o combinación de éstas para ser utilizadas en nuestra propuesta.

1.4. Objetivo General

Diseñar e Implementar un mecanismo para proteger la equidad del mercado en un sistema de automóvil compartido ocultando ofertas y manteniendo la capacidad de determinar la cotización más baja usando cifrado homomórfico en una Red oportunista.

1.5. Objetivos Específicos

- Conocer a las Redes oportunistas, el cifrado homomórfico y cripto-sistemas homomórficos aplicables, así como el estado del arte para compilar soluciones similares o alternativas antes realizadas con Redes oportunistas y cifrado homomórfico.

- Conocer cómo funcionan los sistemas de automóvil compartido y obtener información de mapas, rutas, cotizaciones y herramientas que utilizaremos para la simulación de una Red oportunista e implementación de criptografía homomórfica que conforman nuestro mecanismo de protección de equidad del mercado.

- Diseñar un esquema distribuido basado en una Red oportunista que implica el uso de algoritmos de enrutamiento DTN (Delay Tolerant Network) para luego analizar y diseñar un esquema basado en criptografía homomórfica en una Red oportunista para mantener los precios en un sistema de automóvil compartido descentralizado mediante el cifrado de los datos y operaciones que determinan las mejores ofertas.

- Simular nuestro esquema distribuido con el simulador The ONE para comprobar que cumple con los requisitos de análisis y escalamiento en comunidades grandes de usuarios e implementar nuestro esquema basado en criptografía homomórfica con el cripto-sistema parcialmente homomórfico de Paillier para verificar la viabilidad técnica de la propuesta.

- Analizar e interpretar la simulación de nuestra Red oportunista e implementación del cifrado homomórfico para luego presentar los resultados de la implementación de nuestro mecanismo de protección de equidad del mercado.

1.6. Metodología

El mecanismo de protección de equidad del mercado para un sistema de automóvil compartido estará constituido por un protocolo criptográfico que implica cifrado homomórfico sobre una Red tolerante a retardo DTN - Red oportunista.

La Red oportunista que conformará nuestro esquema distribuido será simulada por The ONE un simulador capaz de generar movimientos

de nodos utilizando diferentes modelos de movimientos, emulando una Red oportunista donde utilizaremos algoritmos de enrutamiento DTN como Epidemic, PRoPHET y Spray and Wait que nos permitirán la protección de la privacidad de la localización de los usuarios. Los resultados estadísticos que produzca la simulación nos servirá para determinar la factibilidad de usar una Red oportunista, el protocolo de enrutamiento tolerante a retardo con mejor desempeño, probabilidad de recepción y respuesta, la escalabilidad en comunidades grandes de usuarios y la viabilidad técnica de la propuesta. Luego de simular nuestra Red oportunista y comprobar su funcionamiento procederemos con la implementación de un protocolo criptográfico utilizando el criptosistema homomórfico de Paillier donde utilizaremos las propiedades homomórficas del criptosistema para darle al pasajero la opción de pagar por lo que crea justo por viaje.

El criptosistema homomórfico se implementará usando librerías del programa Habcast, programa implementado en PrivHab+ (protocolo de geo-enrutamiento seguro para DTN) para hacer uso del criptosistema homomórfico de Paillier en operaciones matemáticas de suma, resta y multiplicación. El protocolo criptográfico está definido por los siguientes algoritmos:

Algoritmo 1. La información de los viajes lo enviará el conductor mediante un mensaje a un grupo de pasajeros con información del precio cifrado, fecha, lugar de origen, lugar de destino, id del conductor, número de plazas y la clave pública de Paillier. Cuando un pasajero recepta el mensaje este responde con su identificación, número de teléfono y la operación cifrada de su límite a pagar menos el precio ofertado por el conductor. Por lo tanto, ni el pasajero puede saber cuánto ofertó el conductor, ni el conductor una vez que reciba el mensaje puede saber cuánto fue el límite a pagar por el pasajero. Si la resta cifrada homomórficamente da como resultado un número entero positivo, el conductor perderá una plaza en su automóvil y deberá llamar al pasajero para indicarle que su límite a pagar está dentro de lo ofertado.

Algoritmo 2. La información de los viajes lo enviará el pasajero mediante un mensaje a un grupo de conductores con información del precio límite a pagar cifrado, fecha, lugar de origen, lugar de destino, id del pasajero, y la clave pública de Paillier. Cuando un conductor recepta el mensaje este responde con su identificación, número de teléfono y la operación cifrada del límite a pagar menos el precio ofertado por el conductor. Por lo tanto, ni el pasajero puede saber cuánto ofertó el conductor, ni el conductor una vez que reciba el

mensaje puede saber cuánto fue el límite a pagar por el pasajero. Si la resta cifrada homomórficamente da como resultado un número entero positivo, este número se agregará a una lista cotizaciones de otros conductores, de la cual se determinará cual es el valor más bajo para que luego el pasajero llame al conductor con el viaje más económico para informarle que desea viajar con él.

Finalmente, procederemos a analizar la información obtenida de la simulación de la Red oportunista conjunto a los resultados de las pruebas de ejecución del protocolo criptográfico basado en el criptosistema homomórfico de Paillier para conformar y consolidar nuestro Mecanismo de protección de equidad del mercado, MPEM. Finalmente presentaremos los resultados de la implementación de nuestro MPEM.

CAPÍTULO 2

2. MARCO TEÓRICO

A medida que la tecnología avanza, la infraestructura tecnológica es más sofisticada, y nuevos restos de comunicación aparecen, un nuevo desafío surge, la seguridad de la información, que hoy en día cumple un rol importante debido al constante descubrimiento de brechas en los sistemas que ponen en riesgo la confiabilidad, disponibilidad e integridad de la información, por lo que surge la tendencia y necesidad del estudio y aplicación de la seguridad de la información de tal forma que existe el esfuerzo de crear productos seguros en nuevos diseños y arquitecturas de soluciones informáticas. Con el estudio de Redes oportunistas podemos tomar el desafío de comunicarnos en zonas con pocos recursos de conexión generando sistemas distribuidos y seguros, además, mediante el

cifrado homomórfico podríamos alcanzar en un futuro el desarrollar programas totalmente cifrados utilizando datos cifrados en entradas y salidas, y en todo el procesamiento, entregando confidencialidad total para los datos del cliente, idea que ilusiona a los servicios en la nube.

2.1. Estado del Arte

Las Redes tolerantes a retardo o DTN por sus siglas en inglés, son Redes que desafían la conectividad intermitente, retardos largos y variables, velocidades de datos asimétricos y altos índices de error [24]. Por ello su aplicación se estudia en varios campos como Redes espaciales, Redes de táctica militar, y Redes para diversos desafíos comunitarios [25]. A continuación revisaremos algunos proyectos sobre Redes DTN de los cuales sobresalen las siguientes investigaciones y aplicaciones:

El grupo DTNRG (IRTF). Por sus siglas en inglés (The Delay-Tolerant Networking Research Group) es un grupo caracterizado como parte del Internet Research Task Force (IRTF), este grupo se preocupa de la interconexión de Redes muy heterogéneas entre sí, incluso si la conectividad de extremo a extremo no está disponible. Ejemplos de este tipo de entornos incluyen viajes espaciales, tácticas militares,

algunas formas de respuesta a desastres, bajo el agua, y algunas formas de Redes de sensores / actuadores ad-hoc. También puede incluir la conectividad a Internet en lugares remotos. El proyecto lanzado por DTNRG es dtnbone, un esfuerzo para establecer una Red de nodos DTN (Principalmente DTN2 e ION) sobre el Internet terrestre (global) conformada por varias organizaciones alrededor del planeta para la interoperabilidad, despliegue de aplicaciones, y el desarrollo de una Red de administración y enrutamiento convencional. Los nodos están ubicados en Ohio University (Athens, OH), Shawnee State University (Portsmouth, OH), Comnet, TKK (Espoo, Finland), Trinity College Dublin (Dublin, Ireland), NASA Glenn Research Center (Cleveland, Ohio), Viagénie (Québec City, Canada), BBN Technologies (Cambridge, MA), IBR, TU Braunschweig (Brunswiek, Germany) [26].

DARPA, Desde el 2003 mantiene un programa de DTN con el objetivo de desarrollar servicios de Red que proporcionen información crítica de forma fiable incluso cuando no existe una ruta de punto a punto a través de la Red. DARPA basa dos fases de su programa en Spindle (Survivable Policy-Influenced Networking: Disruption-Tolerance through Learning and Evolution, por sus siglas en inglés) proyecto dirigido por BBN Technologies. La tercera fase del programa DARPA

DTN pretende crear el primer equipamiento fiable con DTN para acceder a información de táctica militar [25].

El proyecto TIER (Technology and Infrastructure for Developing Regions, por sus siglas en inglés) es conformado por un grupo de investigación de la Universidad de California, Berkeley, sus proyectos principales son las herramientas de educación, salud, inalámbricas (WiLDNet), almacenamiento distribuido (TierStore), y las tecnologías del habla [25].

DakNet fue un proyecto DTN inicial de MIT Media Lab, que aparentemente, es comercializado por First Mile Solutions. Este fue una de las primeras instancias en usar transporte programado (data mules) para transportar paquetes entre “kioskos” equipados con Wi-Fi en pueblos sobre una base regular [25].

El proyecto N4C (EU’s Seventh Framework Network for Communication Challenged Communities, por sus siglas en inglés) también usa data mules y está basado en trabajos previos del proyecto SNC (Sámi Network Connectivity, por sus siglas en inglés). Estos proyectos usan encuentros oportunistas con data mules en el ártico

Sueco (en su mayor parte helicópteros, y posiblemente motos de nieve para excursionistas) para transferir paquetes entre campos temporales de la comunidad Sámi y la Internet [25].

En el contexto de Redes de soporte para desastre y emergencias. El laboratorio MMLAB (Multimedia and Mobile Communications Laboratory, por sus siglas en el inglés) de las Universidad Nacional de Seúl ha investigado la arquitectura de Redes inteligentes de emergencia DTN usando comunicaciones extensivas inalámbricas temporales [25].

La NASA, en el 2008 llevó a cabo experimentos simulando las comunicaciones con vehículos de exploración en la superficie de Marte retransmitida a través de un agente DTN instalado en la nave espacial Epoxi. La Figura 2.1 – Red de impacto profundo (DINET) experimento. Nos muestra la configuración usada por el experimento Deep Impact Network (DINET, por sus siglas en inglés). Recientemente, el CCSDS (Consultative Committee on Space Data Systems, por sus siglas en inglés) ha iniciado un grupo de trabajo DTN que examina la idoneidad de los experimentos RFC de BP y LPT para usarlos como un estándar CCSDS para las misiones espaciales

futuras. Tanto la NASA y la Agencia Espacial Europea (European Space Agency, ESP por sus siglas en inglés) tienen varias actividades relacionadas con la DTN y están alimentando el proceso de normalización [25].

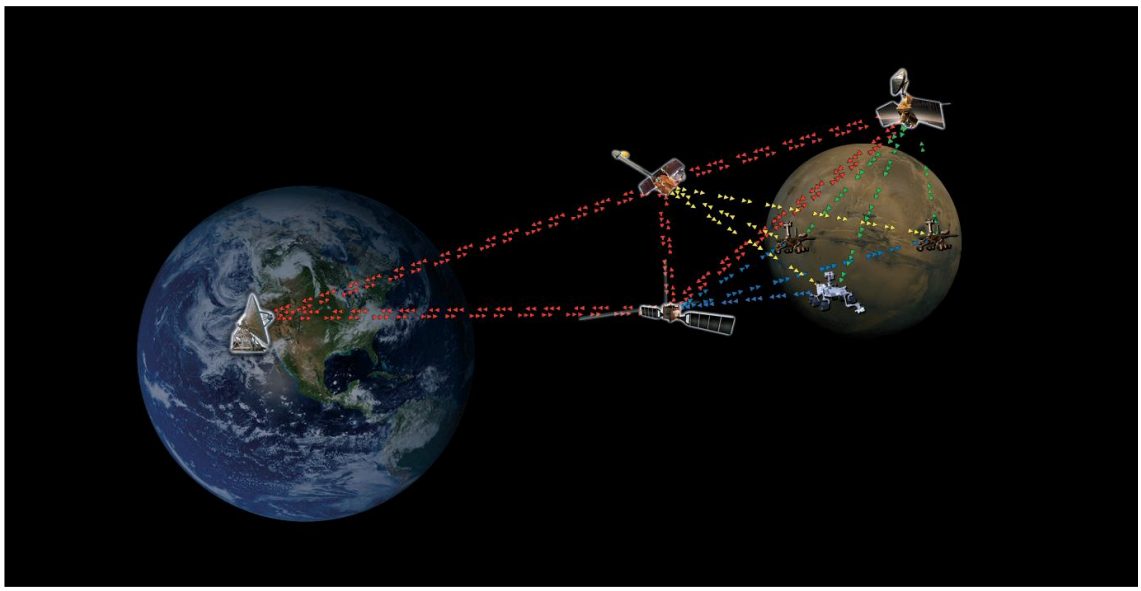


Figura 2.1 - Red de impacto profundo (DINET) experimento [25].

Por otro lado tenemos a la criptografía homomórfica. En 1978 Rivest fue el primero en investigar el diseño de un esquema de cifrado homomórfico. Por desgracia, su homomorfismo de privacidad se rompió un par de años más tarde por Brickell y Yacobi (Brickell y Yacobi, 1987). El homomorfismo volvió a surgir en 1991, cuando Feigenbaum y Merritt (Feigenbaum y Merritt, 1991) plantearon una pregunta importante: ¿Existe una función de cifrado (E) de tal manera

que tanto E (x + y) y E (x * y) sean fáciles de calcular a partir de E (x) y E (y)? En esencia, la pregunta estaba destinada a investigar si existe algún esquema de cifrado algebraico homomórfico que pueda ser diseñado. Lamentablemente, hubo muy poco progreso en la determinación de la existencia de tales esquemas de cifrado que sean eficientes y seguros hasta el 2009, cuando Craig Gentry, en su artículo seminal, teóricamente demostró la posibilidad de la construcción de un sistema de este tipo de cifrado (Gentry, 2009).

Otras implementaciones e investigaciones las mencionamos anteriormente en el Subcapítulo 1.1 Antecedentes, donde se destacan implementaciones de cifrado homomórfico en Redes de sensores, a Craig Gentry y Shai Halevi por la implementación funcional de un sistema totalmente homomórfico, a IBM por la liberación de HElib en GitHub, [11], una librería que implementa el sistema de cifrado homomórfico de Brakerski, Gentry y Vaikuntanathan (BGV) [12], escrita en lenguaje C++ y bajo licencia GNU GPL [13].

Los principales investigadores de Europa sobre criptografía homomórfica se reúnen en el proyecto HEAT, con sello de la bandera de la Unión Europea, lo conforman KU Leuven, Bélgica (Coordinador),

Universidad de Bristol, Reino Unido y la Universidad de Luxemburgo, Luxemburgo, con los principales expertos en criptoanálisis (Universidad Pierre et Marie Curie, Francia), y tres socios con intereses industriales existentes en el campo (CryptoExperts, Francia, NXP Semiconductors, Bélgica y Thales UK, Reino Unido). El objetivo principal de HEAT es producir un cambio radical en la eficiencia y la aplicabilidad de la criptografía homomórfica. HEAT aspira como producto una biblioteca de software de código abierto para soporte de aplicaciones que deseen hacer uso de la criptografía homomórfica. Los resultados del proyecto HEAT serán altamente beneficioso para la industria europea y la investigación académica ya que permiten el uso de la criptografía homomórfica por una variedad más amplia de desarrolladores finales [27].

Construcciones recientes y los esfuerzos de implementación han mejorado drásticamente la eficiencia del cifrado totalmente homomórfico, el sistema criptográfico totalmente homomórfico de Gentry se considera muy ineficiente y computacionalmente intensivo. Los esfuerzos iniciales de implementación se centraron en el esquema original de Gentry y sus variantes, que parecían representar los cuellos de botella inherentes en lugar de eficiencia. Implementaciones posteriores aprovechan los últimos avances que dan lugar a sistemas

de cifrado totalmente homomórficos asintóticamente mejores, así como nuevos mecanismos algebraicos para mejorar la eficiencia global de estos [20].

2.2. Redes Tolerantes a Retardos, DTN

Las Redes DTN surgieron de los intentos de desarrollar una Internet interplanetaria, pero se ha convertido en un área activa de investigación de Redes, con aplicaciones en Redes espaciales, la creación de Redes de táctica militar, y la creación de Redes para diversos desafíos comunitarios [25].

Una DTN es una Red de pequeñas Redes, soporta interoperabilidad con otras Redes con la capacidad de soportar retardos e interrupciones y la traducción entre protocolos de comunicación entre estas Redes. Al proporcionar estas funciones, las DTNs se adaptan a la movilidad y el poder limitado que involucra los dispositivos de comunicación inalámbrica [24]. Una ilustración de una Red DTN como conjunto de Redes que manejan diferentes protocolos e infraestructura la podremos observar en la Figura 2.2 – Redes Tolerantes a Retardos.

Los entornos DTN están caracterizados por la conectividad intermitente, retardos largos y variables, velocidades de datos asimétricos y altos índices de error. Las DTN superan los problemas asociados con la conectividad mediante el uso de la conmutación de mensajes store-and-forward. Este es un método muy antiguo, utilizado por los sistemas de pony-express y postales desde la antigüedad. Mensajes enteros (bloques enteros de datos de usuario application-program) -o- piezas (fragmentos) de tales mensajes que se mueven -(reenvían) desde un lugar de almacenamiento en un nodo (switch intersection) a un lugar de almacenamiento en otro nodo, a lo largo de un camino que finalmente llegan al destino [24].

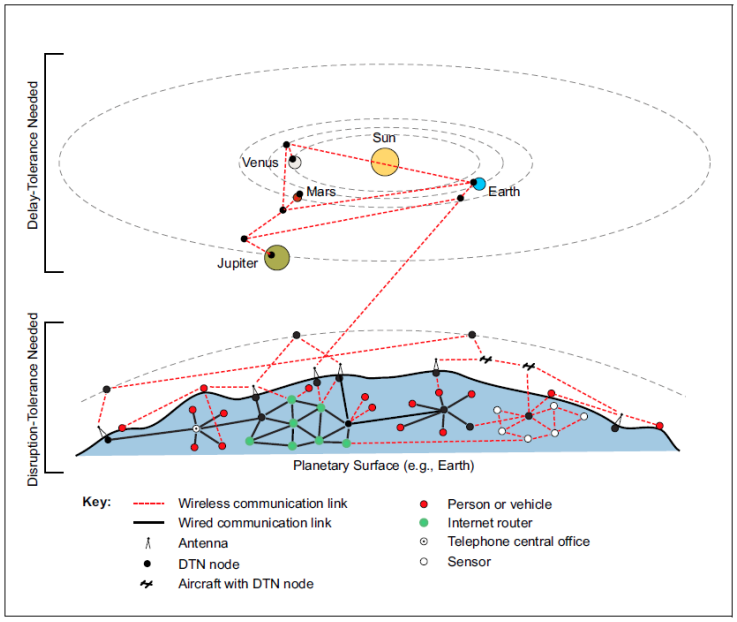


Figura 2.2 - Redes Tolerantes a Retardos [24].

2.2.1. Conectividad DTN

Las DTNs soportan las conexiones intermitentes entre nodos aislando los retardos e interrupciones con la técnica de store-and-forward dando lugar a una conexión intermitente la cual puede ser oportunista o programada [24].

Los contactos oportunistas DTN lo estudiaremos en el Capítulo 2.3 Redes Oportunistas. Los Contactos programados involucran el envío de mensajes entre nodos que no están en contacto directamente, guardando la información hasta que pueda ser reenviada o hasta que una aplicación receptora pueda sincronizarse a la velocidad de datos del remitente. Los contactos programados requieren de sincronización del tiempo a través de la DTN [24].

Un ejemplo de contactos programados son las comunicaciones interplanetarias, donde es aplicable una Red DTN para permitir la interoperabilidad de los residentes de Internet en la Tierra con otros residentes de Internet remotos en otros planetas o naves espaciales en tránsito para el intercambio eficiente de datos científicos como telemetría e imágenes [28]. Potencialmente los nodos de comunicación se mueven a lo largo trayectorias orbitales previsibles,

por lo que pueden predecir o recibir los horarios de sus posiciones futuras y por lo tanto organizar sus sesiones de comunicaciones futuras [24]. Este tipo de contacto lo podemos ilustrar en la Figura 2.3 – DTN, Contacto programado.

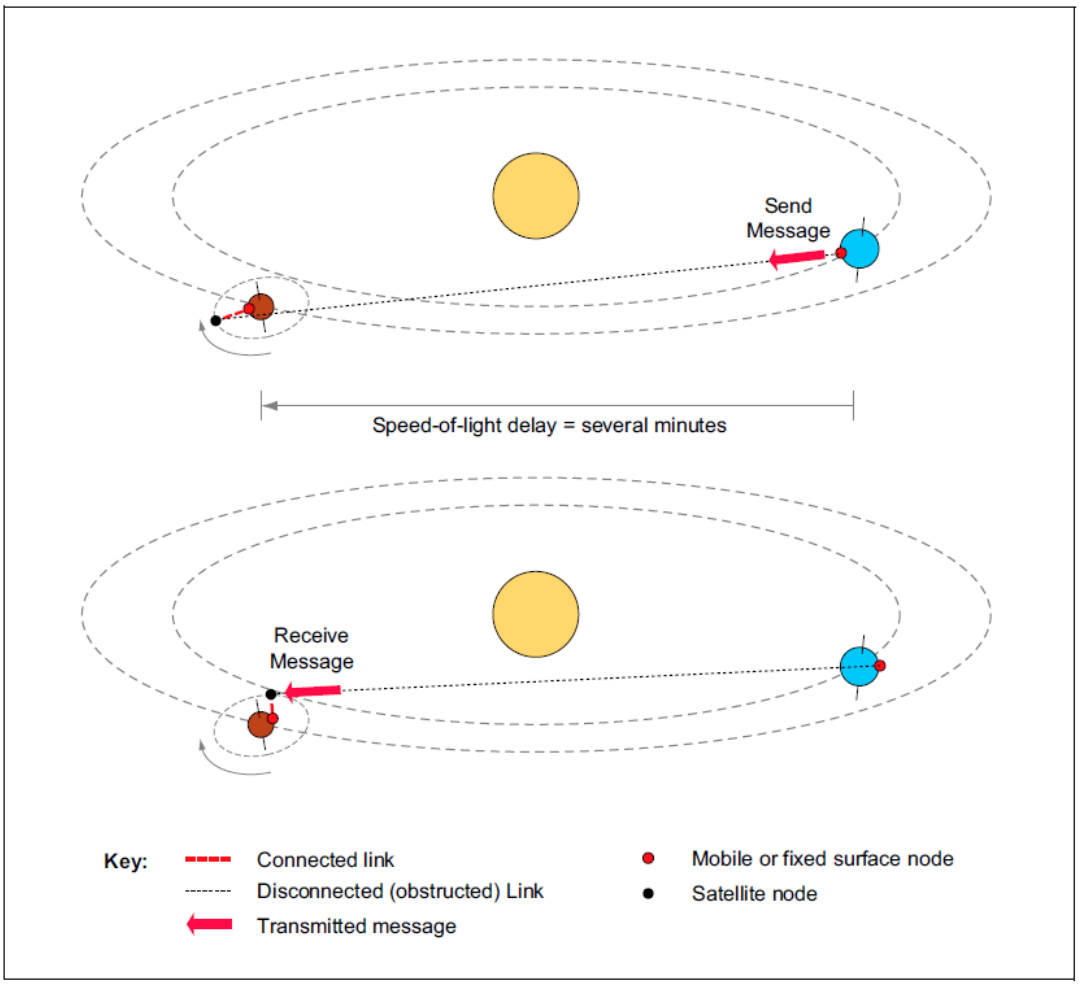


Figura 2.3 - DTN, Contacto programado [24].

2.2.1. Enrutamiento DTN

En la Internet, los protocolos TCP e IP son usados en la Red de datos. TCP trabaja sobre rutas destino, donde existe disponibilidad punto a punto para entregar segmentos TCP, por otro lado IP trabaja sobre todos los nodos en la Red, donde enruta datagramas IPs.

En las Redes tolerantes a retardo, DTN, todos los nodos implementan el protocolo Bundle y los protocolos de capa 1, 2, 3 y 4 del Modelo TCP/IP (Físico, Enlace, Red, Transporte), cada nodo posee un agente Bundle, el cual recepta, almacena, emite, o retransmite Bundles enteros o fragmentos de estos - al igual como lo hace el protocolo IP al transmitir datagramas o fragmentos de estos – del otro lado el agente Bundle dentro del nodo receptor reensambla los fragmentos de Bundle. El protocolo Bundle permite el almacenamiento y envió de Bundles sobre los nodos, conformando una Red tolerante a retardos, DTN. El protocolo Bundle consiste de tres cosas, 1) una cabecera Bundle, la cual consiste en uno o más bloques DTN insertados por el agente Bundle, 2) data de usuario de la capa de Aplicación, incluyendo cómo manejar la data del usuario e información de control de como procesar, almacenar y disponer. Podemos observar en la Figura 2.4

– Internet vs Enrutamiento DTN. Las diferencias en la transferencia de la información de ambos modelos [24].

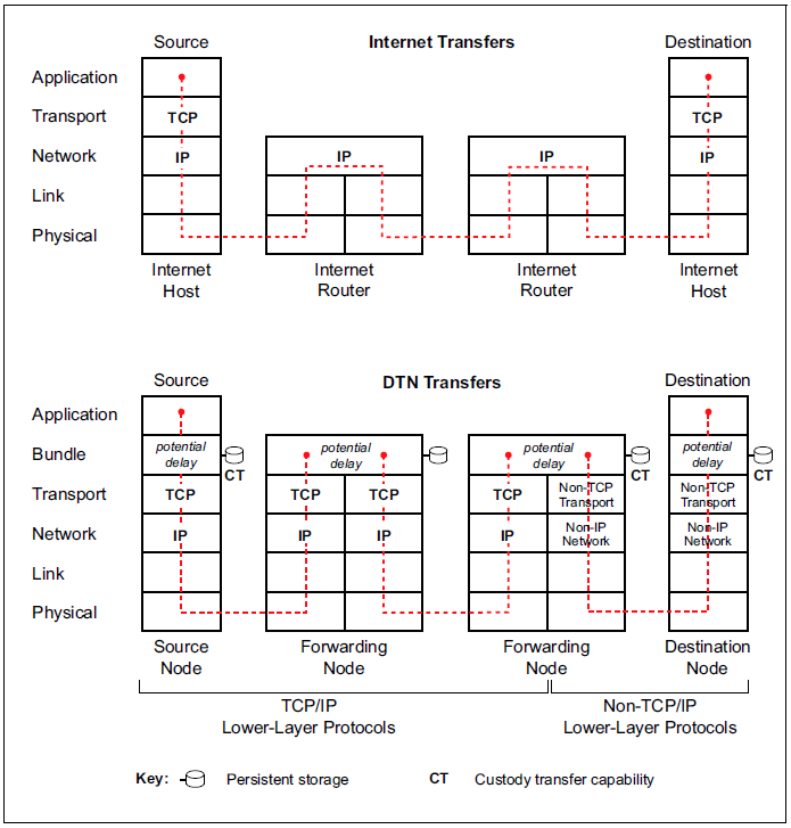


Figura 2.4 - Internet vs Enrutamiento DTN [24].

2.2.3. Aplicaciones potenciales de tecnología DTN

La arquitectura de conmutación de mensajes store-and-forward de las Redes DTN es una generalización del trabajo original concebido para soportar la Internet Interplanetaria (IPN). Los objetivos principales son

la interoperabilidad a través del entorno de Red y la confianza en la capacidad de sobrevivir a fallos de Hardware (Red) y Software (protocolos). Aunque las DTN fueron diseñadas para el uso interplanetario, existe un sin número de aplicaciones en la Tierra. A continuación presentaremos un resumen de posibles aplicaciones [24]:

- Agencias espaciales. La comunicación de la Estación Espacial Internacional (actualmente en funcionamiento para la investigación), la comunicación interplanetaria y futuro control de los desechos espaciales.

- Militares e inteligencia. Redes Ad-Hoc móviles (MANETs) para la comunicación inalámbrica y monitoreo, seguimiento de la carga, búsqueda y rescate de comunicación, la comunicación aérea y control de vehículos no tripulados (UAV)

- Comercial. Cargo y localización de vehículos (por carretera, ferrocarril, mar y aire), transacciones de datos (por ejemplo, financieros, reservas), monitoreo de cultivos agrícolas,

seguimiento de procesamiento en plantas, la comunicación en minas subterráneas.

- Servicio Público y Seguridad. La seguridad y la comunicación en desastres, búsqueda y comunicación de rescate, monitoreo de la ayuda humanitaria, Redes de transporte inteligentes, Redes eléctricas de potencia inteligentes, control global aeropuerto de tráfico, monitoreo de infraestructura integral, comunicación y control de vehículo aéreo no tripulado (UAV) y aprendizaje a distancia.

- Control ambiental. Migración animal, propiedades del suelo y la estabilidad, condiciones atmosféricas y oceanográficas, eventos sismológicos.

- Ingeniería e Investigación Científica. Expertos en la materia de la Red, investigación académica por profesores y estudiantes.

2.3. Redes oportunistas

El número cada vez mayor de dispositivos personales con capacidad de comunicación inalámbrica genera la posibilidad de crear Redes espontáneas ocasionales entre los dispositivos, dependiendo de las oportunidades de contacto [29]. Las Redes oportunistas son una de las evoluciones más interesantes de las Redes móviles Ad-Hoc (mobile Ad-Hoc network, MANET). En las Redes oportunistas, los nodos móviles están en la capacidad de comunicarse con otros nodos incluso si la ruta de conexión nunca ha existido. Por otra parte, no se espera que los nodos posean o adquieran conocimiento acerca de la topología de la Red, lo cual es tradicionalmente necesario en protocolos de enrutamiento en Redes MANET [18].

Las Redes oportunistas están caracterizadas por ser altamente dinámicas, conformadas por nodos estáticos y móviles que tienen la capacidad de obtener ventajas de contactos oportunistas en un determinado tiempo realizando comunicaciones intermitentes donde la movilidad y limitaciones de los dispositivos, obstáculos físicos y distancias resultan en una no posible existencia de rutas punto a punto hacia el destino. Por lo tanto en las Redes oportunistas las rutas se construyen dinámicamente mientras los mensajes son enrutados entre

los emisores y receptores, y cualquier nodo puede oportunistamente ser escogido para el siguiente salto, siempre y cuando sea probable llevar el mensaje más cerca del destino final, cuando no exista un envío oportunista, el nodo guarda el mensaje y espera para un futuro contacto oportunista con otro nodo para reenviar la información. Estos requisitos hacen de las Redes oportunistas un campo de investigación desafiante y prometedor [29] [18]. Un ejemplo del comportamiento de una Red oportunista las podemos observar en la Figura 2.5 – DTN, Contacto oportunista.

Las Redes oportunistas o OppNets pueden ser aplicadas como Redes de emergencia o desastres, Redes de seguimiento animal, Redes sensoriales, Redes tolerantes a retardos DTN o hasta Redes interplanetarias [29].

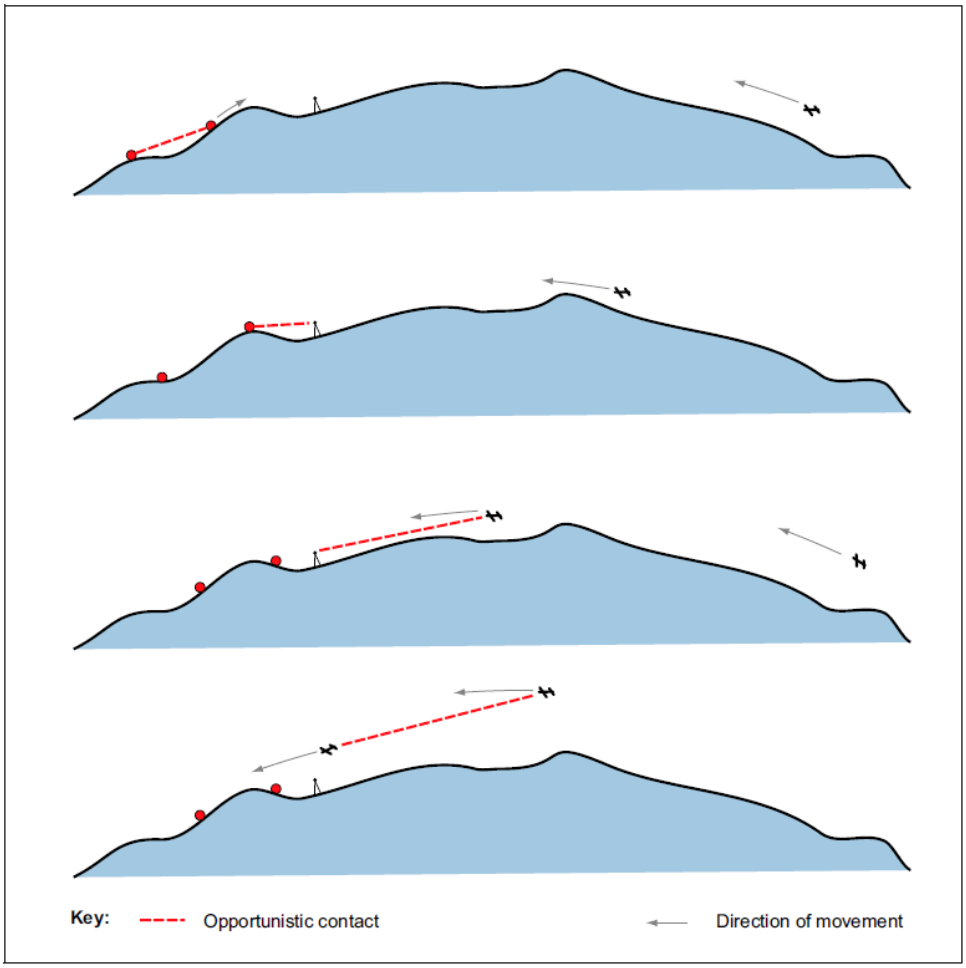


Figura 2.5 - DTN, Contacto oportunista [24].

2.3.1. Enrutamiento en Redes oportunistas

El enrutamiento en Redes oportunistas se basa generalmente en algún tipo de inundación controlada. Pero a menudo esto se traduce en un consumo muy elevado de recursos y congestión de la Red [30]. El enrutamiento considera que cualquier contacto entre nodos y decisiones de envío son realizadas utilizando conocimientos

adquiridos localmente acerca del comportamiento de los nodos para predecir que nodos son propensos a entregar un mensaje o que puedan acercar el mensaje al destino [29].

Dentro del enrutamiento oportunista podemos encontrar enfoques agrupados en tres categorías: Simple-Copy, cuyo objetivo es mejorar el uso de recursos de la Red; Epidemic, que se propone incrementar la probabilidad de entrega; y, Probabilistic-based, que pretende encontrar un balance óptimo entre las dos categorías previas [31].

El enfoque Simple-Copy, con el propósito de optimizar los recursos de la Red, genera solo una copia de cada mensaje que cruza la Red hacia su destino. Cada copia puede ser retransmitida si el nodo portador decide (ya sea por decisión aleatoria o basada en una función) que en el siguiente encuentro existe una alta probabilidad de alcanzar el destino. Este enfoque es interesante ya que conserva el uso de recursos de la Red (ancho de banda) y del nodo (energía, almacenamiento). Sin embargo, este enfoque sufre de un alto porcentaje de retardos, que resultan en un porcentaje bajo de entregas. Otro problema es la cantidad de conocimiento que necesita para intercambiar con el fin de ayudar a enviar el mensaje, por lo tanto

en algunos escenarios genera demasiada sobrecarga y puede ser imposible de implementarlo [31].

En el enfoque Epidemic, las Redes son creadas por gente que se mueve alrededor, los contactos oportunistas son considerados para incrementar la probabilidad de entrega de mensajes. Con un enfoque de enrutamiento Epidemic, cada nodo en la Red obtiene al menos una copia de cada mensaje como una estrategia de replicación completa que permite el incremento del porcentaje de entregas. La replicación de los mensajes es realizada por medio de vectores de resumen que se intercambian entre dos encuentros. Tales vectores de resumen contienen la lista de mensajes que cada nodo está llevando, permitiendo a los nodos intercambiar todos los mensajes que el otro nodo puede. La propuesta de hecho aumenta la tasa de entrega, ya que cada nodo tiene una copia del mensaje y si asumimos contactos con una duración significativa y suficiente espacio de memoria de intercambio en cada nodo, obtenemos una alta probabilidad de entrega. Con el fin de evitar que los mensajes se repliquen indefinidamente, un campo de número de saltos determina el número máximo de intercambios epidémicos al que un mensaje en particular está sujeto; donde los paquetes son descartados basándose en el espacio de búfer disponible a nivel local. Dado que el número de saltos

hacia el destino no se conoce de antemano el establecimiento de un número de saltos puede disminuir la probabilidad de entrega [32].

El enfoque Probabilistic-Based está basado en la estimación y/o predicción de cuál es la siguiente mejor conjunto de portadores para cada mensaje en función de alguna métrica de probabilidad con el objetivo de maximizar la probabilidad de entrega. Protocolos de reenvío probabilísticos requieren patrones de movilidad de nodos que exhibe regularidades a largo plazo de tal manera que algunos nodos constantemente se reúnen con mayor frecuencia que otros lo largo del tiempo: el tiempo medio entre reuniones entre dos nodos en el pasado sugeriría que se reunirán nuevamente en el futuro con una alta probabilidad. Desde 2003, Diferentes métricas de probabilidad de entrega se han propuesto incluyendo Frequency-Encounters, Aging-Encounters, Aging-Messages, Resource-Allocation, y Social-Similarity [31].

2.3.2. Taxonomía para Redes oportunistas en DTN

En la Figura 2.6 – Taxonomía para Redes oportunistas en DTN, podemos ilustrar una taxonomía que complementa con la tendencia replicación vs envío, con el análisis realizado de propuestas

publicadas entre 2000 y 2010. Por lo tanto, la taxonomía propuesta se basa en una clasificación inicial de todas las propuestas como Forwarding-Flooding-, o Replication-Based. Se centra en un análisis de las características topológicas (por ejemplo, la frecuencia de contacto y edad, utilización de recursos, la formación de comunidades, de los intereses comunes, la popularidad de nodo) asumidos por cada propuesta puede dar lugar a una taxonomía más estable útil para estudiar las Redes del mundo real tales como las Redes informáticas que funcionan sobre la base de la conducta social [31].

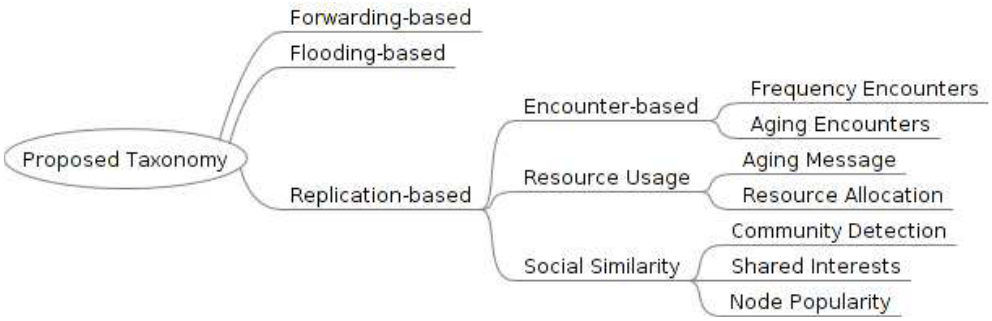


Figura 2.6 - Taxonomía para Redes oportunistas en DTN [31].

2.3.3. Privacidad en Redes oportunistas

Los protocolos de enrutamiento designados a operar en escenarios DTN usualmente generan y usan información acerca del comportamiento de los nodos, así como el histórico de contactos

establecidos con otros nodos. Luego, los nodos comparten esta información con nodos vecinos con el objetivo de mejorar la toma de decisiones de enrutamiento. Además, en algunos casos un nodo es vinculado a una persona, porque el paquete es transportado en su bolsillo o maleta. Por lo tanto, la información que los protocolos de enrutamiento están usando y compartiendo puede ser vista como información privada sobre el paradero de las personas o comportamientos frecuentes. Debido a este problema surgió PrivHab+, un protocolo de enrutamiento geográfico seguro que aprende acerca de los hábitos de movilidad de los nodos de la Red y utiliza esta información de una manera segura. PrivHab+ está diseñado para operar en zonas que carecen de Red, utilizando el enfoque de Store-Carry-and-Forward. PrivHab+ compara nodos y elige la mejor opción para llevar mensajes hacia un lugar geográfico conocido. Para lograr un alto rendimiento y bajo costo operativo, PrivHab+ utiliza la información sobre el paradero de los nodos habituales para tomar decisiones de encaminamiento óptimos. Además hace uso de técnicas criptográficas de cómputo seguro multi-parte como el cifrado homomórfico de Paillier para preservar la privacidad de los nodos mientras toma decisiones de enrutamiento. PrivHab+ es un ejemplo de sistemas funcionales en base de cifrado homomórfico sobre Redes oportunistas [15].

2.4. Cifrado Homomórfico

En matemáticas, el homomorfismo describe la transformación de un conjunto de datos en otro preservando las relaciones entre los elementos en ambos conjuntos, el término es derivado del griego “misma estructura” [33]. Un esquema de cifrado homomórfico es un criptosistema que permite realizar cálculos u operaciones matemáticas complejas en datos cifrados sin descifrarlos [34], donde las mismas operaciones matemáticas - ya sea que se realizan en los datos cifrados o descifrados - darán resultados equivalentes.

Si representamos una función de cifrado con Cif y la función de descifrado con Des, el cifrado homomórfico cumpliría con la igualdad:

$$Des\left(o_p^{'}\left(\Psi_1,\ldots,\Psi_n\right)\right) = o_p\left(r_1,\ldots,r_n\right) \tag{2.1}$$

Donde $\Psi_i = Cif(r_i)$, o_p representa una operación algebraica sobre el dato o texto plano y $o_p^{'}$ representa la operación equivalente en el dato cifrado. Es importante recalcar que $o_p^{'}$ puede ser igual o diferente a o_p. Un ejemplo de cifrado homomórfico lo podemos encontrar en la siguiente igualdad:

$$\mathcal{C}_{r1}(t_1) \odot \mathcal{C}_{r2}(t_2) = \mathcal{C}_r(t_1 \oplus t_2) \tag{2.2}$$

Donde $\mathcal{C}_r(t)$ representa la función de cifrado del dato **t** usando el valor aleatorio **r**, el operador $\odot$ representa la operación binaria producto y el operador $\oplus$ la operación binaria suma. De esta manera, el resultado del producto sobre dos datos cifrados es equivalente a cifrar el resultado de la operación suma sobre los mismos datos descifrados [21].

Actualmente existen dos tipos de esquemas o sistemas criptográficos que clasifican al cifrado homomórfico, en un esquema están los sistemas criptográficos parcialmente homomórficos que son de mayor eficiencia, y por otro lado están los sistemas criptográficos totalmente homomórficos de menor eficiencia, estos esquemas lo estudiaremos en los Subcapítulos 2.4.1 y 2.4.2 respectivamente.

2.4.1. Sistemas criptográficos parcialmente homomórficos

Los esquemas parcialmente homomórficos corresponden a los criptosistemas homomórficos que funcionan con una operación, ya sea esta la suma o multiplicación. Los sistemas criptográficos parcialmente homomórficos o también denominados solamente criptosistemas homomórficos están conformados por dos tipos de

esquemas de cifrado denominados probabilísticos o determinísticos. Por ejemplo si el algoritmo de cifrado obtiene de entrada adicional un número aleatorio uniforme perteneciente al conjunto de los reales, el esquema de cifrado se llama probabilístico, de lo contrario, se llama determinista [20].

En los siguientes ejemplos, presentamos esquemas homomórficos determinista de forma multiplicativa, esquemas homomórficos probabilísticos de forma aditiva, y sistemas criptográficos clásicos, donde la notación E(x) se usa para representar el cifrado del mensaje x:

RSA sin relleno. Es un ejemplo de un criptosistema homomórfico de forma multiplicativa determinista, en el sistema criptográfico RSA, dada cierta clave pública definida como el módulo *m* y el exponente *e*, el cifrado de un mensaje *x* se expresa como:

$$\mathcal{E}(x) = x^e \mod m \tag{2.3}$$

La propiedad homomórfica viene dada por la expresión [21]:

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = x_1^e \cdot x_2^e \mod m = (x_1 \cdot x_2)^e \mod m = \mathcal{E}(x_1 \cdot x_2) \tag{2.4}$$

Podemos observar que el cifrado del producto de dos mensajes se puede calcular de manera eficiente por la multiplicación de los textos cifrados correspondientes [20].

Goldwasser-Micali. Es un ejemplo de un criptosistema homomórfico aditivo probabilístico, dada cierta clave pública definida como el módulo *m* y la cuadrática no residual *x*, el cifrado de un bit *b* se expresa como:

$$\mathcal{E}(b) = x^b r^2 \mod m \tag{2.5}$$

Donde el algoritmo de cifrado obtiene de entrada adicional un valor *r* para un $r \in \{0, \ldots, m-1\}$ aleatorio. La propiedad homomórfica viene dada por la expresión:

$$\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) = x^{b_1} r_1^2 x^{b_2} r_2^2 = x^{b_1+b_2}(r_1 r_2)^2 = \mathcal{E}(b_1 \oplus b_2) \tag{2.6}$$

Donde $\oplus$ denota la suma de módulo 2 o suma binaria (XOR) [20] [21].

ElGamal. En el sistema criptográfico ElGamal, dado un grupo *G* con clave pública (*G; q; g; h*), donde $h = g^x$, y *x* representa la clave privada, el cifrado de un mensaje *m* se expresa como:

$$\mathcal{E}(m) = (g^r, m \cdot h^r) \tag{2.7}$$

Para un $r \in \{0, \ldots, q-1\}$ aleatorio. La propiedad homomórfica viene dada por la expresión [21]:

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) = (g^{r_1+r_2}, (x_1 \cdot x_2)h^{r_1+r_2}) = \mathcal{E}(x_1 \cdot x_2) \tag{2.8}$$

Benaloh. En el sistema criptográfico Benaloh, dada cierta clave pública definida como el módulo *m* y la base *g*, con un tamaño de bloque *c*, el cifrado del mensaje *x* se expresa como:

$$\mathcal{E}(x) = g^x r^c \mod m, \tag{2.9}$$

Para un $r \in \{0, \ldots, m-1\}$ aleatorio. La propiedad homomórfica viene dada por la expresión [21]:

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{x_1} r_1^c)(g^{x_2} r_2^c) = g^{x_1+x_2}(r_1 r_2)^c = \mathcal{E}(x_1 + x_2 \mod c) \tag{2.10}$$

Paillier. En el sistema criptográfico Paillier, dada cierta clave pública definida como el módulo *m* y la base *g*, entonces el cifrado de un mensaje *x* se expresa como:

$$\mathcal{E}(x) = g^x r^m \mod m^2 \tag{2.11}$$

Para un $r \in \{0, \ldots, m-1\}$ aleatorio. La propiedad homomórfica viene dada por la expresión:

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{x_1} r_1^m)(g^{x_2} r_2^m) = g^{x_1+x_2}(r_1 r_2)^m = \mathcal{E}(x_1 + x_2 \mod m) \quad (2.12)$$

2.4.2. Sistemas criptográficos totalmente homomórficos

Un sistema de cifrado homomórfico que soporta realizar cálculos complejos arbitrariamente elegidos de forma dinámica en que los datos permanecen cifrados, a pesar de no tener la clave secreta de descifrado, se conoce como cifrado totalmente homomórfico o FHE por sus siglas en inglés (Fully Homomorphic Encryption, FHE) [12].

El primer esquema de cifrado totalmente homomórfico fue descrito por Gentry en el 2009, este esquema soportaba ambas operaciones matemáticas, la suma y la multiplicación. El cifrado totalmente homomórfico propuesto por Gentry consiste en varios pasos: En primer lugar, se construye un esquema homomórfico *"somewhat homomorphic"* que soporta la evaluación de polinomios de grado bajo en los datos cifrados. Luego, se concentra el procedimiento de descifrado de modo que se puede expresar como un polinomio de bajo grado que está soportado por el sistema, y finalmente, se aplica una

transformación bootstrapping para obtener un esquema totalmente homomórfico [20].

El enfoque fundamental de este esquema es obtener y establecer un proceso que puede evaluar polinomios de grado suficientemente alto utilizando un procedimiento de descodificación que puede ser expresado como un polinomio de grado bajo. Una vez que el grado de polinomios que se puede evaluar por el esquema supera el grado del polinomio descifrado por un factor de dos, el esquema se llama bootstrappable y luego se puede convertir en un esquema totalmente homomórfico [20].

2.4.3. Aplicaciones del cifrado homomórfico

Con el cifrado homomórfico, una empresa podría cifrar toda su base de datos de e-mails y subirla a la nube. Después podría utilizar los datos guardados en ella como desee: por ejemplo, para hacer una búsqueda en la base de datos y entender cómo colaboran sus trabajadores. Los resultados se podrían descargar y descifrar sin exponer los detalles de ningún correo electrónico [8].

Un motor de búsqueda con cifrado homomórfico, por ejemplo, podría tener términos de búsqueda cifrada y comparaciones con un índice cifrado en la Web. O una base de datos cifrada homomórficamente almacenada en la nube podría permitir a los usuarios consultar cuánto dinero ganó un empleado en cierto trimestre, aceptando un nombre de empleado cifrado y de salida una respuesta cifrada, evitando los problemas de privacidad que normalmente afectan a los servicios en línea que se ocupan de este tipo de datos sensibles [34]. A continuación encontraremos más aplicaciones de un esquema de cifrado homomórfico:

- Protección de agentes móviles. Una de las aplicaciones más interesantes del cifrado homomórfico está en la protección de agentes móviles. Ya que todas las arquitecturas de cómputo convencionales están basadas en binarios y solo requieren de la multiplicación y adición, los criptosistemas homomórficos ofrecerían la posibilidad de cifrar todo un programa manteniéndolo ejecutable [20].

- Cómputo multipartidista. Esquema donde las partes interesadas en cómputo común utilizan funciones públicas

manteniendo la privacidad de las entradas, por lo tanto la función que debe ser calculada es de conocimiento público, mientras que las entradas son datos cifrados y privados [20].

- Esquema de secreto compartido. Las partes comparten un secreto de modo que las partes individuales no puedan reconstruir el secreto desde la información disponible. Por lo tanto, si alguna de las partes coopera con la otra, podrían reconstruir el secreto. En este escenario, la propiedad homomórfica implica que la composición del secreto compartido es equivalente a la compartición de los secretos [20].

- Esquemas de elección (voto electrónico). La propiedad homomórfica provee una herramienta para obtener la cantidad total de votos cifrados sin descifrar los votos individuales [20].

- Protocolos de lotería. Por lo general, en una lotería con cifrado, un número que señala al boleto ganador tiene que ser conjuntamente y aleatoriamente elegido por todos los participantes. El uso de un esquema de cifrado homomórfico

puede ser realizado de la siguiente forma: Cada jugador elige un número aleatorio que se cifra. Luego, utilizando la propiedad homomórfica, el cifrado de la suma de los valores aleatorios se puede calcular de manera eficiente. La combinación de este y un esquema de umbral de descifrado conduce a la funcionalidad deseada [35].

- La agregación de datos en Redes de sensores inalámbricos. Una Red de agregación de datos en WSNs es una técnica que combina los resultados parciales de los nodos intermediarios en el enrutamiento hacia la estación de base, reduciendo así la sobrecarga de comunicación y optimizando el ancho de banda utilizado en los enlaces inalámbricos. Sin embargo, esta técnica plantea cuestiones de privacidad y seguridad si los nodos sensores necesitan compartir sus datos con el nodo que agrega los nuevos datos “nodo agregador”. En aplicaciones tales como cuidado de la salud y la vigilancia militar donde la sensibilidad de los datos privados del sensor es muy alta, la agregación tiene que ser llevado a cabo de una manera que preserve la privacidad, de manera que los datos sensibles no son revelados al nodo agregador [36].

2.5. Criptosistema de Paillier

El criptosistema de Paillier es un esquema modular o algoritmo asimétrico probabilístico de clave pública creado por el francés Pascal Paillier en 1999. Dentro del sistema criptográfico de Paillier se cree que el problema de calcular las clases del enésimo residuo es computacionalmente difícil. Esto se conoce como Residuidad Compuesta y es la base de este sistema criptográfico [37].

La criptografía de clave pública aplica como técnica el uso de algoritmos de clave asimétrica, donde la clave utilizada para cifrar un mensaje no es la misma utilizada para descifrarla. Cada usuario tiene un par de claves criptográficas - una clave pública y una clave privada. La clave privada se mantiene en secreto, mientras que la clave pública puede ser ampliamente distribuida. Los mensajes se cifran con la clave pública del destinatario y sólo pueden ser descifrados con la clave privada correspondiente. Las claves están relacionadas matemáticamente, pero la clave privada no es fácilmente derivada de la clave pública (es decir, en la práctica real o proyectada) [38].

2.5.1. Esquema del criptosistema de Paillier

El criptosistema de Paillier funciona de la siguiente forma [38]:

Generación de la llave

1. Elija dos grandes números primos *p* y *q* al azar y de forma independiente el uno del otro de tal manera que

$$\gcd\big(pq,(p-1)(q-1)\big)=1 \tag{2.13}$$

2. Se calcula el módulo RSA

$$n = pq \tag{2.14}$$

Y la función Carmichael

$$\lambda = lcm(p-1, q-1) \tag{2.15}$$

Puede ser calculada usando

$$\lambda = \frac{(p-1)(q-1)}{\gcd(p-1,q-1)}. \tag{2.16}$$

3. Selecciona el generador *g* donde $g \in \mathbb{Z}^*_{n^2}$. Existen dos formas de seleccionar *g*.

   a. Selección aleatoria de *g* de un conjunto $\mathbb{Z}^*_{n^2}$ donde

$$\gcd\left(\frac{g^{\lambda} \bmod n^2-1}{n}, n\right) = 1 \quad (2.17)$$

Hay una serie ϕ (n) * ϕ (n) de generadores válidos, por lo tanto, la probabilidad de elegir fuera de nϕ (n) elementos del conjunto $\mathbb{Z}^*_{n^2}$ es relativamente alta para el gran *n*.

b. Seleccionar *α* y *β* al azar de un conjunto $\mathbb{Z}^*_{n^2}$ luego calcular

$$g = (\alpha n + 1)\beta^n \bmod n^2 \quad (2.18)$$

En este caso el generador seleccionado siempre satisface la condición anterior.

4. Calcula el siguiente inverso multiplicativo modular

$$\mu = \left(L(g^{\lambda} \bmod n^2)\right)^{-1} \bmod n \quad (2.19)$$

Donde la función L es definida como

$$L(x) = \frac{x-1}{n} \quad (2.20)$$

Este inverso multiplicativo existe si y sólo si el generador válido fue seleccionado en el paso anterior.

Por lo tanto:

- La clave pública (cifrado) es (*n, g*).

- La clave privada (descifrado) es (λ, μ).

Si se utiliza p, q de longitud equivalente, una variante más simple de los pasos anteriores de generación de claves sería establecer

$g = n + 1$, (2.21)

$\lambda = \varphi(n)$ y (2.22)

$$\mu = \varphi(n)^{-1} \bmod n, \quad (2.23)$$

Donde $\varphi(n) = (p-1)(q-1)$ [38]. (2.24)

Cifrado

1. Sea m un mensaje a cifrar en $m \in \mathbb{Z}n$
2. Se selecciona r al azar, donde $r \in \mathbb{Z}^*n$
3. Se calcula el texto cifrado como:

$$c = g^m \cdot r^n \bmod n^2 \quad (2.25)$$

Descifrado

1. Con el texto cifrado $c \in \mathbb{Z}^*{}_{n^2}$
2. Se calcula el mensaje

$$m = L\left(c^{\lambda} \bmod n^2\right) \cdot \mu \bmod n \tag{2.26}$$

2.5.2. Propiedades homomórficas

Una característica notable del criptosistema de Paillier es la propiedad homomórfica. Como la función de cifrado es aditivamente homomórfica, las siguientes entidades se pueden describir [37]:

Suma homomórfica de textos planos

El producto de dos textos cifrados descifrará a la suma de sus correspondientes textos planos,

$$D(E(m_1, r_1) * E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n \tag{2.27}$$

El producto de un texto cifrado con *g* elevado al texto plano descifrará la suma del texto claro correspondiente [37],

$$D(E(m_1, r_1) * g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n \tag{2.28}$$

En la práctica, esto lleva a las siguientes identidades, donde $\forall\, \boldsymbol{m_1, m_2} \in \mathbb{Z}_n$ y $\boldsymbol{k} \in \mathbb{N}$:

$$D(E(m_1)E(m_2)\ mod\ n^2) = m_1 + m_2\ mod\ n$$
$$D\big(E(m)^k\ mod\ n^2\big) = km\ mod\ n$$
$$D(E(m_1)g^{m_2}\ mod\ n^2) = m_1 + m_2\ mod\ n \quad (2.29)$$

Multiplicación homomórfica de textos planos

Un texto cifrado elevado a la potencia de otro texto en plano descifrará al producto de los dos textos planos:

$$\left.\begin{matrix} D(E(m_1)^{m_2}\ mod\ n^2) \\ D(E(m_2)^{m_1}\ mod\ n^2) \end{matrix}\right\} = m_1 m_2 \bmod n \quad (2.30)$$

Más generalmente, un texto cifrado elevado a una constante *k* descifrará al producto de la de texto claro y la constante,

$$D(E(m_1, r_1)^k \bmod n^2) = km_1 \bmod n. \quad (2.31)$$

Sin embargo, no hay manera de calcular el producto cifrado de dos mensajes cifrados de Paillier sin conocer la clave privada [37].

Resta homomórfica de textos planos

Para el desarrollo de este proyecto es necesaria la sustracción homomórfica, la cual no se encuentra contemplada en el esquema criptográfico original de Paillier. Para ello el proyecto Habcast

desarrolló una función de sustracción la cual se basa en obtener el inverso del sustraendo en modulo *n^2*, el cual se multiplica por el minuendo en modulo *n^2* para encontrar la diferencia. Este procedimiento se resume en la fórmula:

$$E(a - b) = E(a)/E(b) \bmod n^2 \qquad (2.32)$$

Este procedimiento posee la restricción de que el sustraendo debe tener un inverso en *Zn^2* [15]. La probabilidad de que en la ejecución no se encuentre el inverso es baja, por lo que lo hace una función práctica para las operaciones de sustracción en el MPEM.

Otra técnica para lograr la resta homomórfica es la sustitución de la sustracción por la adición de un valor negativo. Sin embargo, ya que no hay valores negativos en *Zn*, se mapea de manera que se mantenga la adición con otros operadores cifrados.

Se mapea enteros positivos menores que *n/2* usando la función de mapeo *Map(x)* y enteros negativos mayores que *–n/2* con su representación en modulo *n*, como se muestra en la siguiente ecuación:

$$\text{Map}(x) = \begin{cases} x & x \in [0, n/2) \\ x + n & x \in (-n/2, 0) \end{cases} \quad (2.33)$$

De esta forma, se usa la adición de Paillier entre un entero positivo *a* y un entero negativo *–b* (mapeado como *–b + n*) para obtener *(a - b) + n mod n*. Notemos que si *a > b* tenemos *(a - b) + n mod n = (a – b)*, y que si a < b tenemos *(a - b) + n mod n = (a - b) + n*. Finalmente el resultado de la operación puede ser recuperada usando el inverso de la función de mapeo mediante la siguiente función:

$$\text{Inverse Map}(x) = \begin{cases} x & x \in [0, n/2) \\ x - n & x \in (n/2, n-1] \end{cases} \quad (2.34)$$

Para poder usar las funciones de mapeo, debemos estar seguros de que las operaciones usadas en el sistema nunca excederán el límite de *n/2*, esto significa que el resultado del cómputo cifrado nunca deber ser un entero positivo mayor que *n/2*, ni un numero negativo menor que *–n/2*. En la Figura 2.7 – Esquema de mapeo de sistemas de 32bits, podremos observar cómo se emplea la función de mapeo para un sistema de tamaño de cálculo de 32bits [39].

Figura 2.7 - Esquema de mapeo de sistemas de 32bits [39].

CAPÍTULO 3

3. LEVANTAMIENTO DE INFORMACIÓN

En el siguiente capítulo presentaremos las rutas a simular en nuestro MPEM, así como los valores de los viajes en buses interprovinciales. Con estos mapas, rutas y cotizaciones obtenemos la base para la simulación de una Red oportunista capaz de intercambiar información de las cotizaciones mediante el uso de protocolos DTN y de geo-enrutamiento seguro. Luego presentaremos las herramientas que utilizaremos para la simulación e implementación del MPEM basado en cifrado homomórfico sobre una Red oportunista. Estas herramientas serán un simulador para la Red oportunista y las librerías para la implementación del mecanismo de cifrado homomórfico basado en el criptosistema parcialmente homomórfico de Paillier con soporte para la sustracción o resta.

3.1. Sistema de automóvil compartido

Como lo mencionamos anteriormente en los Subcapítulos 1.1 Antecedentes y 1.2. Descripción del problema; los sistemas de automóvil compartido tienen como objetivo reducir los gastos del conductor compartiendo el viaje con pasajeros, por lo que toma la ideología de ser un esquema económico, social y sin ánimo de lucro. Permitiendo a un pasajero el cual desee viajar más cómodo o ahorrar dinero, dirigirse de una ciudad a otra, o de una parroquia a otra junto a un conductor el cual necesita compartir los gastos como gasolina o peaje con otras personas, fomentando el ahorro y socialización compartiendo el tiempo con nuevos compañeros. También mencionamos que el uso del sistema de automóvil compartido data de la Segunda Guerra Mundial en Estados Unidos donde el automóvil compartido fue una técnica para el racionamiento de recursos [5] como la gasolina la cual fue escasa en ese tiempo. En Europa, el auto compartido se ha vuelto cada vez más popular en los últimos años, gracias a la alemana Mitfahrgelegenheit (carpooling.com), y la francesa BlaBlaCar [6], a tal punto, de encontrase en los noticieros españoles que rutas tradicionales de buses viajan con dos o tres pasajeros, debido a que las personas buscan el automóvil compartido para ahorrar costos, tener mayor comodidad, y reducir la contaminación ambiental.

3.2. Mapas, rutas y cotizaciones

Utilizaremos el mapa de la ciudad de Guayaquil, donde simularemos una Red oportunista con personas portadoras de la aplicación caminando, en automóviles, buses interprovinciales y dentro del sistema de transporte Metrovía. Como mencionamos anteriormente la simulación la realizaremos con la herramienta The ONE. Las rutas las obtuvimos del sitio Web Google Maps. Observar la Figura 3.1 - Mapa de la ciudad de Guayaquil.

Los valores de los pasajes lo obtuvimos desde la página Web del terminal terrestre de Guayaquil, lo exponemos en la Tabla 1 – Cotizaciones de viajes desde Guayaquil Terminal Terrestre. Estas cotizaciones nos servirán como referencia en el momento de ingresar datos de precios en las pruebas de la aplicación del conductor.

Tabla 1 – Cotizaciones de viajes desde Guayaquil Terminal Terrestre [40].

De Guayaquil a	Empresa	Valor en Dólares
Salinas	C.I.C.A.	3,31
Manta	Coactur	3,8
Baños	Riobamba	8,55
Cuenca	Express Sucre	4,7
Quito	Transportes Ecuador	8,15

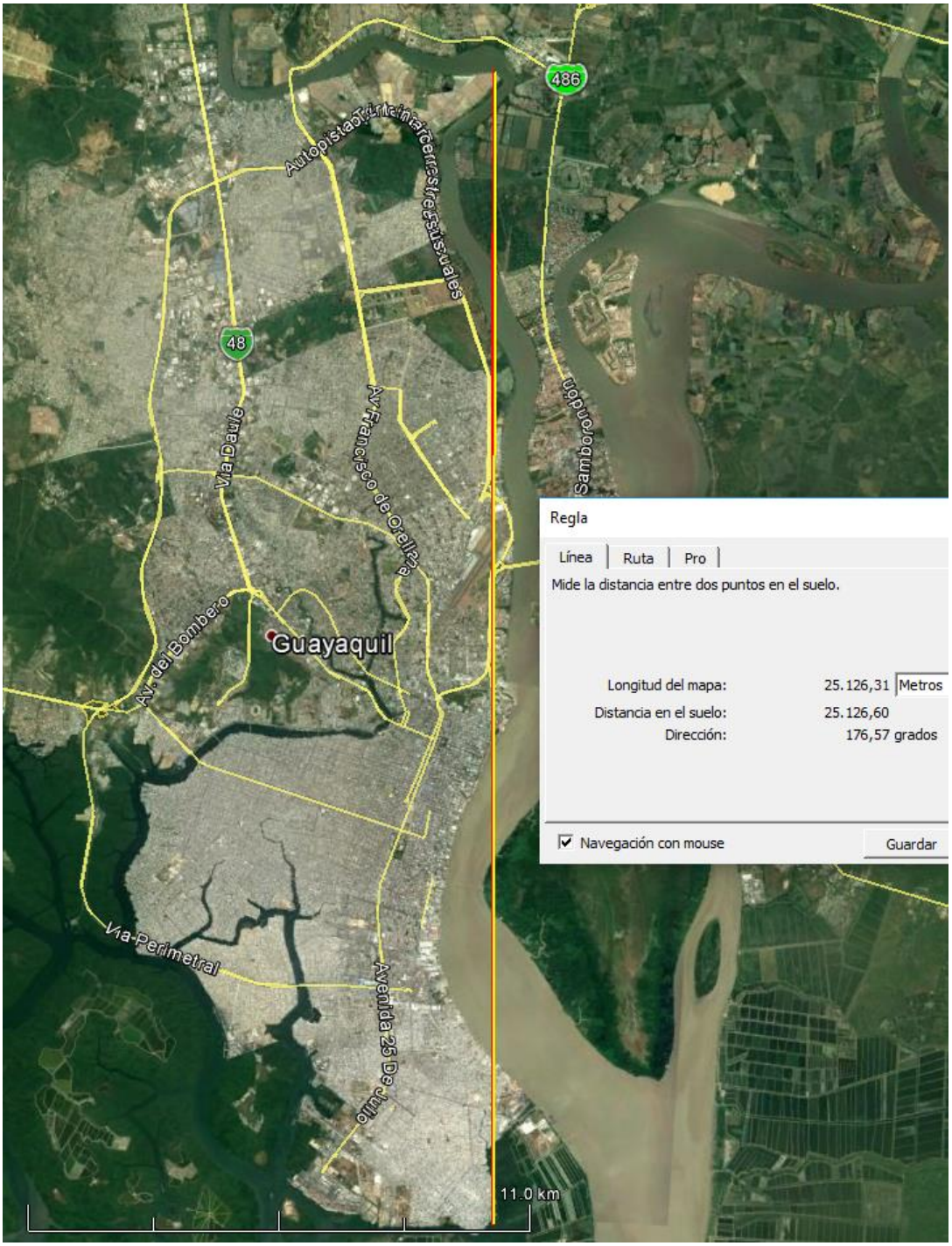


Figura 3.1 - Mapa de la ciudad de Guayaquil [41].

3.3. Herramientas para la simulación de Redes oportunistas

La simulación cumple un rol importante en el análisis del comportamiento y rendimiento del enrutamiento DTN y la aplicación de protocolos, por lo tanto es crucial elegir un simulador que nos permita analizar el comportamiento de nuestra Red oportunista ya sea en comunidades grandes o con el uso de diferentes protocolos de enrutamiento DTN según lo que demande el diseño de nuestro esquema distribuido.

Existen cuatro simuladores DTN para la mayoría de escenarios, estos simuladores son NS2, OMNET++, DTNSim, y The ONE. NS2 y OMNET++ son simuladores maduros, de Redes en general, proveen una plataforma de simulación genérica abierta para comunicaciones basadas en paquetes. NS2 posee varias herramientas para la implementación de simulaciones de Redes MANET. El simulador OMNET++ tiene como objetivo especial la simulación de Redes tolerantes a retardos. Además OMNET++ posee un paquete de análisis estadístico que extiende considerablemente la capacidad del simulador. DTNSim creado por la Universidad de Waterloo es un simulador exclusivo para el enrutamiento DTN, donde se implementan

dos escenarios: enrutamiento a una aldea remota y una Red de autobuses urbanos [42].

The ONE (Opportunistic Networking Environment, por sus siglas en inglés) es un simulador para Redes oportunistas en entornos DTN con la capacidad de implementar algunos protocolos de enrutamiento típicos y modelos de movilidad en DTN, y además es fácil de extender. The ONE es capaz de generar movimientos de nodos utilizando diferentes modelos de movimientos y rutas reales, emulando una Red oportunista con soporte de algoritmos de enrutamiento DTN [17]. Ofrece un marco de referencia para la implementación de protocolos de enrutamiento y aplicación (incluye por defecto seis protocolos de enrutamiento ya conocidos). La visualización interactiva y herramientas de pos procesamiento dan soporte a evaluaciones experimentales y modelos de emulación permitiendo al simulador The ONE ser parte de pruebas DTN reales [43]. El simulador escogido para nuestro proyecto es The ONE, debido a la exclusividad, rendimiento y aproximación a la realidad al trabajar con Redes oportunistas, además de la escalabilidad de The ONE que nos permitirá extender ciertos parámetros para ajustarlo al enrutamiento que demanda el diseño de nuestro esquema distribuido.

Para la creación de mapas y rutas que se utilizaran en el simulador The ONE, se usará la herramienta OpenJUMP, ésta herramienta es un Sistema de Información Geográfica (GIS, por sus siglas en Inglés Geographic Information System) de código abierto escrita en JAVA [44]. Con este GIS manualmente crearemos las avenidas y calles principales de la ciudad a simular dibujando líneas sobre la figura de fondo del mapa de Guayaquil, al unir todas estas líneas y retirar la figura de fondo, obtendremos un mapa de rutas que al guardarlo producirá un archivo WKT (compatible con el simulador The ONE) con información geográfica en texto de las calles de la ciudad de Guayaquil. Finalmente este archivo WKT podrá ser cargado a la herramienta The ONE para que ésta sea capaz de simular el movimiento de los nodos sobre la ciudad.

3.4. Herramientas para la implementación de criptografía homomórfica

Para la implementación de nuestro mecanismo de protección de equidad del mercado basado en el criptosistema parcialmente homomórfico de Paillier disponemos de las siguientes librerías obtenidas del más grande repositorio de código abierto GitHub, donde encontramos librerías para la implementación del sistema criptográfico

de Paillier en lenguajes de programación JAVA, C y Python entre las más destacadas.

En JAVA disponemos de las siguientes librerías:

- THEP (The Homomorphic Encryption Project, por sus siglas en Inglés). Cálculo de datos cifrados para las masas. Este proyecto tiene como objetivo proporcionar librerías de cifrado homomórfico a los desarrolladores para permitir mayor privacidad y confidencialidad de software. Actualmente el código implementa el sistema de cifrado de Paillier en Java, junto con sus operaciones homomórficas y generación de llaves.

- NICTA/javallier. Una librería en JAVA para el cifrado parcialmente homomórfico basado en python-paillier [23].

- Kunerd/jpaillier. Una implementación en JAVA del criptosistema de Paillier, esta librería se desarrolló como parte del proyecto de investigación CoPPDA, (Corporate Privacy Preserving Data

Analysis, por sus siglas en inglés). Esta librería se desarrolló para la investigación y actualmente encuentra en un estado temprano y por lo tanto no listo para producción [45].

- Qstokkink/PHENet. Implementación de un esquema de Red experimental basado en el sistema de cifrado de Paillier [46].

En C disponemos de las siguientes librerías:

- Camillevuillaume/Paillier-GMP. Implementación del criptosistema homomórfico Paillier usando GMP [47].

- GerardGarcia/paillier-c. Implementación del criptosistema homomórfico de Paillier usando librerías OpenSSL [48].

- Habcast. Proyecto desarrollado en el Departamento de Ingeniería de comunicación e información de la Universidad Autónoma de Barcelona por los investigadores Adrián Sánchez-Carmona, Sergi Robles, Carlos Borrego, para la

implementación del criptosistema de Paillier en lenguaje de programación C, esta librería es utilizada en el proyecto PrivHab+ el cual utiliza el cifrado homomórfico para mantener la privacidad de la ubicación de nodos en una Red DTN, generando un protocolo de geo-enrutamiento seguro.

En Python disponemos de las siguientes librerías:

- Mikeivanov/Paillier. Se trata de una implementación pura y muy básica en Python del criptosistema homomórfico de Paillier. Esta implementación en Python posee aritmética de precisión arbitraria. La clave pública es serializable, por lo que puede ser recolectada, junto con los números cifrados y enviada a un servidor remoto para el cálculo. El código se basa libremente en THEP y algunas recetas de ActiveState. Se debe tener en cuenta que el propósito principal de esta aplicación es la educación; no es adecuada para el uso en producción [49].

- NICTA/python-paillier. Una biblioteca de cifrado parcialmente homomórfico en Python. Desarrollado en Data61 | CSIRO.

> Utiliza partes derivadas de la licencia Apache proyecto de Google: encrypted-bigquery-client. Este código no ha sido escrito ni examinado por algún experto en criptografía. Las partes criptográficas son afortunadamente cortas [50].

Finalmente, escogemos a Habcast, proyecto que ha sido probado en simulación y en la práctica en campo por el proyecto PrivHab+ demostrando ser eficiente al aplicar el cifrado homomórfico de Paillier. Disponemos del código fuente de Habcast permitiéndonos usar la operación de adición homomórfica, necesaria para realizar la resta de números cifrados utilizando la técnica de mapping.

CAPÍTULO 4

4. ANÁLISIS Y DISEÑO

En el siguiente capítulo analizaremos los protocolos de enrutamiento DTN a utilizar, los cuales deben cumplir con las características de una Red tolerante a retardo oportunista y la capacidad de guardar la privacidad de la localización de los nodos. A partir de esta información procederemos con el diseño de nuestro esquema distribuido basado en una Red oportunista DTN donde se expondrá la dinámica del enrutamiento para establecer una conexión oportunista entre cliente-servidor o en nuestro caso conductor-pasajero y viceversa. Luego procederemos con el análisis del esquema de cifrado homomórfico para el MPEM, utilizaremos la librería Habcast más funciones de mapping desarrolladas en este proyecto para el soporte de resta homomórfica de textos cifrados. Finalmente estableceremos el diseño

del MPEM basado en criptografía homomórfica en una Red oportunista, donde expondremos los casos de uso y la integración de la aplicación de cifrado homomórfico de Paillier a la Red oportunista mediante el esquema de intercambio de mensajes durante la ejecución del MPEM.

4.1. Análisis del algoritmo de enrutamiento tolerante a retardos (DTN)

La simulación cumple un rol importante en el análisis del comportamiento y rendimiento del enrutamiento DTN y la aplicación de protocolos, por lo tanto es crucial elegir un simulador que nos permita analizar el comportamiento de nuestra Red oportunista ya sea en comunidades grandes o con el uso de diferentes protocolos de enrutamiento DTN según lo que demande el diseño de nuestro esquema distribuido.

Como lo definimos en el Capítulo 3: Levantamiento de información, utilizaremos el Simulador The ONE, con el cual estaremos en la capacidad de simular una Red oportunista con enrutamiento DTN utilizando protocolos basados en replicación como Epidemic, PRoPHET o SprayAndWait. Los resultados de las simulaciones determinarán el protocolo de mayor desempeño y eficiencia para el

MPEM. A continuación analizaremos a cada uno de los protocolos para DTN mencionados.

Epidemic. Está basado en técnicas de inundación, cada nodo constantemente replica y transmite mensajes a nuevos contactos que aún no posean una copia del mensaje, una vez que este contacto recepta el mensaje, se lo denomina nodo infectado. Este protocolo incluye técnicas para limitar el número de mensajes transferidos, además se asegura de mantener sincronizadas sus bases de datos [51]. Este protocolo provoca saturación debido a que genera mucha sobrecarga y mantiene un gran número de copias de los mensajes en la Red [52].

PRoPHET. Es un nuevo protocolo de enrutamiento para DTN diseñado para mejorar la probabilidad de entrega de Bundles y reducir la sobrecarga y consumo de recursos de la Red. Si un nodo ha visitado una misma ubicación en varias ocasiones y ha generado encuentros con otros nodos en esta misma ubicación, es probable que este patrón se repita en el futuro, por lo tanto el protocolo guarda y aprende esta información de encuentros pasados para optimizar la entrega de Bundles. Cada nodo utiliza métricas probabilísticas denominadas

predictibilidad de entrega que son vectores utilizados para transferir los mensajes a un nodo confiable. Un valor alto de predictibilidad de entrega indica que ese nodo es más confiable que los otros nodos en la Red para reenviar un mensaje a su destino. PRoPHET genera menos sobrecarga a la Red, pero por otro lado posee un alto promedio de retrasos en comparación al enrutamiento Epidémico [52].

SprayAndWait. Enrutamiento basado en replicación, este protocolo limita la estrategia de envío de mensajes del enrutamiento Epidémico agregando un número máximo de copias permitidas de cada mensaje. Este protocolo consiste en dos fases, la primera, Spray, en la cual el nodo emisor envía un mensaje con un número límite N de copias permitidas en la Red, el emisor es el encargado de enviar una copia del mensaje a cada distinto retransmisor. La segunda fase, Wait, los nodos de retrasmisión receptan las copias, y esperan la transmisión directa del mensaje al nodo destino [52] [51].

Hemos analizado el funcionamiento de los tres protocolos de enrutamiento para DTN que podemos simular Epidemic, PRoPHET o SprayAndWait, por lo tanto determinamos lo siguiente: Descartamos el uso del protocolo PRoPHET debido a que utiliza y reenvía al resto

de nodos de la Red información del usuario como el histórico de contactos establecidos, patrones de localización y comportamiento de nodos para calcular la predictibilidad de entrega. Por lo tanto estamos infringiendo en la primicia de utilizar un protocolo de geo-enrutamiento seguro para DTN, con la cual necesitamos asegurar la privacidad de la localización, así como los comportamientos frecuentes de los nodos o usuarios en una Red oportunista.

Por otro lado tenemos a los protocolos de enrutamiento Epidemic y SprayAndWait, ambos basados en replicación, donde el protocolo SprayAndWait es una mejora del protocolo Epidemic debido a la técnica de limitar el número de copias en la Red, con el objetivo de no saturarla y consumir menos recursos. Ambos protocolos no utilizan información sobre la localización y comportamiento de los nodos vecinos por lo tanto entran en la primicia de utilizar un protocolo de geo-enrutamiento seguro que permita la privacidad de localización de los usuarios. Por lo tanto, SprayAndWait es un buen candidato a utilizar debido a las mejoras que posee sobre Epidemic, procederemos en el Subcapítulo 5.1 - Simulación del esquema distribuido en una Red oportunista. Al realizar las simulaciones con ambos protocolos de enrutamiento DTN Epidemic y SprayAndWait, compararemos

resultados, analizaremos y determinaremos cual pose mejor desempeño para nuestro MPEM.

4.2. Diseño de un esquema distribuido basado en una Red oportunista

Nuestra Red oportunista se simulará sobre el mapa de la ciudad de Guayaquil. Las rutas sobre las cuales correrá la simulación serán las principales avenidas y calles de la ciudad. Para crear el mapa de Guayaquil se utilizará la herramienta OpenJUMP la cual produce archivos WKT (con información geográfica del mapa en texto) compatible con el simulador The ONE. La Red oportunista se conformará por nodos, de los cuales existen 3 tipos: conductor, pasajero y retransmisor; además se definirá la velocidad a la que recorrerá cada nodo, el tipo de interface, alcance y velocidad de comunicación que utilizará así como el tipo de movimiento que realizará sobre el mapa de Guayaquil. Con respecto a la transferencia de mensajes en nuestra Red oportunista, se configurará a cada nodo un tamaño de Buffer y tiempo de vida por defecto de los mensajes que genera. Finalmente, los nodos ejecutarán un esquema de intercambio de mensajes utilizando el generador de eventos en el simulador The

ONE, con el uso de este generador de eventos simularemos el envío y recepción de un mensaje a través de nuestra Red oportunista DTN.

4.2.1. Rutas que conforman la Red oportunista

Para simular de mejor manera el flujo vehicular de la ciudad de Guayaquil se diseñan tres tipos de rutas, la primera corresponde a la ruta realizada por los “buses interprovinciales” (se supondrá que el sistema de buses interprovinciales soportan dispositivos de la Red oportunista DTN) que ingresan por el norte de la ciudad específicamente sobre la Vía Daule y terminan en el Terminal Terrestre de Guayaquil. La segunda ruta está conformada por los recorridos del Sistema de transporte Metrovía (se supondrá que el sistema de transporte Metrovía soporta dispositivos de la Red oportunista DTN), como la “Metrovía 25 de Julio” (Inicia en el Sur de la ciudad y termina en el Terminal Terrestre de Guayaquil) y la “Metrovía Río Daule” (Inicia en el sur de Guayaquil y termina en el Terminal Terrestre de Guayaquil). Por último, simularemos la ruta “Calles de Guayaquil”, esta comprende las dos rutas antes mencionadas (Buses interprovinciales y Metrovía) y el resto de avenidas y calles principales de la ciudad de Guayaquil.

Observemos la Figura 4.1 – Rutas a simular de la ciudad de Guayaquil. Donde se definen las rutas antes mencionadas con los siguientes colores:

- Azul, ruta de los buses interprovinciales
- Rojo, ruta Metrovía 25 de Julio
- Verde, ruta Metrovía Río Daule
- Blanco, amarillo, azul, rojo y verde, ruta Calles de Guayaquil

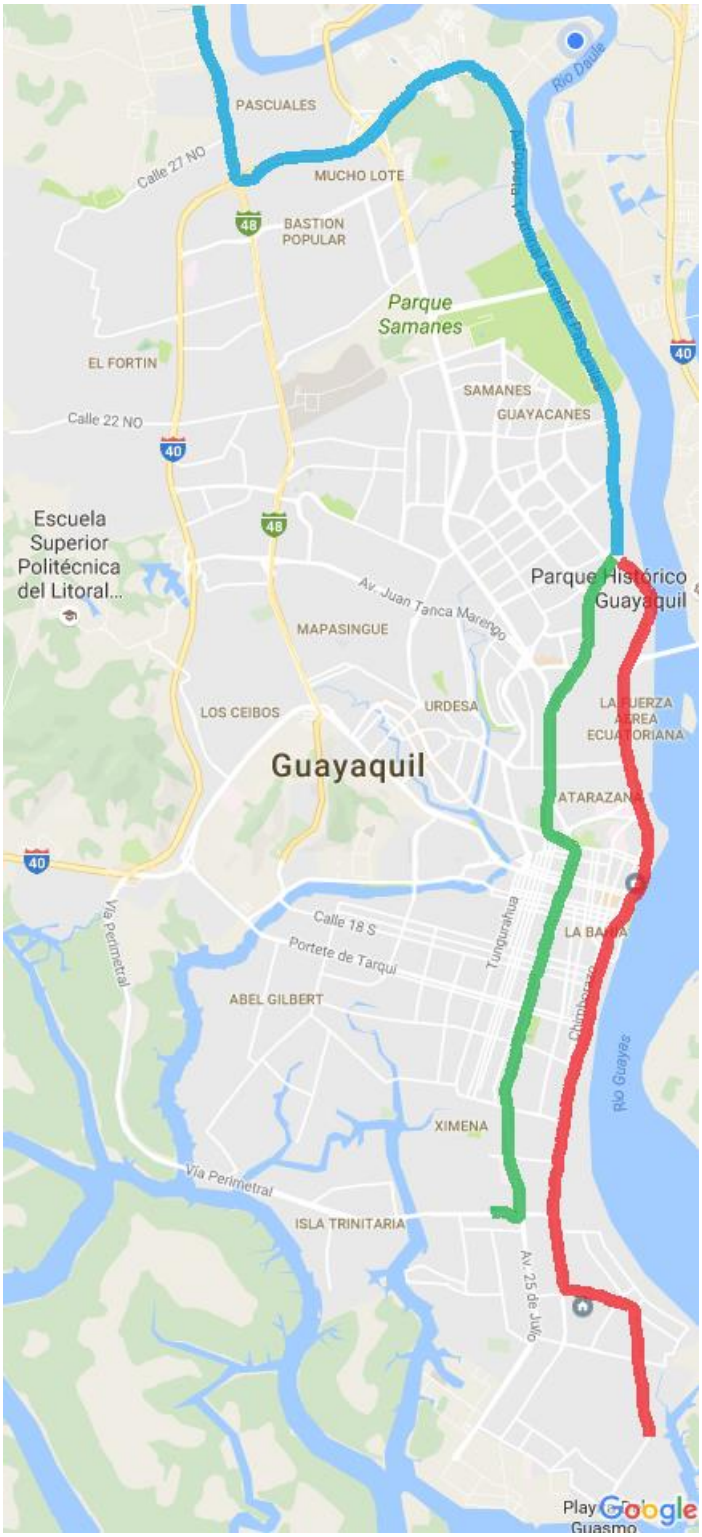


Figura 4.1 - Rutas a simular de la ciudad de Guayaquil [41].

4.2.2. Nodos que conforman la Red oportunista

El diseño de nuestro esquema distribuido estará conformado por 3 tipos de nodos, estos son los nodos conductor, pasajero y retransmisor; estos nodos se definen a continuación:

- Nodo conductor. Son aquellos vehículos cuyos movimientos se simularán a una velocidad entre 2.7 m/s y 13.9 m/s (10Km/h y 50Km/h respectivamente). Velocidad en promedio en la que los autos tienen permitido recorrer en la ciudad. Para simular su comportamiento se utilizará el modelo MapBasedMovement, este tipo de modelo permite a los nodos C (Carros) utilizar todas las avenidas y calles de la ciudad de Guayaquil.

- Nodo pasajero. Está dividido en 2 clases de pasajeros. Los posibles pasajeros que recorren a pie la ciudad, y los pasajeros que recorren la ciudad en auto.

  Los pasajeros que recorren a pie la ciudad forman una población baja en nuestra simulación. Para simular su comportamiento se utilizara el modelo

ShortestPathMapBasedMovement, este tipo de modelo permite a los nodos PP (Pasajeros a pie) utilizar todas las calles de la ciudad pero sólo podrán recorrer rutas cortas. La velocidad a la que se simulara es de 0.5 m/s a 1.5 m/s (1.8 Km/h a 5.4 Km/h) velocidad promedio de una persona.

Los pasajeros que recorren la ciudad en auto forman una población más grande que la de las PP. Para simular su comportamiento se utilizara el modelo MapBasedMovement, este tipo de modelo permite a los nodos PC (Pasajeros en carro) utilizar todas las calles de la ciudad de Guayaquil. La velocidad a la que se simulará es de 2.7 m/s y 13.9 m/s (10Km/h y 50Km/h respectivamente). Velocidad en promedio en la que los autos tienen permitido recorrer la ciudad.

- Nodo retransmisor. Son aquellos nodos que no emiten o generan mensajes, pero si reciben mensajes para entregarlos a los destinatarios. Estos nodos están conformados por los sistemas de transportes como la Metrovía, y los buses interprovinciales.

Para simular el comportamiento de los retransmisores se utilizará el modelo MapBasedMovement, este tipo de modelo permite a los nodos B (Buses interprovinciales), MJ (Metrovía 25 de Julio) y MR (Metrovía Río Daule) utilizar todas las calles pertenecientes a la ruta designada para cada clase de nodo retransmisor. La velocidad a la que se simularán los buses de la Metrovía es de 5.4 m/s a 17.5 m/s (20 Km/h y 63 Km/h respectivamente). Finalmente la velocidad a la que se simularán los buses interprovinciales es de 8.1 m/s a 18.9 m/s (30 Km/h y 70 Km/h respectivamente), velocidad permitida para los Buses interprovinciales en el camino hacia el Terminal terrestre de Guayaquil.

Todos los nodos contaran con una interfaz WiFi 802.11n, escogemos este tipo de interfaz debido a que es comercial y fácilmente asequible y además posee una mejor tasa de transferencia y alcance que los protocolos 802.11 a, b y g que son igualmente comerciales [53]. Este proyecto está orientado para tecnologías de software libre y de código abierto, por lo que si en un futuro si se implementaría se podría utilizar un dispositivo Raspberry Pi con adaptador inalámbrico Wi-Pi Raspberry Pi 802.11n el cual soporta hasta 150Mps de transferencia y 250m de alcance en exteriores [54]. O utilizar el último modelo de

tarjeta Raspberry Pi 3 model B, el cual posee integrado una interfaz WiFi 802.11n y soporte de tarjeta de almacenamiento externa MicroSDHC cuya capacidad puede ser de 2GB hasta 32GB [55].

Todos los nodos tendrán un tamaño de Buffer de 1GB para el almacenamiento de mensajes, el Gigabyte restante se lo dedicará al Sistema Operativo asumiendo que la tarjeta programable Raspberry PI posea una tarjeta de almacenamiento externa de 2GB. Dado que se simulará sobre una Red intermitente, tolerante a retardos, la cual presenta desafíos para la comunicación, nuestro esquema distribuido se verá limitado a una comunicación de dos pasos que involucra el envío de un mensaje por el emisor y la respuesta del mensaje por el receptor, en resumen, se comunicaran de la forma “ping pong”.

4.3. Análisis del esquema de cifrado homomórfico para el MPEM

Como se definió en el Subcapítulo 3.4 - Herramientas para la implementación criptografía homomórfica. Nuestro MPEM se desarrollará utilizando el lenguaje de programación C, las librerías del proyecto Habcast para el cifrado homomórfico y OpenSSL.

Con el proyecto Habcast, estamos en la capacidad de utilizar las siguientes funciones que pertenecen a la librería de pallier.h:

- **GenerateRandomKeys**. Función que genera las llaves de Paillier pública y privada.

- **EncryptII.** Función que permite cifrar los precios, el precio es ingresado en texto plano.

- **DecryptII**. Función que permite descifrar los precios, el precio descifrado se entregará en texto plano.

- **Encrypt.** Función que permite cifrar los precios. El precio ingresado debe ser un tipo de variable BIGNUM.

- **Decrypt**. Función que permite descifrar los precios, el precio descifrado se entregará en tipo de variable BIGNUM.

- **Mapping.** Función que permite mappear un precio (x + n), donde x = -b; x ∈ (-n/2,0).

```
int mapping(BIGNUM *result, const long long plain, const BIGNUM *n,
BN_CTX *ctx)
{
    int ret = 1;
    BN_CTX_start(ctx);

    BIGNUM *m = BN_CTX_get(ctx);

    if (!BN_set_word(m, plain))
        goto end;

    if (!BN_sub(result,n,m))
        goto end;

    ret = 0;
    end:
    if (ret)
    {
        ERR_load_crypto_strings();
        fprintf(stderr, "Can't mapping: %s",
ERR_error_string(ERR_get_error(), NULL));
    }

    BN_CTX_end(ctx);
    return ret;
}
```

- **Inv_mapping**. Función que permite regresar el mapeo de un precio (x - n), donde x = a - b + n; x ∈ (n/2,n-1).

```
int inv_mapping(long long *plain, BIGNUM *map, const BIGNUM *n,
BN_CTX *ctx)
{
    int ret = 1;
    BN_CTX_start(ctx);

    BIGNUM *result = BN_CTX_get(ctx);

    if (!BN_sub(result,map,n))
        goto end;

    *plain = BN_get_word(result);
    if (*plain == 0xffffffffL)
        goto end;

    ret = 0;
    end:
    if (ret)
    {
        ERR_load_crypto_strings();
        fprintf(stderr, "Can't mapping: %s",
ERR_error_string(ERR_get_error(), NULL));
    }
```

```
BN_CTX_end(ctx);
return ret;
```

- **Sub**. Función que permite restar dos números cifrados con Paillier, en nuestro caso los precios. El producto de la función Sub es un numero cifrado con el criptosistema de Paillier que al descifrarlo resultará en un entero positivo en el caso de que *a* > *b*, donde se opera *(a – b);* o “-1” en el caso de que *a* < *b*, donde se opera (a – b).

  Como se mencionó en subcapítulo 2.5.2. Propiedades homomórficas. La función de sustracción utiliza las librerías OpenSLL bn.h, bio.h, err.h, rand.h para obtener el inverso del minuendo en modulo *n^2* con la siguiente función: **BN_mod_inverse(b_inv, b, n2, ctx),** donde *b* es el minuendo, *n2* es *n^2* y *ctx* es una variable contenedor temporal de Bignums.

  Luego se procede con la multiplicación del sustraendo cifrado representado por *a* y el inverso del minuendo cifrado representado por *b_inv*, en modulo *n^2* para obtener la resta

homomórfica de Paillier representado por la variable *result* en la función: **BN_mod_mul(result, a, b_inv, n2, ctx).**

- **Add**. Función que permite sumar dos números cifrados con Paillier.

En este proyecto se implementó el artificio de mapping de un número en texto plano, para luego cifrarlo y sumarlo al minuendo, el resultado de esta suma se descifra y pasa a la función inv_mapping con la cual obtenemos la resta a partir de la suma de Paillier, es una técnica elegante para superar el problema de la sustracción homomórfica no contemplada en el esquema original de Paillier.

Para poder utilizar la librería de Paillier del proyecto Habcast, es necesario desarrollar el proyecto en lenguaje de programación C e instalar previamente las librerías OpenSSL y OpenSSL-devel. El uso de las funciones de cifrado homomórfico de Paillier se pueden observar con mayor detalle en el Anexo A – Código fuente del MEPM.

4.4. Diseño del MPEM basado en criptografía homomórfica en una Red oportunista

El MPEM correrá sobre una Red oportunista, por lo que está limitado a realizar una comunicación de ping pong entre el nodo conductor y pasajero. Se implementará una aplicación para el conductor y otra para el pasajero, la interacción de estas dos aplicaciones conforman nuestro protocolo basado en criptografía homomórfica que más la Red oportunista DTN crean nuestro esquema distribuido formando el Mecanismo de protección de equidad del mercado - MPEM. Para cumplir con el objetivo de ocultar ofertas y calcular la cotización más baja, el MPEM trabajará con los siguientes casos de uso y esquemas de intercambio de mensajes. Cabe recalcar que durante todos los procesos del MPEM ningún pasajero sabrá cuánto ofertó el conductor, y ningún conductor podrá saber cuánto fue el límite a pagar por el pasajero, solo podrá descifrar la operación final el nodo que posea la llave privada de Paillier.

4.4.1. Caso de uso por oferta de viaje del conductor

En este caso, el conductor se dispone a viajar y desea compartir su automóvil, el intercambio de mensajes para este caso de uso

comprende dos fases sobre las cuales se desarrollan los siguientes procesos:

Fase 1. La aplicación del conductor generará las llaves pública y privada de Paillier. Con la llave pública se cifrará el precio del viaje. Se enviará a todos los pasajeros un mensaje que contendrá información de la fecha, lugar de origen, lugar de destino, id del conductor, número de plazas, el precio del viaje cifrado y la llave pública de Paillier.

Fase 2. Una vez que el mensaje alcanza al pasajero, la aplicación del pasajero cifrará el precio límite que el pasajero está dispuesto a pagar con la llave pública de Paillier enviada por el conductor, luego se procederá a restar el precio límite cifrado menos el precio del viaje cifrado, el resultado se enviará al conductor conjunto información de id y número de teléfono del pasajero a través de la Red oportunista.

Finalmente el conductor recibirá el contacto del pasajero, y el resultado de la resta. La aplicación del conductor descifrará el resultado de la resta y si el valor de la resta es “-1”, indicará que el precio del viaje sobrepasa al límite a pagar por el pasajero. Por otro lado, si el resultado es un número entero positivo, significa que el pasajero está

dispuesto a pagar un valor superior al ofertado por el conductor y el número de plazas del conductor se reducirá en uno. Para ambos resultados, por política el conductor deberá informarle al pasajero el resultado de la operación. La aplicación del conductor se cerrará una vez que el contador de plazas llegue a “0” o todas las demandas de los pasajeros hayan llegado al conductor. El esquema de intercambio de mensajes de este caso se lo puede observar en la Figura 4.2 – Esquema de intercambio de mensajes, conductor oferta viaje.

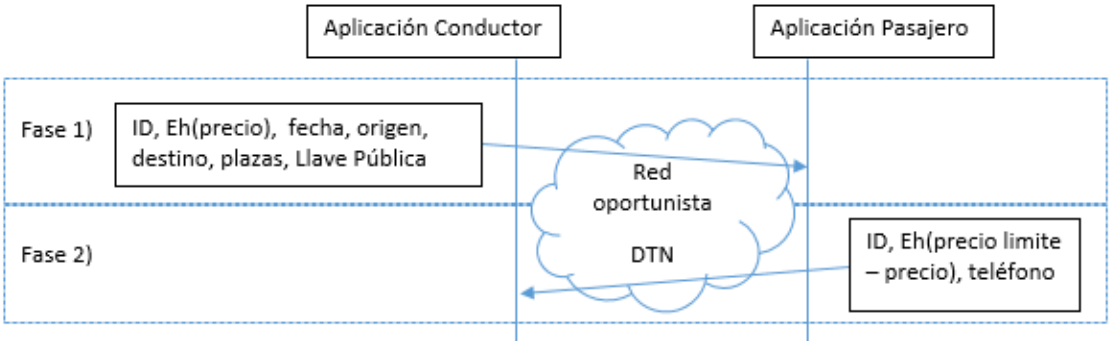


Figura 4.2 - Esquema de intercambio de mensajes, conductor oferta viaje.

4.4.2. Caso de uso por demanda de viaje del pasajero

En este caso, un pasajero desea viajar utilizando el MPEM para un sistema de automóvil compartido, por lo que se diseña el siguiente intercambio de mensajes que comprende dos fases:

Fase 1. La aplicación del pasajero generará las llaves pública y privada de Paillier, el pasajero enviará un mensaje a todos los conductores del MPEM, este mensaje contendrá información del pasajero como su id, precio límite cifrado a pagar por el viaje, fecha del viaje, origen, destino y la llave pública de Paillier.

Fase 2. El mensaje llegará a los conductores a través de la Red oportunista DTN, Los conductores utilizarán su aplicación para cifrar el precio del viaje con la llave pública enviada por el pasajero; luego responderán el mensaje con información de su id, número de teléfono, el resultado del precio límite a pagar cifrado menos el precio del viaje cifrado.

Finalmente el mensaje llegará al pasajero. La aplicación del pasajero procederá a descifrar el resultado de la resta del límite a pagar cifrado por el pasajero menos el precio del viaje cifrado por el conductor. Si el resultado es igual a “-1”, entonces el precio del viaje de ese conductor es considerado elevado, si el resultado es un entero positivo, se agregará a la lista de precios de conductores para luego utilizar una función que identificará el valor más bajo para el viaje y la identificación del conductor que ofreció dicho valor. El esquema de

intercambio de mensajes para este caso se lo puede observar en la Figura 4.3 – Esquema de intercambio de mensajes, pasajero demanda viaje.

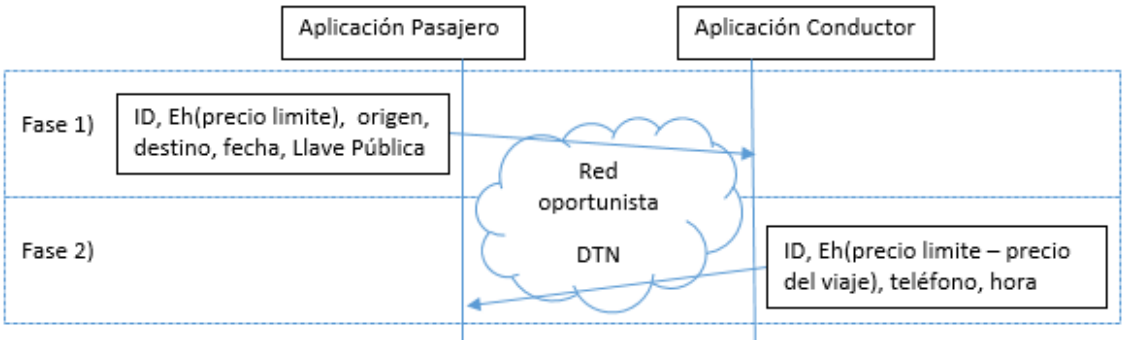


Figura 4.3 - Esquema de intercambio de mensajes, pasajero demanda viaje.

4.4.3. Diagrama de flujo de la aplicación del conductor

En base a las casos de uso establecidos para el Conductor en los Subcapítulos 4.4.1. Caso de uso por oferta de viaje del conductor y 4.4.2. Caso de uso por demanda de viaje del pasajero, se diseñó los diagramas de flujo para la aplicación conductor caso conductor y caso pasajero; donde se determinan los procesos que deberá realizar la aplicación conductor en el MPEM para el caso de uso del conductor y caso de uso del pasajero. Los procesos que ejecutará la aplicación conductor para el caso de uso del conductor pueden ser observados

en la Figura 4.4 – Diagrama de flujo de la aplicación conductor caso conductor.

Los procesos que ejecutará la aplicación conductor para el caso de uso del pasajero pueden ser observados en la Figura 4.5 – Diagrama de flujo de la aplicación conductor caso pasajero.

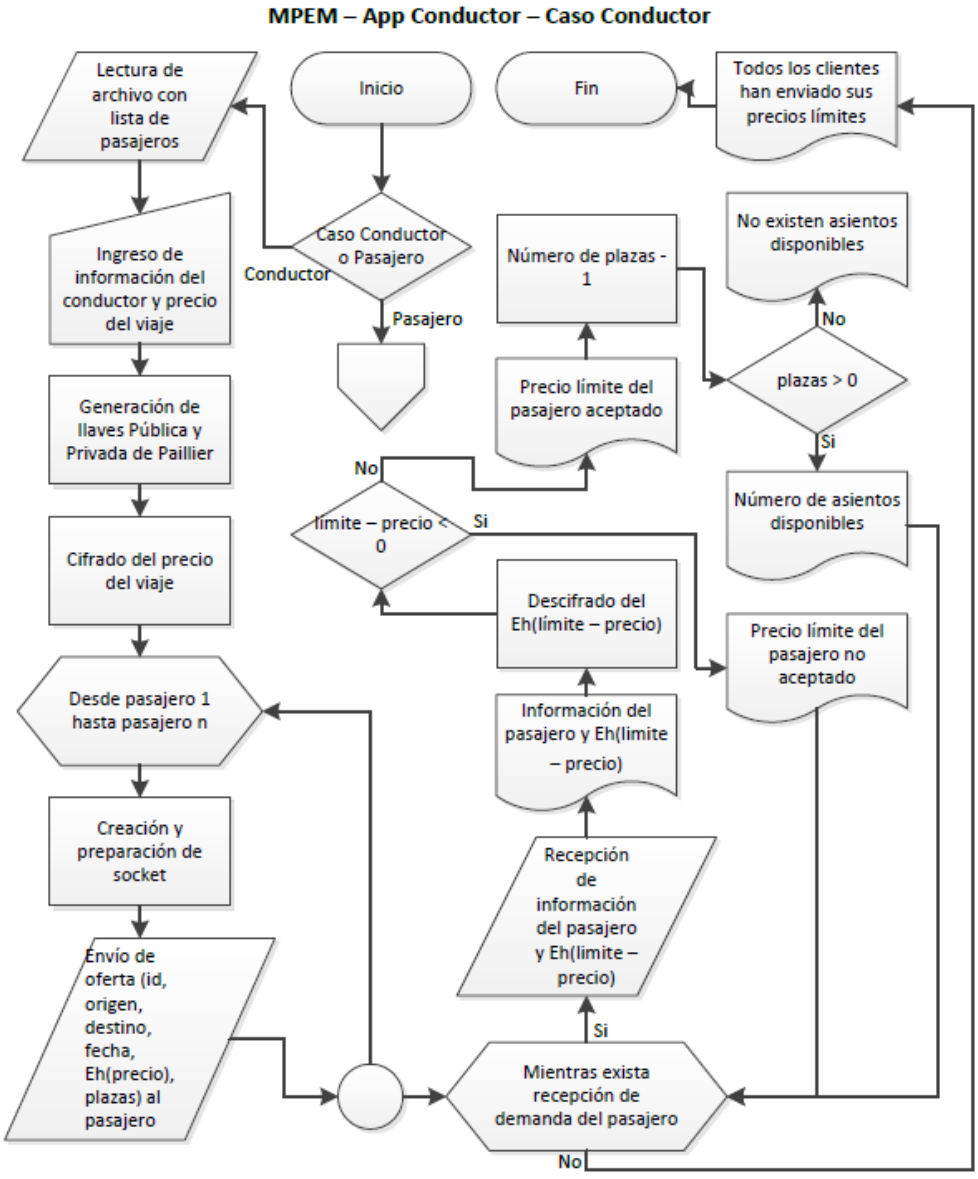


Figura 4.4 - Diagrama de flujo de la aplicación conductor caso conductor.

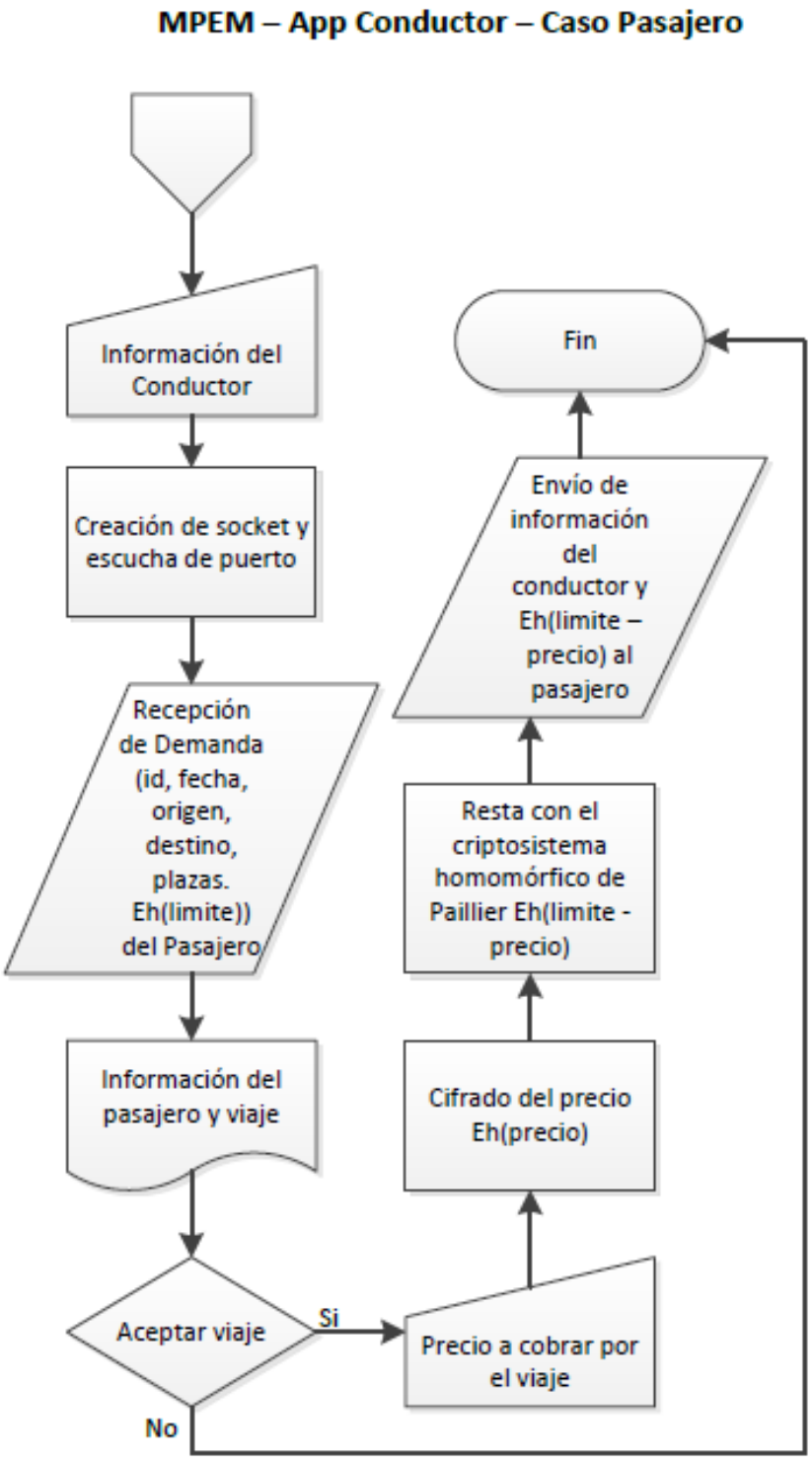


Figura 4.5 - Diagrama de flujo de la aplicación conductor caso pasajero.

4.4.4. Diagrama de flujo de la aplicación del pasajero

En base a las casos de uso establecidos para el pasajero en los Subcapítulos 4.4.1. Caso de uso por oferta de viaje del conductor y 4.4.2. Caso de uso por demanda de viaje del pasajero. Se diseñó los diagramas de flujo para la aplicación pasajero caso conductor y caso pasajero; donde se determinan los procesos que deberá realizar la aplicación pasajero en el MPEM para el caso de uso del conductor y caso de uso del pasajero. Los procesos que ejecutará la aplicación pasajero para el caso de uso del conductor pueden ser observados en la Figura 4.6 – Diagrama de flujo de la aplicación pasajero caso conductor.

Los procesos que ejecutará la aplicación pasajero para el caso de uso del pasajero pueden ser observados en la Figura 4.7 – Diagrama de flujo de la aplicación pasajero caso pasajero.

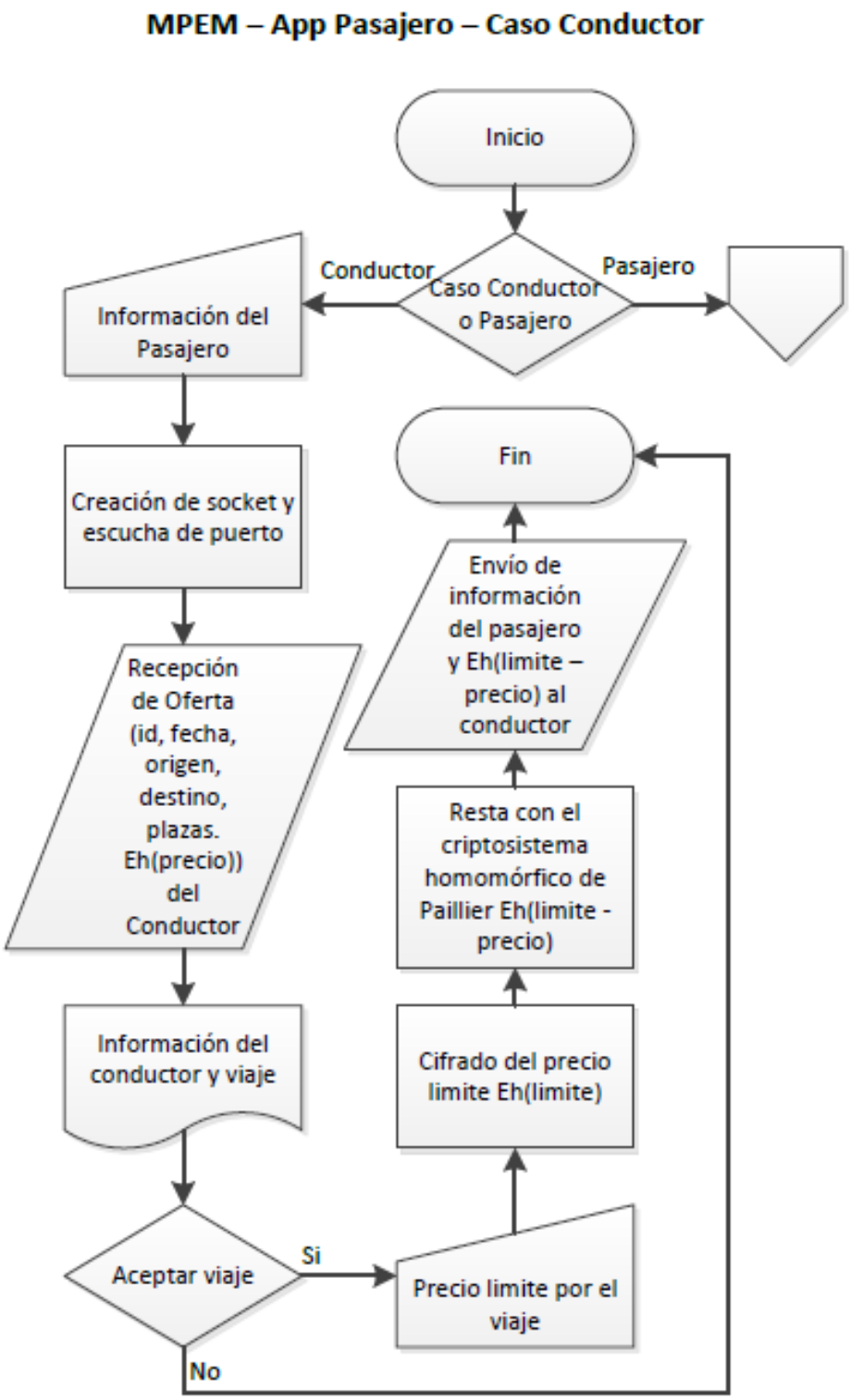


Figura 4.6 - Diagrama de flujo de la aplicación pasajero caso conductor.

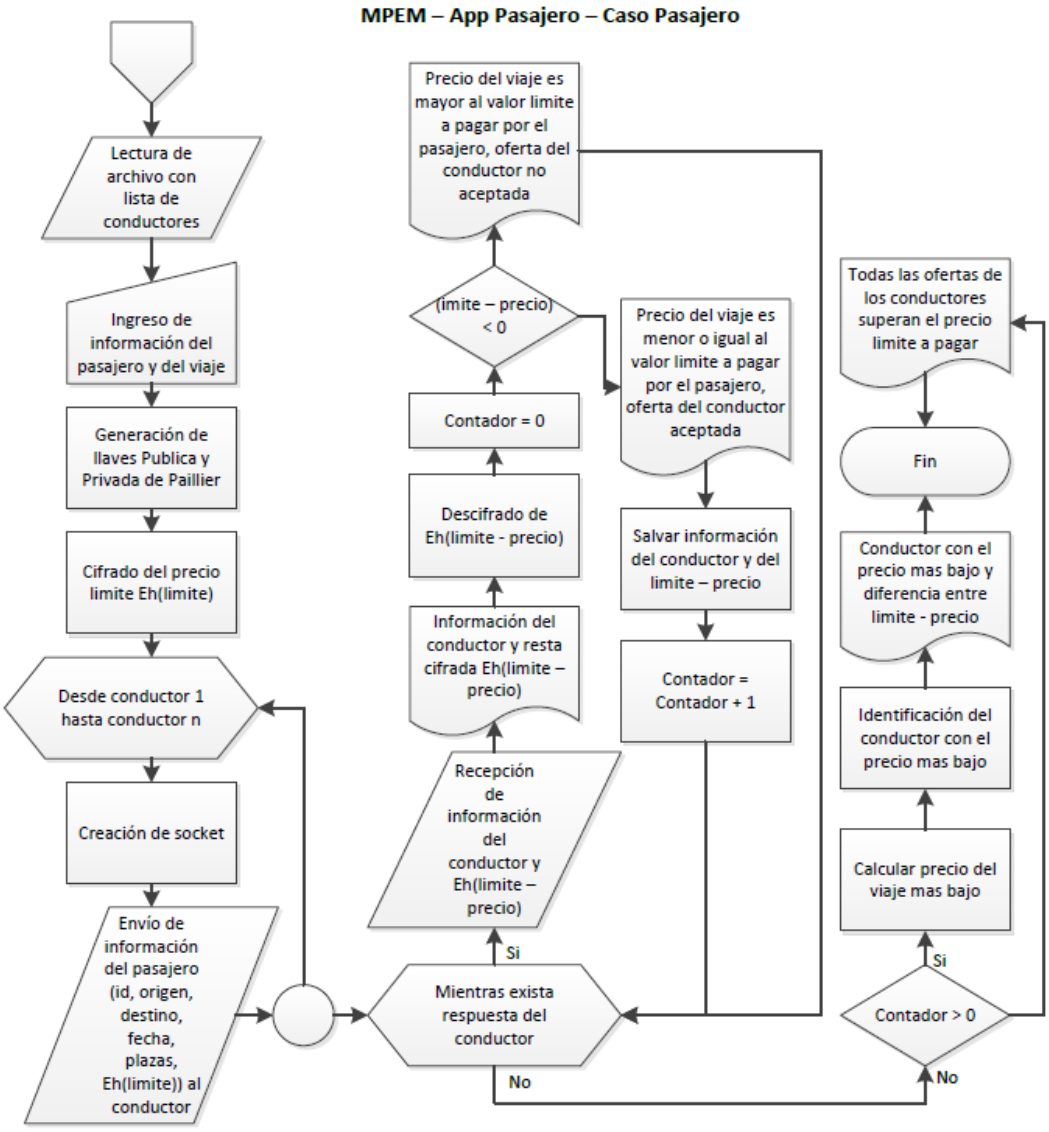


Figura 4.7 - Diagrama de flujo de la aplicación pasajero caso pasajero.

CAPÍTULO 5

5. SIMULACIÓN, IMPLEMENTACIÓN Y PRUEBAS

En el siguiente capítulo simularemos nuestro esquema distribuido y aplicaremos los dos casos de uso mencionados en el Subcapítulo 4.4 Diseño del MPEM basado en criptografía homomórfica en una Red Oportunista. El Sistema Operativo sobre el cual se desarrollará y correrá el MPEM es un CentOS 7 x86_64, Sistema Operativo de software libre soportado por la comunidad de desarrolladores, enfocado a entregar un ecosistema robusto de código abierto y una plataforma manejable y consistente que se adapta a una amplia variedad de implementaciones [56].

En este capítulo expondremos la configuración de parámetros de la simulación en la herramienta The ONE, parámetros correspondientes a la velocidad, movimiento, rutas, alcance, tasa de transferencia de los nodos, y generador de mensajes para el esquema de intercambio de mensajes entre los conductores y pasajeros. Luego de la simulación de la Red oportunista DTN, procederemos con la implementación del MPEM, donde expondremos el desarrollo de las aplicaciones para el conductor y pasajero basados en el criptosistema homomórfico de Paillier para realizar operaciones matemáticas con datos cifrados, finalizando con la ejecución de las aplicaciones y prueba de los casos de uso.

5.1. Simulación del esquema distribuido en una Red oportunista

La simulación de la Red oportunista se realizará sobre el mapa de Guayaquil definido en la Figura 4.1 - Rutas a simular de la ciudad de Guayaquil. Para ello utilizamos la herramienta OpenJump con la cual trazamos las rutas a simular utilizando de fondo el mapa de la ciudad, manteniendo la proporción y área de Guayaquil, como se puede observar en la Figura 5.1 – OpenJump, rutas a simular de las calles de Guayaquil. Comparece la proporción y tamaño con la Figura 3.1 - Mapa de la ciudad de Guayaquil.

La herramienta GIS Geographic Information System OpenJump la podemos descargar desde la página http://www.openjump.org/, esta es una herramienta de código abierto, su instalación es sencilla, bajo la plataforma Linux, para instalarlo basta con ejecutar el ./OpenJUMP-Installer-*.jar que has descargado y se creará un icono en el menú de programas desde el cual se pude abrir OpenJump.

Luego de instalar OpenJump procedemos a trazar todas las rutas dentro de un proyecto que al guardarlo genera un archivo en formato WKT por cada ruta. Cada archivo contiene información geográfica en texto de las rutas trazadas. Estos archivos nos servirán para el desplazamiento de los nodos en la simulación. Los archivos generados son: metro_riodaule.wkt, metro_25julio.wkt, buses_interprovinciales.wkt, calles_gye.wkt, el grafico de estos archivos se pueden observar en la Figura 5.1 – OpenJump, rutas a simular de las calles de Guayaquil.

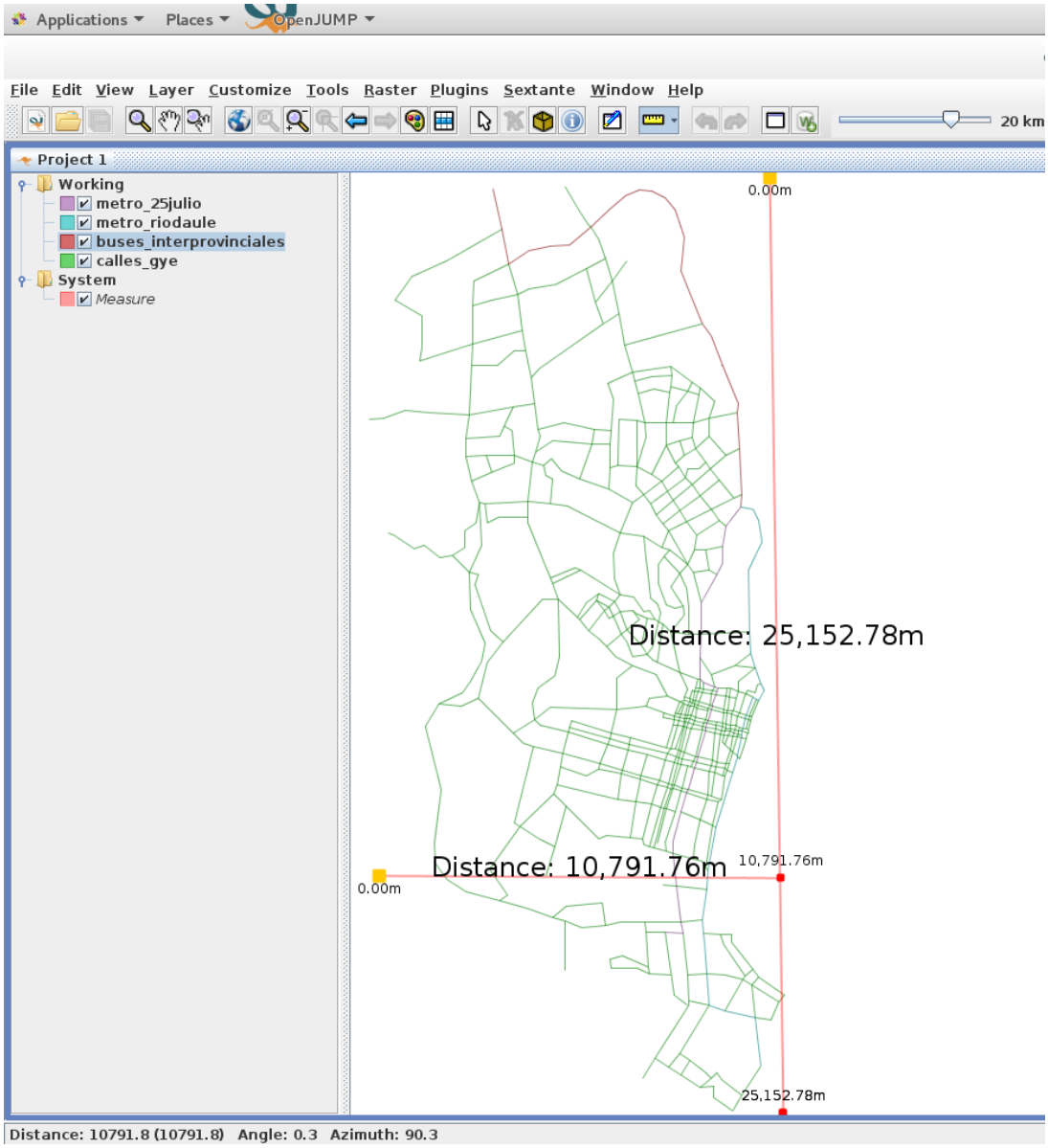


Figura 5.1 - OpenJump, rutas a simular de las calles de Guayaquil.

Como lo definimos en capítulos anteriores, el simulador para Redes DTN a utilizar es The ONE, para obtenerlo procedemos a descargarlo desde la página web https://akeranen.github.io/the-one/, la herramienta se

encuentra comprimida en formato tarball tar.gz, se la deberá descomprimir, y luego compilar ejecutando el script compile.sh, finalmente para correr el simulador se debe ejecutar el script one.sh.

El script one.sh lee un archivo de configuración para cargar la simulación, si no se envía parámetros en la ejecución de script, este cargará el archivo de configuración por defecto default_setings.txt. Para realizar la simulación del MPEM se crearon ocho archivos de configuración, cuatro para cada caso de uso. Del primero al cuarto archivo de configuración corresponden al caso conductor, el quinto al octavo archivo de configuración corresponden al caso pasajero.

Las simulaciones cubren la ejecución MPEM en tiempos continuos para todos los conductores y pasajeros, con lo cual buscamos observar y analizar cómo se desempeña nuestro MPEM en una Red oportunista a su máxima capacidad de uso. Por lo tanto en las simulaciones 1 a 4 todos los conductores al mismo tiempo enviarán solicitudes a todos los pasajeros y viceversa para las simulaciones 5 a 6. Las configuraciones de todas las simulaciones en The ONE se encuentran con mayor detalle en el Anexo B – Configuración de simulaciones del MPEM sobre una Red Oportunista.

5.1.1. Configuración por defecto de las simulaciones

Los archivos de configuración poseen los siguientes parámetros por defecto: Observar las tablas del 2 al 10.

La simulación correrá en un tiempo de 43200 segundos correspondientes a 12 horas. Tiempo suficiente para que el esquema de intercambio de mensaje se complete o algún mensaje sea descartado. El tiempo es razonable para la espera de respuesta por parte del usuario. Observar la Tabla 2 – Configuración por defecto del escenario.

Configuración por defecto de escenario

Tabla 2 – Configuración por defecto del escenario.

Parámetro	Definición
Scenario.endTime = 43200	Tiempo de simulación en segundos
Scenario.nrofHostGroups = 6	Número de grupos de nodos

Como se determinó en el Subcapítulo 4.2. Diseño de un esquema distribuido basado en una Red Oportunista, la interfaz de Red a utilizar en la simulación es una interface WiFi la cual trabajará con el protocolo 802.11n, otorgando 150Mps de transferencia y 250m de alcance en

exteriores. Observar la Tabla 3 - Configuración por defecto de interfaz de Red.

Configuración por defecto de interfaz de Red

Tabla 3 - Configuración por defecto de interfaz de Red.

Parámetro	Definición
WiFi80211.type = SimpleBroadcastInterface	Definición de una interface
WiFi80211.transmitSpeed = 18750k	Velocidad de tasa de trasferencia en KB
WiFi80211.transmitRange = 250	Alcance (metros)

Conforme a lo definido para el esquema distribuido de la Red oportunista, todos los nodos tendrán disponible 1GB de tamaño de Buffer para el almacenamiento de mensajes, cada mensaje tendrá un tiempo de vida de 300 y el modelo de movimiento a emplear es MapBasedMovement, lo que le permite a los nodos moverse por todo el mapa. Observar la Tabla 4 - Configuración por defecto de grupos de nodos.

Configuración por defecto de grupos de nodos

Tabla 4 - Configuración por defecto de grupos de nodos.

Parámetro	Definición
Group.movementModel = MapBasedMovement	Modelo de movimiento por defecto de los nodos
Group.bufferSize = 1024MB	Tamaño de Buffer
Group.waitTime = 10, 30	Tiempo de espera mínimo y máximo (segundos) después de alcanzar el destino
Group.nrofInterfaces = 1	Numero de interfaces de cada nodo
Group.interface1 = WiFi80211	Interfaz a utilizar
Group.speed = 2.7, 13.9	Velocidades mínima y máxima (m/s) de movimiento en la ruta
Group.msgTtl = 300	TTL del mensaje

Los grupos correspondientes a las rutas Metrovía y buses interprovinciales, utilizarán el modelo de movimiento MapRouteMovement, el cual les permite moverse solo por la ruta definida para el grupo. Con el parámetro routeFile asignamos un mapa de movimiento al grupo. En todas las simulaciones se toma como número de nodos retransmisores 25, distribuidos en 10 de Metrovía Río Daule, 10 Metrovía 25 de Julio y 5 buses interprovinciales. Observar la Tabla 5 - Configuración por defecto de cada grupo de nodo.

Configuración por defecto de cada grupo de nodo

Tabla 5 - Configuración por defecto de cada grupo de nodo.

Parámetro	Definición
Group1.groupID = c	Prefijo del grupo
Group1.movementModel = MapBasedMovement	Modelo de movimiento del nodo
Group1.speed = 2.7, 13.9	
Group2.groupID = pp	Prefijo identificador del grupo
Group2.movementModel = ShortestPathMapBasedMovement	
Group2.speed = 0.5, 1.5	
Group3.groupID = pc	
Group3.movementModel = MapBasedMovement	
Group3.speed = 2.7, 13.9	
Group4.groupID = mr	
Group4.movementModel = MapRouteMovement	
Group4.routeFile = data/wkt/wkt_sim_gye/metro_riodaule.wkt	Ruta por la cual el nodo podrá moverse
Group4.routeType = 2	Tipo de ruta, la opción 2 es de ida y vuelta
Group4.speed = 5.4, 17.55	
Group4.nrofHosts = 10	Número de nodos en el grupo
Group5.groupID = mj	
Group5.movementModel = MapRouteMovement	
Group5.routeFile = data/wkt/wkt_sim_gye/metro_25julio.wkt	
Group5.routeType = 2	
Group5.speed = 5.4, 17.55	
Group5.nrofHosts = 10	Número de nodos en el grupo
Group6.groupID = b	
Group6.movementModel = MapRouteMovement	
Group6.routeFile = data/wkt/wkt_sim_gye/buses_interprovinciales.wkt	
Group6.routeType = 2	
Group6.speed = 8.1, 18.9	
Group6.nrofHosts = 5	Número de nodos en el grupo

Configuración por defecto de carga de los mapas

Tabla 6 - Configuración por defecto de carga de los mapas.

Parámetro	Definición
MapBasedMovement.nrofMapFiles = 4	Numero de mapas
MapBasedMovement.mapFile1 = data/wkt/wkt_sim_gye/calles_gye.wkt	Ubicación del mapa
MapBasedMovement.mapFile2 = data/wkt/wkt_sim_gye/metro_riodaule.wkt	
MapBasedMovement.mapFile3 = data/wkt/wkt_sim_gye/metro_25julio.wkt	

Configuración por defecto de reportes

Tabla 7 - Configuración por defecto de reportes.

Parámetro	Definición
Report.nrofReports = 2	Numero de reportes
Report.report1 = MessageStatsReport	Reporte de estadísticas de rendimiento
Report.report2 = PingAppReporter	Reporte de estadísticas de ping
Report.report3 = MessageGraphvizReport	Reporte de conexiones de nodos y trayectoria de mensajes

Configuración por defecto de imagen de mapa

Tabla 8 - Configuración por defecto de imagen de mapa.

Parámetro	Definición
GUI.UnderlayImage.fileName = data/images/guayaquil.png	Imagen de fondo
GUI.EventLogPanel.nrofEvents = 100	Número de eventos visibles en el panel

Configuración por defecto de dimensión de movimientos

Tabla 9 - Configuración por defecto de dimensión de movimientos.

Parámetro	Definición
MovementModel.worldSize = 90000, 90000	Tamaño de la simulación (ancho, altura, metros)
MovementModel.warmup = 1000	Tiempo (segundos) de movimiento de los nodos antes de la simulación

Configuración por defecto de enrutamiento SprayAndWait

Tabla 10 - Configuración por defecto de enrutamiento SprayAndWait.

Parámetro	Definición
SprayAndWaitRouter.nrofCopies = 10	Número de copias máximas para el enrutamiento SprayAndWay

5.1.2. Configuración de la simulación Caso Conductor

Para el Caso Conductor, se creó cuatro archivos de configuración, los dos primeros corresponden a la simulación del MPEM utilizando el protocolo de enrutamiento Epidemic (Simulación 1 y 2) y los dos últimos utilizan el protocolo de enrutamiento DTN SprayAndWait (Simulación 3 y 4), la diferencia entre la simulación 1 y 2, 3 y 4 son los números de hosts, el primer y tercer archivo de configuración mantiene una relación de 5 conductores y 20 pasajeros, 5:20, y el segundo y cuarto archivo de configuración corresponde a la relación 20 conductores y 80 pasajeros, 20:80. Con estas 4 configuraciones simularemos el esquema de intercambio de mensajes del MPEM para

determinar el rendimiento del esquema distribuido y el escalamiento en comunidades grandes de usuarios, cubriendo de esta manera uno de nuestros objetivos específicos.

Para la simulación 1, Caso Conductor 5:20 Epidemic se configuraron parámetros adicionales a los parámetros por defecto, los parámetros adicionales corresponden al Protocolo de enrutamiento DTN Epidemic, y número de hosts. Para la simulación 1 el total de hosts es 50 debido a la suma de 5 conductores, 20 pasajeros y 25 retransmisores.

El envío de los mensajes se lo simulará mediante la aplicación pingApplication, con la cual un conductor emitirá un ping hacia un pasajero y este responderá con un pong hacia el conductor, este envío de mensajes ping pong es satisfactorio para nuestro esquema de intercambio de mensajes de dos vías.

Todos los eventos generarán un mensaje cada 10 segundos, el tiempo máximo de generación de mensajes es de 209 segundos en 12 horas de simulación, lo que permite generar 20 mensajes por cada conductor, cada mensaje se enviará a un pasajero en particular. El

tamaño del mensaje de envío es de 1371 bytes (Conductor → Pasajeros), este tamaño se obtuvo de la captura de tráfico generado de las pruebas de la implementación del MPEM, la captura de trafico TCP se puede observar con mayor detalle en el Anexo C – Captura de trafico TCP entre las Aplicaciones conductor y pasajero del MPEM. El tráfico se capturó con la herramienta Wireshark, se observó que para el caso conductor el envío de la oferta de viaje por pate del conductor pesa 1346 bytes y la respuesta del pasajero pesa 410 bytes, a estos valores se les sumó 25 bytes los cuales pertenecen a la cabecera Bundle que utilizan los protocolos de comunicación DTN [57]. Observar la Tabla 11 - Configuración 1 Caso Conductor 5:20 Epidemic.

En el Caso Conductor 20:80 existen pequeños cambios. Primero se cambió el número de hosts en la simulación a 125, luego el número de hosts conductores y pasajeros, distribuidos en 20 conductores, 20 hosts pasajeros a pie y 60 hosts pasajeros en automóvil. Finalmente se configuro la aplicación pingApplication para que cada uno de los 20 conductores genere 80 mensajes destinados a los 80 pasajeros en intervalos de 10 segundos durante los primeros 809 segundos de la simulación. Observar la Tabla 12 - Configuración 2 Caso Conductor 20:80 Epidemic.

Configuración 1 Caso Conductor 5:20 Epidemic

Tabla 11 - Configuración 1 Caso Conductor 5:20 Epidemic.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_CONDUCTOR_EPIDEMIC_5_20	Nombre de la simulación
Group.router = EpidemicRouter	Protocolo DTN
Group.nrofHosts = 50	Numero de nodos
Events.nrof = 5	Numero de eventos generadores de mensajes
Group1.nrofHosts = 5	Numero de hosts conductores
Group2.nrofHosts = 5	Numero de hosts pasajeros a pie
Group3.nrofHosts = 15	Numero de hosts pasajeros en automóvil
pingApp.type = PingApplication	Definición de la aplicación PinApplication
pingApp.interval = 10	Intervalo de generación de pings
pingApp.destinationRange = 5,25	Rango de hosts de destino
pingApp.pingSize = 1371	Tamaño del ping
pingApp.pongSize = 435	Tamaño del pong
pingApp.passive = false	Genera ping y pong
pingApp.pingTime = 209.0000	Tiempo máximo de generación de pings
pingAppResponse.type = PingApplication	Definición de la aplicación PinApplication
pingAppResponse.pongSize = 435	Respuesta pong
pingAppResponse.passive = true	Genera solo pongs
Group1.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group1.application1 = pingApp	Asignación de una aplicación a un grupo
Group2.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group2.application1 = pingAppResponse	Asignación de una aplicación a un grupo
Group3.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group3.application1 = pingAppResponse	Asignación de una aplicación a un grupo
Report.report2 = PingAppReporter	Generación de reporte de pingApp

Configuración 2 Caso Conductor 20:80 Epidemic

Tabla 12 - Configuración 2 Caso Conductor 20:80 Epidemic.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_CONDUCTOR_EPIDEMIC_20_80	Nombre de la simulación
Group.nrofHosts = 125	Numero de nodos
Group.nrofHosts = 125	Numero de nodos
Events.nrof = 20	Numero de eventos generadores de mensajes
Group1.nrofHosts = 20	Numero de hosts conductores
Group2.nrofHosts = 20	Numero de hosts pasajeros a pie
Group3.nrofHosts = 60	Numero de hosts pasajeros en automóvil
pingApp.type = PingApplication	Definición de la aplicación PinApplication
pingApp.interval = 10	Intervalo de generación de pings
pingApp.destinationRange = 20,100	Rango de hosts de destino
pingApp.pingSize = 1371	Tamaño del ping
pingApp.pongSize = 435	Tamaño del pong
pingApp.passive = false	Genera ping y pong
pingApp.pingTime = 809.0000	Tiempo máximo de generación de pings
pingAppResponse.type = PingApplication	Definición de la aplicación PinApplication
pingAppResponse.pongSize = 435	Respuesta pong
pingAppResponse.passive = true	Genera solo pongs
Group1.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group1.application1 = pingApp	Asignación de una aplicación a un grupo
Group2.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group2.application1 = pingAppResponse	Asignación de una aplicación a un grupo
Group3.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group3.application1 = pingAppResponse	Asignación de una aplicación a un grupo
Report.report2 = PingAppReporter	Generación de reporte de pingApp

La configuración de la simulación 3 Caso Conductor 5:20 SprayAndWait es igual a la configuración de la simulación 1 Caso Conductor 5:20 Epidemic, excepto por el protocolo de enrutamiento a utilizar, el parámetro Group.router cambia a SprayAndWait. Observar la Tabla 13 - Configuración 1 Caso Conductor 5:20 SprayAndWait.

Configuración 3 Caso Conductor 5:20 SprayAndWait

Tabla 13 - Configuración 3 Caso Conductor 5:20 SprayAndWait.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_CONDUCTOR_SPRAYANDWAIT_5_20	Nombre de la simulación
Group.router = SprayAndWait	Protocolo DTN

La configuración de la simulación 4 Caso Conductor 20:80 SprayAndWait es igual a la configuración de la simulación 2 Caso Conductor 20:80 Epidemic, excepto por el protocolo de enrutamiento a utilizar, el parámetro Group.router cambia a SprayAndWait. Observar la Tabla 14 - Configuración 4 Caso Conductor 20:80 SprayAndWait.

Configuración 4 Caso Conductor 20:80 SprayAndWait

Tabla 14 - Configuración 4 Caso Conductor 20:80 SprayAndWait.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_CONDUCTOR_SPRAYANDWAIT_20_80	Nombre de la simulación
Group.router = SprayAndWait	Protocolo DTN

5.1.3. Configuración de la simulación Caso Pasajero

Para el Caso Pasajero, se creó cuatro archivos de configuración, los dos primeros corresponden a la simulación del MPEM utilizando el protocolo de enrutamiento Epidemic (Simulación 5 y 6) y los dos últimos utilizan el protocolo de enrutamiento DTN SprayAndWait (Simulación 7 y 8), la diferencia entre la simulación 5 y 6, 7 y 8 son los números de hosts, el primer y tercer archivo poseen una relación de 5 conductores y 20 pasajeros, 5:20, y el segundo y cuarto archivo de configuración corresponde a la relación 20 conductores y 80 pasajeros, 20:80. El Caso pasajero difiere con el Caso conductor en la generación de mensajes, en esta simulación los hosts pasajeros tendrán el rol de emisor del mensaje y los conductores serán los receptores.

La simulación 5 utiliza la configuración por defecto más el uso del protocolo de enrutamiento DTN Epidemic, el uso de la aplicación pingApplication para la generación de mensajes en intervalos de 10 segundos durante los primeros 59 segundos, lo que asegura la creación de 5 mensajes, un mensaje para cada conductor de los 5 registrados. Además se genera un reporte estadístico de pings y pongs creados y entregados. Esta configuración la podemos observar en la Tabla 15 - Configuración 5 Caso Pasajero 5:20 Epidemic.

Configuración 5 Caso Pasajero 5:20 epidemic

La simulación 6 utiliza la configuración por defecto más el uso del protocolo de enrutamiento DTN Epidemic y el uso de la aplicación pingApplication y su respectivo reporte estadístico. Esta configuración diferencia de la simulación 5 por la cantidad de hosts en la simulación donde encontramos una relación de conductores y pasajeros de 20:80, también usamos la aplicación pingApplication con la cual se generaran 20 mensajes en intervalos de 10 segundos durante los primeros 209 segundos, cada mensaje tendrá como destino un conductor de los 20 registrados. La configuración la podemos observar en la Tabla 16 - Configuración 6 Caso Pasajero 20:80 Epidemic.

Tabla 15 - Configuración 5 Caso Pasajero 5:20 Epidemic.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_PASAJERO_5_20_EPIDEMIC	Nombre de la simulación
Group.router = EpidemicRouter	Protocolo DTN
Group.nrofHosts = 50	Numero de nodos
Group1.nrofHosts = 5	Numero de hosts conductores
Group2.nrofHosts = 5	Numero de hosts pasajeros a pie
Group3.nrofHosts = 15	Numero de hosts pasajeros en automóvil
pingApp.type = PingApplication	Definición de la aplicación PinApplication
pingApp.interval = 10	Intervalo de generación de pings
pingApp.destinationRange = 0,5	Rango de hosts de destino
pingApp.pingSize = 1371	Tamaño del ping
pingApp.pongSize = 435	Tamaño del pong
pingApp.passive = false	Genera ping y pong
pingApp.pingTime = 59.0000	Tiempo máximo de generación de pings
pingAppResponse.type = PingApplication	Definición de la aplicación PinApplication
pingAppResponse.pongSize = 5	Respuesta pong
pingAppResponse.passive = true	Genera solo pongs
Group1.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group1.application1 = pingAppResponse	Asignación de una aplicación a un grupo
Group2.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group2.application1 = pingApp	Asignación de una aplicación a un grupo
Group3.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group3.application1 = pingApp	Asignación de una aplicación a un grupo
Report.report13 = PingAppReporter	Generación de reporte de pingApp

La simulación 7 utiliza la configuración por defecto más el uso del protocolo de enrutamiento SprayAndWait y el uso de la aplicación pingApplication y su respectivo reporte estadístico, esta configuración es similar a la configuración 5, diferenciando únicamente en el

parámetro router, la configuración que cambia la podemos observar en la Tabla 17 - Configuración 7 Caso Pasajero 5:20 SprayAndWait.

Configuración 6 Caso Pasajero 20:80 Epidemic

Tabla 16 - Configuración 6 Caso Pasajero 20:80 Epidemic.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_PASAJERO_20_80_EPIDEMIC	Nombre de la simulación
Group.router = EpidemicRouter	Protocolo DTN
Group.nrofHosts = 125	Numero de nodos
Group1.nrofHosts = 20	Numero de hosts conductores
Group2.nrofHosts = 20	Numero de hosts pasajeros a pie
Group3.nrofHosts = 60	Numero de hosts pasajeros en automóvil
pingApp.type = PingApplication	Definición de la aplicación PinApplication
pingApp.interval = 10	Intervalo de generación de pings
pingApp.destinationRange = 0,20	Rango de hosts de destino
pingApp.pingSize = 1371	Tamaño del ping
pingApp.pongSize = 435	Tamaño del pong
pingApp.passive = false	Genera ping y pong
pingApp.pingTime = 209.0000	Tiempo máximo de generación de pings
pingAppResponse.type = PingApplication	Definición de la aplicación PinApplication
pingAppResponse.pongSize = 5	Respuesta pong
pingAppResponse.passive = true	Genera solo pongs
Group1.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group1.application1 = pingAppResponse	Asignación de una aplicación a un grupo
Group2.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group2.application1 = pingApp	Asignación de una aplicación a un grupo
Group3.nrofApplications = 1	Número de aplicaciones a utilizar de un grupo
Group3.application1 = pingApp	Asignación de una aplicación a un grupo
Report.report13 = PingAppReporter	Generación de reporte de pingApp

Configuración 7 Caso Pasajero 5:20 SprayAndWait

Tabla 17 - Configuración 7 Caso Pasajero 5:20 SprayAndWait.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_PASAJERO_5_20_ SprayAndWait	Nombre de la simulación
Group.router = SprayAndWait	Protocolo DTN

La simulación 8 utiliza la configuración por defecto más el uso del protocolo de enrutamiento SprayAndWait y el uso de la aplicación pingApplication y su respectivo reporte estadístico, esta configuración es similar a la configuración 6, diferenciando únicamente en el parámetro router, la configuración que cambia la podemos observar en la Tabla 18 - Configuración 8 Caso Pasajero 20:80 SprayAndWait.

Configuración 8 Caso Pasajero 20:80 SprayAndWait

Tabla 18 - Configuración 8 Caso Pasajero 20:80 SprayAndWait.

Parámetro	Definición
Scenario.name = MPEM_GYE_CASO_PASAJERO_20_80_ SprayAndWait	Nombre de la simulación
Group.router = SprayAndWait	Protocolo DTN

5.2. Pruebas de la simulación

Para dar inicio a la simulación se debe ejecutar el script one.sh seguido del archivo de configuración que se desea cargar, ejemplo:

```
[root@serverone the-one]#./one.sh
data/settings/caso_conductor_pingpong_epidemic_5_20.txt
```

El script abrirá una ventana donde se podrá observar las rutas del mapa de la ciudad de Guayaquil y la cantidad de nodos configurados distribuidos en distintas posiciones en el mapa, cada nodo tendrá un prefijo seguido por un identificador único, además cada ping o pong generado tendrá como identificador el tiempo en que fue creado y el número de host emisor. Para identificar las rutas en que los nodos se moverán observemos la Figura 5.2 – Simulación del MPEM en una Red oportunista con The ONE, y la Figura 5.3 – Simulación del MPEM en una Red oportunista fondo de Guayaquil.

De los reportes generados en la simulación como PingAppReporter obtenemos estadísticas de ping, como pings creados, pings recibidos, pongs creados, pongs recibidos, probabilidad de entrega de pings y probabilidad de entrega de pongs. Del reporte MessageStatsReport obtenemos estadísticas de rendimiento como relación de sobrecarga, media de latencia, mediana de la latencia, media de cantidad de siguientes saltos, mediana de la cantidad de siguientes saltos, media del tiempo de almacenamiento de mensajes en el Buffer, mediana del tiempo de almacenamiento de mensajes en el Buffer, media del tiempo de ida y vuelta de una señal (RTT) y mediana del tiempo de ida y vuelta de una señal (RTT).

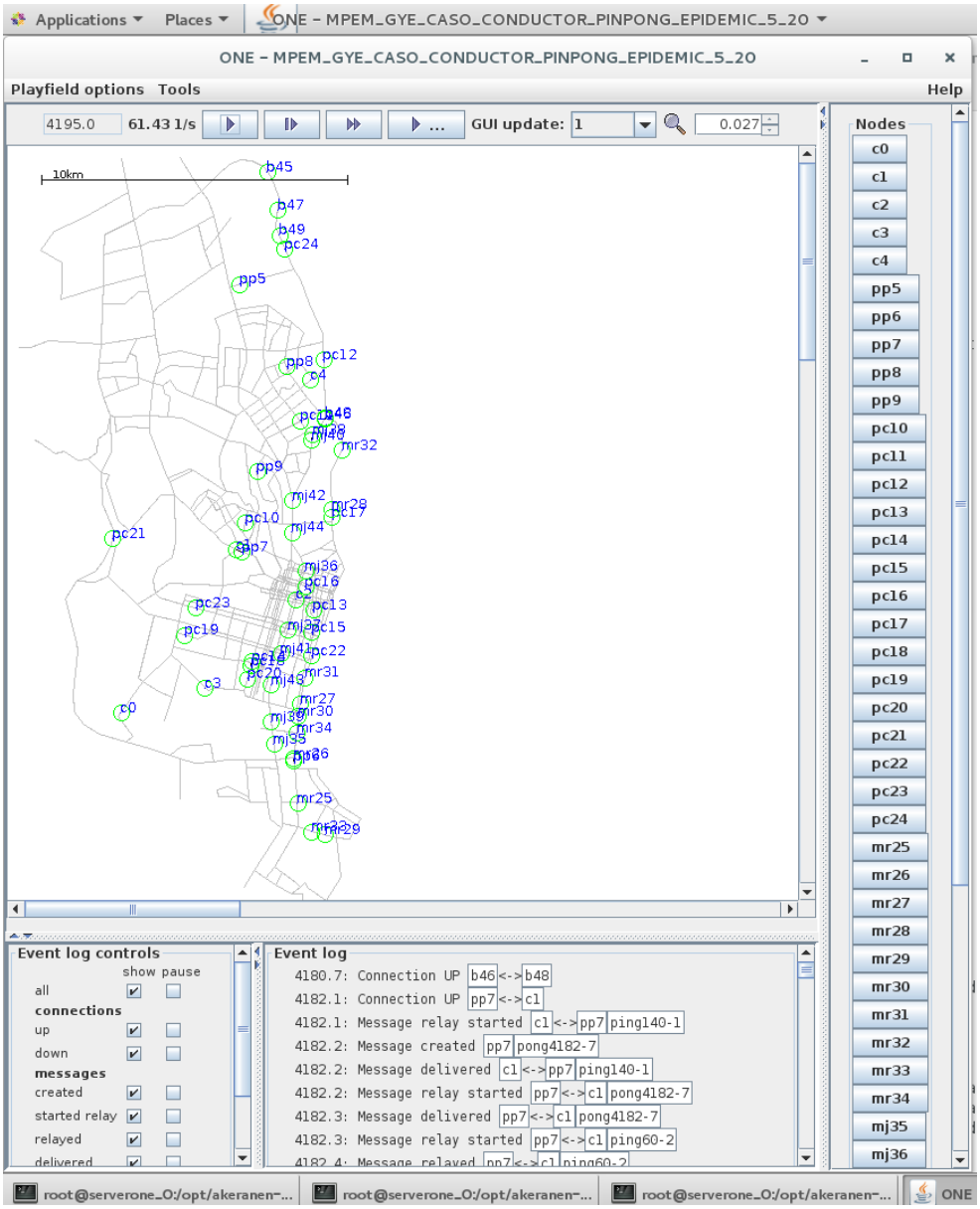


Figura 5.2 - Simulación del MPEM en una Red oportunista con The ONE.

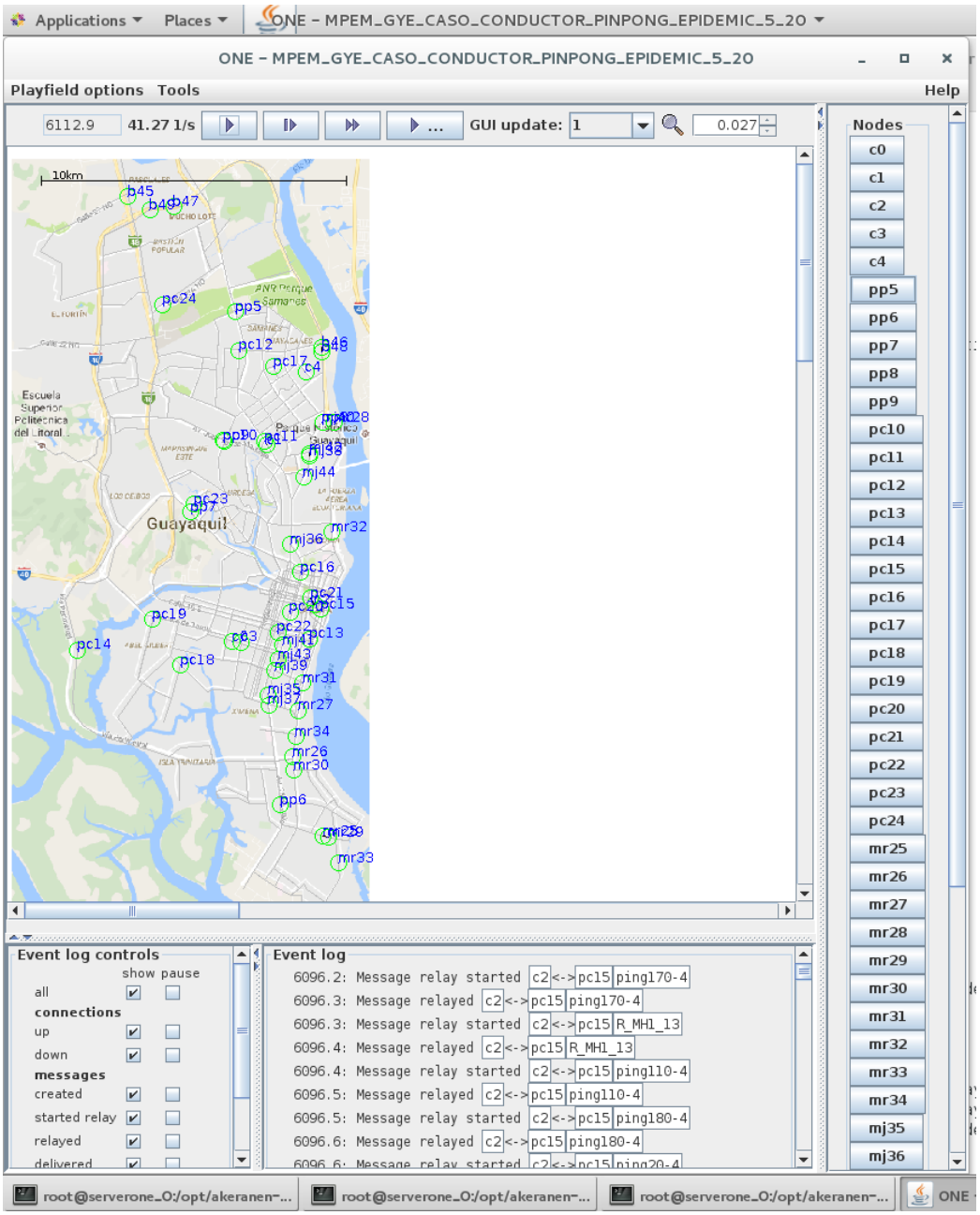


Figura 5.3 - Simulación del MPEM en una Red oportunista fondo de Guayaquil.

Además, para visualizar las conexiones de los nodos y las rutas que los mensajes han recorrido en la Red, generamos grafos a partir del reporte MessageGraphvizReport y la herramienta Graphviz [58], que es un software de visualización de gráficos de código abierto. A continuación muestro los resultados de las 8 simulaciones correspondientes a los Casos Conductor y Pasajero

5.2.1. Simulación 1 Caso Conductor Epidemic 5:20

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINPONG_EPIDEMIC_5_20
sim_time: 43200.1000
pings sent: 100
pings received: 100
pongs sent: 100
pongs received: 50
ping delivery prob: 1.0000
pong delivery prob: 0.5000
ping/pong success prob: 0.5000
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINPONG_EPIDEMIC_5_20
sim_time: 43200.1000
overhead_ratio: 48.0000
latency_avg: 3605.7984
latency_med: 2804.7000
hopcount_avg: 4.0895
```

```
hopcount_med: 3
buffertime_avg: 14341.6849
buffertime_med: 14748.4000
rtt_avg: 6651.3000
rtt_med: 5123.0000
```

En la Figura 5.4 - Grafo Simulación 1 Caso Conductor Epidemic 5:20 observaremos el grafo de las conexiones de los nodos y las rutas que los mensajes han recorrido en la Red. La Figura 5.4 se podrá observar con mayor detalle en el Anexo D – Grafos de las simulaciones.

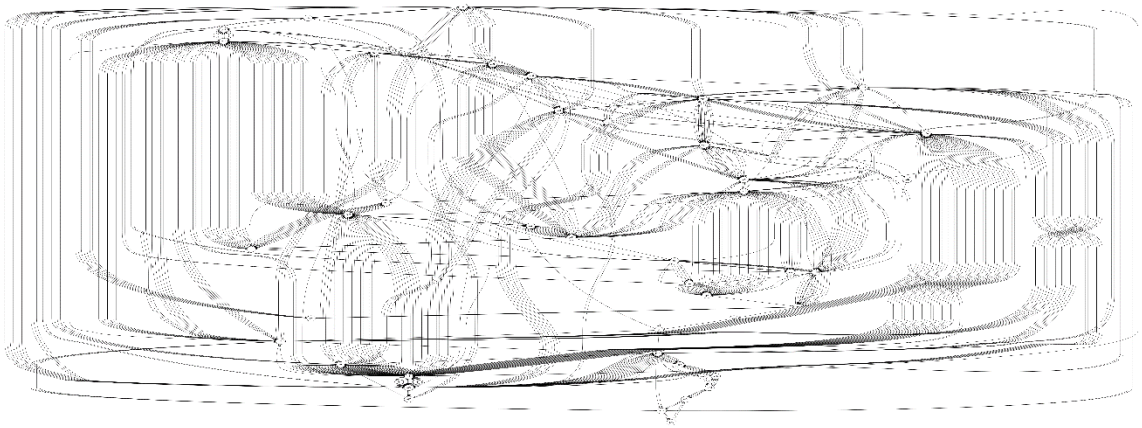


Figura 5.4 - Grafo Simulación 1 Caso Conductor Epidemic 5:20.

5.2.2. Simulación 2 Caso Conductor Epidemic 20:80

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINGPONG_EPIDEMIC_20_80
sim_time: 43200.1000
pings sent: 1600
pings received: 1600
pongs sent: 1600
pongs received: 673
ping delivery prob: 1.0000
pong delivery prob: 0.4206
ping/pong success prob: 0.4206
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINGPONG_EPIDEMIC_20_80
sim_time: 43200.1000
overhead_ratio: 123.0000
latency_avg: 2428.5147
latency_med: 1936.2000
hopcount_avg: 5.6354
hopcount_med: 5
buffertime_avg: 15373.3085
buffertime_med: 15827.9000
rtt_avg: 3239.1988
rtt_med: 2075.5000
```

Las múltiples conexiones entre los nodos y la gran cantidad de mensajes en la Red generan un grafo poco apreciable cuya interpretación de la Red pierde sentido, por lo tanto se determinó no representarla.

5.2.3. Simulación 3 Caso Conductor SprayAndWait 5:20

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINPONG_SAW_5_20
sim_time: 43200.1000
pings sent: 100
pings received: 85
pongs sent: 85
pongs received: 46
ping delivery prob: 0.8500
pong delivery prob: 0.5412
ping/pong success prob: 0.4600
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINPONG_SAW_5_20
sim_time: 43200.1000
overhead_ratio: 9.5802
latency_avg: 4882.0086
latency_med: 3868.7000
hopcount_avg: 2.3642
hopcount_med: 2
buffertime_avg: 15966.5109
buffertime_med: 17066.4000
rtt_avg: 7162.5400
rtt_med: 5122.9000
```

En la Figura 5.5 - Grafo Simulación 3 Caso Conductor SprayAndWait 5:20 observaremos el grafo de las conexiones de los nodos y las rutas que los mensajes han recorrido en la Red. La Figura 5.5 se podrá observar con mayor detalle en el Anexo D – Grafos de las simulaciones.

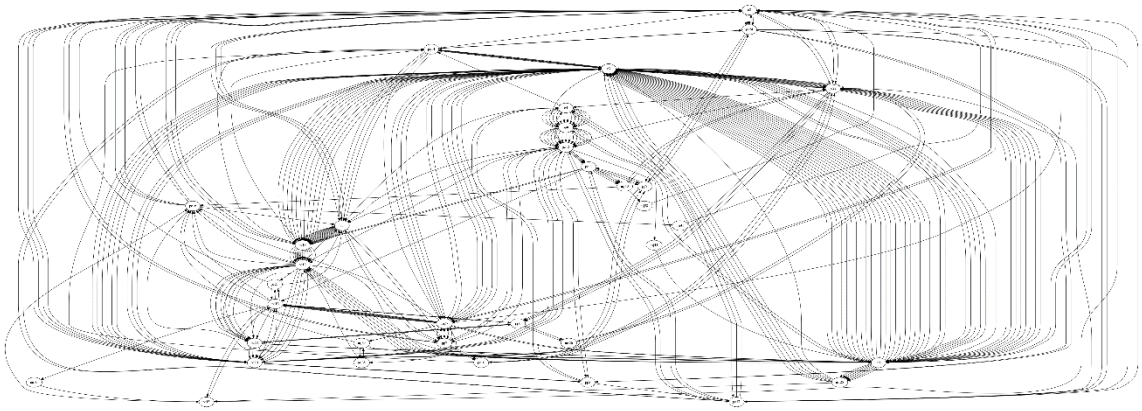


Figura 5.5 - Grafo Simulación 3 Caso Conductor SprayAndWait 5:20.

5.2.4. Simulación 4 Caso Conductor SprayAndWait 20:80

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINGPONG_SAW_20_80
sim_time: 43200.1000
pings sent: 1600
pings received: 1476
pongs sent: 1476
pongs received: 848
ping delivery prob: 0.9225
pong delivery prob: 0.5745
ping/pong success prob: 0.5300
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_CONDUCTOR_PINGPONG_SAW_20_80
sim_time: 43200.1000
overhead_ratio: 9.0806
latency_avg: 4572.9764
latency_med: 3042.9000
```

```
hopcount_avg: 2.6922
hopcount_med: 3
buffertime_avg: 17278.1216
buffertime_med: 17740.6000
rtt_avg: 6002.4696
rtt_med: 2855.3000
```

Las múltiples conexiones entre los nodos y la gran cantidad de mensajes en la Red generan un grafo poco apreciable cuya interpretación de la Red pierde sentido, por lo tanto se determinó no representarla.

5.2.5. Simulación 5 Caso Pasajero Epidemic 5:20

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_5_20
sim_time: 43200.1000
pings sent: 100
pings received: 100
pongs sent: 100
pongs received: 40
ping delivery prob: 1.0000
pong delivery prob: 0.4000
ping/pong success prob: 0.4000
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_5_20
sim_time: 43200.1000
overhead_ratio: 48.0000
```

```
latency_avg: 3664.0940
latency_med: 3319.0000
hopcount_avg: 3.8267
hopcount_med: 3
buffertime_avg: 14312.2841
buffertime_med: 14765.5000
rtt_avg: 9252.2000
rtt_med: 7297.2000
```

En la Figura 5.6 - Grafo Simulación 5 Caso Pasajero Epidemic 5:20 observaremos el grafo de las conexiones de los nodos y las rutas que los mensajes han recorrido en la Red. La Figura 5.6 se podrá observar con mayor detalle en el Anexo D – Grafos de las simulaciones.

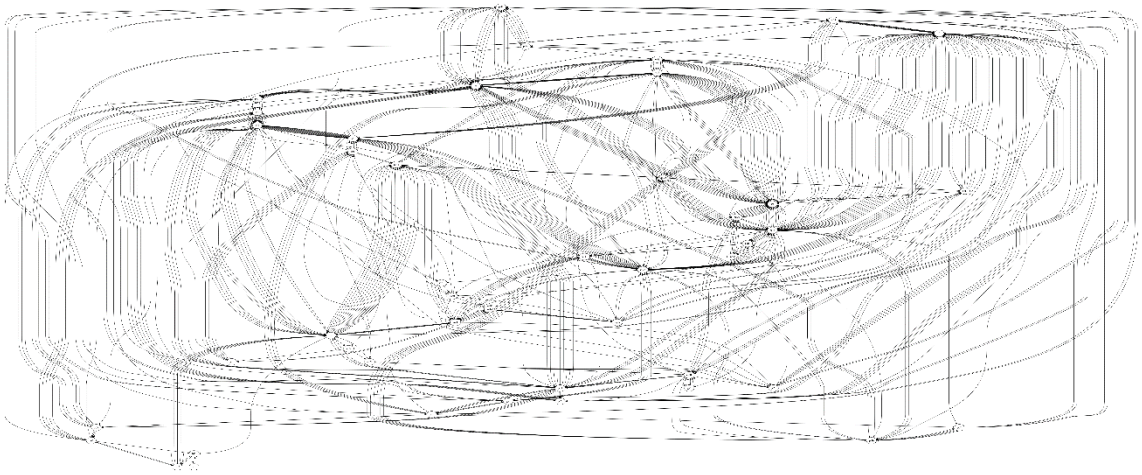


Figura 5.6 - Grafo Simulación 5 Caso Pasajero Epidemic 5:20.

5.2.6. Simulación 6 Caso Pasajero Epidemic 20:80

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_20_80
sim_time: 43200.1000
pings sent: 1600
pings received: 1600
pongs sent: 1600
pongs received: 386
ping delivery prob: 1.0000
pong delivery prob: 0.2413
ping/pong success prob: 0.2413
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_20_80
sim_time: 43200.1000
overhead_ratio: 122.9403
latency_avg: 2281.5175
latency_med: 1902.5000
hopcount_avg: 5.8006
hopcount_med: 5
buffertime_avg: 15460.9512
buffertime_med: 16001.5000
rtt_avg: 8726.0750
rtt_med: 9144.2000
```

Las múltiples conexiones entre los nodos y la gran cantidad de mensajes en la Red generan un grafo poco apreciable cuya interpretación de la Red pierde sentido, por lo tanto se determinó no representarla.

5.2.7. Simulación 7 Caso Pasajero SprayAndWait 5:20

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_5_20
sim_time: 43200.1000
pings sent: 100
pings received: 93
pongs sent: 93
pongs received: 39
ping delivery prob: 0.9300
pong delivery prob: 0.4194
ping/pong success prob: 0.3900
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_5_20
sim_time: 43200.1000
overhead_ratio: 9.0500
latency_avg: 5192.1507
latency_med: 3598.6000
hopcount_avg: 2.6571
hopcount_med: 3
buffertime_avg: 15851.8563
buffertime_med: 17019.5000
rtt_avg: 17598.9750
rtt_med: 16706.7000
```

En la Figura 5.7 - Simulación 7 Caso Pasajero SprayAndWait 5:20 observaremos el grafo de las conexiones de los nodos y las rutas que los mensajes han recorrido en la Red. La Figura 5.7 se podrá observar con mayor detalle en el Anexo D – Grafos de las simulaciones.

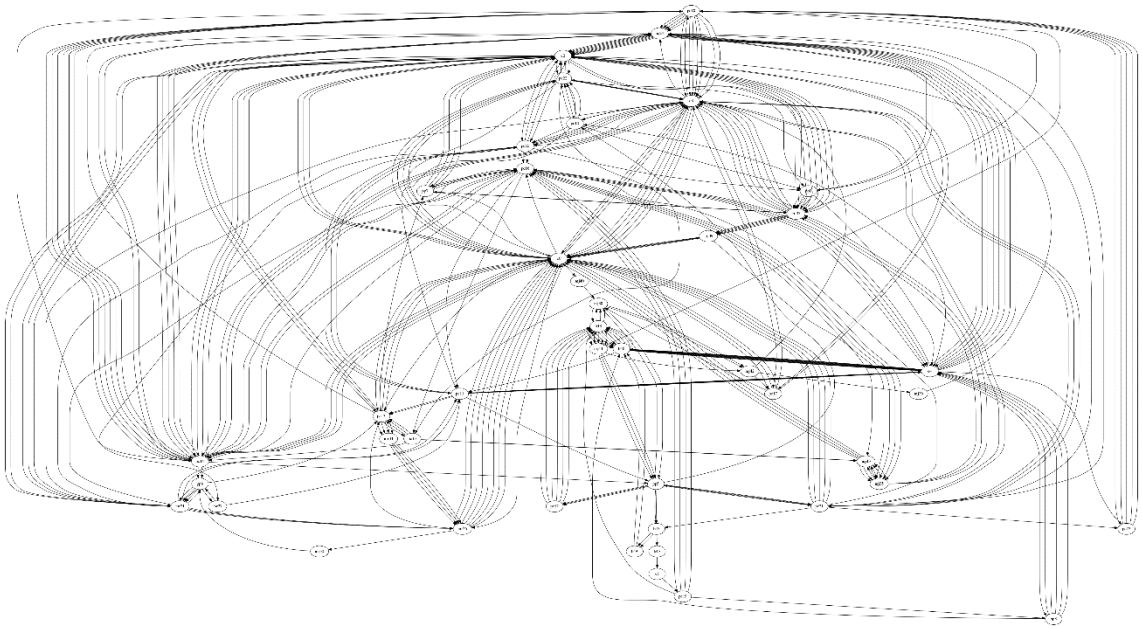


Figura 5.7 - Simulación 7 Caso Pasajero SprayAndWait 5:20.

5.2.8. Simulación 8 Caso Pasajero SprayAndWait 20:80

Los resultados de la simulación generan las siguientes estadísticas de ping:

```
Ping stats for scenario
MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_20_80
sim_time: 43200.1000
pings sent: 1600
pings received: 1539
pongs sent: 1539
pongs received: 449
ping delivery prob: 0.9619
pong delivery prob: 0.2917
ping/pong success prob: 0.2806
```

Estadísticas de rendimiento:

```
Message stats for scenario
MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_20_80
sim_time: 43200.1000
overhead_ratio: 9.0264
latency_avg: 3987.9358
latency_med: 2792.2000
hopcount_avg: 2.6796
hopcount_med: 3
buffertime_avg: 17230.8625
buffertime_med: 17703.7000
rtt_avg: 15594.8636
rtt_med: 12976.6000
```

Las múltiples conexiones entre los nodos y la gran cantidad de mensajes en la Red generan un grafo poco apreciable cuya interpretación de la Red pierde sentido, por lo tanto se determinó no representarla.

5.3. Implementación del MPEM

En base a lo planteado en el Subcapítulo 3.4 - Herramientas para la implementación criptografía homomórfica, nuestro MPEM se desarrollará utilizando el lenguaje de programación C, las librerías del proyecto Habcast para el cifrado homomórfico y OpenSSL. Además el Sistema Operativo base escogido para realizar el desarrollo del MPEM es un Linux CentOS 7 x86_64.

La implementación del MPEM consta del desarrollo de dos aplicaciones; la primera aplicación pertenece a los conductores y la segunda a los pasajeros, estas aplicaciones funcionarán en base a los casos de uso definidos en el Capítulo 4.4. Diseño del MPEM basado en criptografía homomórfica en una Red oportunista. A continuación el desarrollo de las aplicaciones del MPEM. El código fuente de las aplicaciones del conductor y pasajero se encuentra en el Anexo A – Código fuente del MPEM.

5.3.1. Desarrollo de la aplicación conductor para el MPEM

El desarrollo de la aplicación del conductor la dividiremos en módulos para su mayor comprensión, estos módulos siguen los procesos definidos en los diagramas de flujo de la aplicación conductor caso conductor y aplicación conductor caso pasajero expresados en las Figuras 4.4 y 4.5. A continuación los módulos desarrollados:

Módulo de invocación de librerías. En este módulo se invocan las librerías a utilizar por el MPEM aplicación conductor, las librerías invocadas son las siguientes:

```
#include <stdio.h> //librería de entradas y salidas
#include <stdlib.h> //librería de funciones generales
#include <string.h> //librería de manejo de caracteres
#include <time.h> //librería de funciones de tiempo.

//Librerias OpenSSL
```

```
#include <openssl/bn.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/rand.h>
#include "paillier.h" //Liberia del criptosistema homomórfico de
Paillier

//Librería de comunicación
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>
```

Módulo de definición de estructuras de datos y variables globales. En este módulo se desarrollan las estructuras de datos para la aplicación conductor y pasajero, además definimos variables globales:

```
#define KEY_LEN 512

BN_CTX *ctx; //Random que servirá para inicializar variables Bignum

typedef struct oferta_s {
    char n[310];
    char n2[310];
    char g[310];
    char precio_s[310];
    char conductor[11];
    char fecha[9];
    char ciudad_ori[4];
    char ciudad_dest[4];
    int plazas;
    BIGNUM *precio;
} oferta;

typedef struct demanda_s {
    char pasajero[11];
    char contacto[11];
    char limite_precio[310];
    BIGNUM *limite;
} demanda;


paillierKeys K = {0};
```

Módulo de elección de caso (conductor o pasajero). Este módulo permite escoger al conductor el caso de uso que desea activar.

```
     int option = 0;
    while ( option < 1 || option > 2 ) {
    printf("\nBienvenido al MPEM - App Conductor\n\n");
    printf("Ingrese 1 para ejecutar la aplicacion en Caso Conductor
o 2 en Caso Pasajero: \n");
    scanf("%d", &option);
    //printf("Numero ingresado: %d \n", option);
    }
```

Caso conductor, módulo de declaración de variables. En este módulo declaramos las variables que utilizaremos para la aplicación conductor caso conductor.

```
oferta oferta_c;
demanda demanda_c;

int socket_desc[10];
struct sockaddr_in server;
int read_size;

srand(time(NULL));
ctx = BN_CTX_new();

BN_CTX_start(ctx);
int key_len = KEY_LEN;

long long precio_plano = 0;

oferta_c.precio = BN_CTX_get(ctx);
demanda_c.limite = BN_CTX_get(ctx);

BIGNUM *n;
BIGNUM *n2;
BIGNUM *g;
n = NULL;
n2 = NULL;
g = NULL;
```

Caso conductor, módulo de lectura de archivo con lista de pasajeros. Este módulo permite leer un archivo en formato txt, el cual contiene la lista de pasajeros del MPEM.

```
int len = 2;
 int cont = 0;
 FILE *fp;
 char buff[16];
 char ip_clients[3][16];

 fp = fopen("pasajeros.txt", "r");

 printf("\nMPEM - App Conductor - Caso Conductor\n" );

 printf("Pasajeros registrados en el MPEM\n\n" );

 while( len > 1 ) {
     fgets(buff, 16, (FILE*)fp);
     printf("%s", buff );
     len = strlen(buff);
     strcpy(ip_clients[cont],buff);
     buff[0] = '\0';
     cont = cont+1;
 }

 fclose(fp);
```

Caso conductor, módulo de ingreso de información del conductor y precio del viaje. Este módulo permite el ingreso por pantalla de los datos del conductor y el precio a cobrar por el viaje.

```
printf("Informacion del Conductor \n\n");
printf("Ingrese su numero de identificacion: \n");
scanf("%s", &oferta_c.conductor);
printf("Ingrese ciudad origen: \n");
scanf("%s", &oferta_c.ciudad_ori);
printf("Ingrese ciudad destino: \n");
scanf("%s", &oferta_c.ciudad_dest);
printf("Ingrese fecha del viaje: \n");
scanf("%s", &oferta_c.fecha);
printf("Ingrese el precio del viaje: \n");
scanf("%lli", &precio_plano);
printf("Ingrese el numero de asientos disponibles: \n");
scanf("%d", &oferta_c.plazas);
```

Caso conductor, módulo de generación de llaves de Paillier y cifrado. En este módulo se generan las llaves pública y privada del criptosistema homomórfico de Paillier, además con la llave pública procedemos a cifrar el valor del precio ofertado por el conductor.

```
if (generateRandomKeys(&K, &key_len, ctx) != 0)
    return -1;

if (encryptll(oferta_c.precio, precio_plano, &K.pub, ctx) != 0)
    return 1;
```

Caso conductor, módulo de conversión de Bignum a Char. Es necesario que todos los datos enviados a través de una conexión TCP sean de tipo de dato Char, caso contrario no llegarán de manera íntegra al host destino. Por lo tanto este módulo se encarga de convertir los datos Bignum de la llave de Paillier y del precio cifrado del conductor a datos tipo Char.

```
char *result_str_p = BN_bn2dec(oferta_c.precio);  // convert the
 BIGNUM back to string

char *result_str_g = BN_bn2dec(K.pub.g);  // convert the BIGNUM back
 to string

char *result_str_n = BN_bn2dec(K.pub.n);  // convert the BIGNUM back
 to string

char *result_str_n2 = BN_bn2dec(K.pub.n2);  // convert the BIGNUM
 back to string
```

Caso conductor, módulo de creación de conexión y envío de data. Este módulo se encarga de establecer una conexión TCP para el envío de los datos del conductor y del viaje a todos los hosts pasajeros

registrado en la lista de pasajeros, en este proceso se envía la identificación del conductor, fecha del viaje, origen, destino del viaje, precio cifrado con Paillier Eh(precio) y número de plazas disponibles. El puerto utilizado por las aplicaciones del MPEM es el 8.8.8.8.

```
for(i = 0; i < cont; i++)
{
    //Create socket
    socket_desc[i] = socket(AF_INET , SOCK_STREAM , 0);
    if (socket_desc[i] == -1)
    {
        printf("Could not create socket");
    }
    puts("Socket created");

    //Prepare the sockaddr_in structure
    printf("%s", ip_clients[i]);
    server.sin_addr.s_addr = inet_addr(ip_clients[i]);
    server.sin_family = AF_INET;
    server.sin_port = htons( 8888 );

    //Connect to remote server
    if (connect(socket_desc[i] , (struct sockaddr *)&server ,
 sizeof(server)) < 0)
    {
        perror("connect failed. Error");
        return 1;
    }

    puts("Connected");

    //Send some messages to the client
    if( send(socket_desc[i] , (void *)&oferta_c , sizeof(oferta_c) ,
 0) < 0)
    {
        puts("Send failed");
        return 1;
    }
    puts("Sent offer\n");

}
```

Caso conductor, módulo de recepción y proceso de información del pasajero. Este módulo se encarga de recibir la información y límite

a pagar por el viaje de todos los pasajeros registrados en la lista de pasajeros.

El proceso de la información consiste en convertir de Char a Bignum al dato Eh(límite – precio) enviado por el pasajero para luego descifrarlo con la llave privada de Paillier; luego se consulta si el valor descifrado es menor a 0, si lo es, se muestra el mensaje “El precio de la demanda es menor al precio de la oferta, precio límite del pasajero no aceptado”, caso contrario se muestra el mensaje “Límite menos Precio positivo, Oferta aceptada” y se reduce una plaza en el automóvil, para luego evaluar cuantas plazas sobran y si no sobran plazas o asientos se muestra el mensaje "Las plazas se han llenado, no existe asientos disponibles, programa MPEM finalizado, Gracias...” Finalizando el programa, caso contrario se espera por la respuesta de los demás pasajeros hasta que todos hayan correspondido.

```
while( (read_size = recv(socket_desc[k] , (void *)&demanda_tmp ,
 sizeof(demanda_tmp), 0)) > 0 )
{
    //Proccess
    demanda demanda_p = demanda_tmp;
    BIGNUM *result_sub;
    result_sub = NULL;
    long long limite_precio;

    puts("Demanda del cliente \n");
    printf("Identificacion del pasajero:\n%s\n", demanda_p.pasajero);
    printf("Numero de telefono del pasajero:\n%s\n",
 demanda_p.contacto);
    printf("Limite menos precio cifrado con Paillier:\n%s\n",
 demanda_p.limite_precio);
```

```
   BN_dec2bn(&result_sub, demanda_p.limite_precio); // convert the
String back to Bignum

   if (decryptll(&limite_precio, result_sub, &K.priv, ctx) != 0)
       exit(1); // Descifrado de Eh(limite – precio)

   puts("\nProceso de la informacion cifrada \n");
   if (limite_precio < 0)
   printf("El precio demanda es menor al precio oferta, precio
limite del pasajero no aceptado:\n%lli\n", limite_precio);
       else
       {
           printf("Limite menos Precio positivo, Oferta aceptada,
Resta Paillier:\n%lli\n", limite_precio);
           oferta_c.plazas=oferta_c.plazas-1;
           if (oferta_c.plazas > 0)
               printf("Numero de asientos disponibles:\n%d\n",
oferta_c.plazas);
           else{
               printf("Las plazas se han llenado, no existe asientos
disponibles, programa MPEM finalizado, Gracias...\n\n");
               return (EXIT_SUCCESS);
           }
       }
   if (k < i)
       k = k+1;
   else
       k = 0;
}
```

Caso pasajero, módulo de declaración de variables. En este módulo declaramos las variables que utilizaremos para la aplicación conductor caso pasajero.

```
oferta oferta_p;
BIGNUM *n;
BIGNUM *n2;
BIGNUM *g;
BIGNUM *precio;
n = NULL;
n2 = NULL;
g = NULL;
precio = NULL;
long long limite_plano = 0;

BIGNUM *result_sum;
BIGNUM *result_sub;
result_sum = BN_CTX_get(ctx);
result_sub = BN_CTX_get(ctx);
struct _pubKey pub;

demanda demanda_c;
```

```
demanda_c.limite = BN_CTX_get(ctx);
```

Caso pasajero, módulo de ingreso de información del conductor. Este módulo permite el ingreso por pantalla de los datos del conductor como id y teléfono.

```
printf("MPEM - App Conductor - Caso Pasajero\n" );
printf("Informacion del Conductor\n\n");
printf("Ingrese su numero de identificacion: \n");
scanf("%s", &demanda_c.pasajero);
printf("Ingrese su numero de telefono: \n");
scanf("%s", &demanda_c.contacto);
```

Caso pasajero, módulo de conexión. Este módulo se encarga de crear el socket de conexión y escuchar a los pasajeros, el puerto de recepción para establecer las conexiones es 8888.

```
int sock, c, client_sock;
struct sockaddr_in server, client;

//Create socket
sock = socket(AF_INET , SOCK_STREAM , 0);
if (sock == -1)
{
    printf("Could not create socket");
}
puts("Socket created");

//Prepare the sockaddr_in structure
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( 8888 );

//Bind
if( bind(sock,(struct sockaddr *)&server , sizeof(server)) < 0)
{
    //print the error message
    perror("bind failed. Error");
    return 1;
}
puts("bind done");

//Listen
listen(sock , 3);
```

```
//Accept and incoming connection
puts("Waiting for incoming connections...");
c = sizeof(struct sockaddr_in);
```

Caso pasajero, módulo de recepción de data del pasajero. Este módulo se encarga de recibir la data del pasajero, la data recibida consiste en información del pasajero y el límite cifrado con Paillier Eh(límite)

```
//Receive a reply from the server
if( recv(client_sock , (void *)&oferta_p, sizeof(oferta_p) , 0) < 0)
{
    puts("recv failed");
    return 1;
}
puts("Offer received\n");


puts("Demanda del pasajero\n");
printf("Identificacion del pasajero:\n%s\n", oferta_p.conductor);
printf("Fecha del viaje:\n%s\n", oferta_p.fecha);
printf("Ciudad Origen:\n%s\n", oferta_p.ciudad_ori);
printf("Ciudad Destino:\n%s\n", oferta_p.ciudad_dest);
printf("Numero de acientos solicitados:\n%d\n", oferta_p.plazas);
printf("Precio limite cifrado con Paillier:\n%s\n",
 oferta_p.precio_s);
```

Caso pasajero, módulo de proceso de data del pasajero. Este módulo se encarga de consultar al conductor si desea aceptar el viaje, si no lo acepta el programa finaliza. Caso contrario convierte los datos llave de Paillier y límite a pagar (enviados por el pasajero) de Char a Bignum, Para luego consultar al conductor por el precio del viaje que cree conveniente cobrar.

Luego de recibir el precio del viaje por entrada de teclado por parte del conductor, se procede con el cifrado del precio del viaje. Con Eh(precio) se realiza la operación homomórfica de la resta del Eh(límite) – Eh(precio) con el criptosistema de Paillier. Finalmente se envía el dato Eh(límite – precio) al pasajero.

```
int option2 = 0;

while ( option2 < 1 || option2 > 2 ) {
printf("\nDesea aceptar la demanda del viaje\n\n");
printf("Ingrese 1 para aceptar o 2 para rechazar: \n");
scanf("%d", &option2);
}

if ( option2 == 2 )
        printf("No se acepto la demanda de viaje del pasajero. Fin
 MPEM\n\n"); return (EXIT_SUCCESS);

BN_dec2bn(&precio, oferta_p.precio_s);
BN_dec2bn(&n, oferta_p.n);
BN_dec2bn(&n2, oferta_p.n2);
BN_dec2bn(&g, oferta_p.g);

pub.g = BN_dup(g);
pub.n = BN_dup(n);
pub.n2 = BN_dup(n2);

printf("Oferta del conductor\n\n");
printf("Ingrese precio a cobrar por el viaje: \n");
scanf("%lli", &limite_plano);

if (encryptll(demanda_c.limite, limite_plano, &pub, ctx) != 0)
    return 1;

char *result_str_lim = BN_bn2dec(demanda_c.limite);  // convert the
 BIGNUM back to string
printf("Precio del viaje cifrado con Paillier:\n%s\n",
 result_str_lim);

if (sub(result_sub, precio, demanda_c.limite, pub.n2, ctx) != 0)
    return 1;

char *result_str_sub = BN_bn2dec(result_sub);  // convert the BIGNUM
 back to string
printf("Resta, precio del viaje del conductor menos limite a pagar
 por el pasajero, operacion computada con el criptosistema
 homomorfico de Paillier:\n%s\n", result_str_sub);
strcpy(demanda_c.limite_precio,result_str_sub);

//Send some data
if( send(client_sock , (void *)&demanda_c , sizeof(demanda_c) , 0) <
 0)
```

```
{
    puts("Send failed");
    return 1;
}

printf("\nPrecio del viaje menos limite a pagar por el pasajero fue
 enviado, si el precio del viaje es menor al limite, el pasajero se
 comunicara con usted. Fin MPEM\n\n");
```

5.3.2. Desarrollo de la aplicación pasajero para el MPEM

El desarrollo de la aplicación del pasajero la dividiremos en módulos para su mayor comprensión, estos módulos siguen los procesos definidos en los diagramas de flujo de la aplicación pasajero caso conductor y aplicación pasajero caso pasajero representados en las Figuras 4.6 y 4.7. A continuación los módulos desarrollados:

La aplicación pasajero utiliza los módulos: Módulo de invocación de librerías y Módulo de definición de estructuras de datos y variables globales definidos previamente en el Subcapítulo 5.3.1. Desarrollo de la aplicación conductor para el MPEM.

Módulo de elección de caso (conductor o pasajero). Este módulo permite escoger al pasajero el caso de uso que desea activar.

```
int option = 0;

while ( option < 1 || option > 2 ) {
printf("\nBienvenido al MPEM - App Pasajero\n\n");
printf("Ingrese 1 para ejecutar la aplicacion en Caso Conductor o
     2 en Caso Pasajero: \n");
```

```
scanf("%d", &option);
}
```

Caso conductor, módulo de ingreso de información del pasajero. Este módulo permite el ingreso por pantalla de los datos del pasajero como id y teléfono.

```
printf("MPEM - Caso Conductor - App Pasajero\n");
printf("Informacion del Pasajero\n\n");
printf("Ingrese su numero de identificacion: \n");
scanf("%s", &demanda_p.pasajero);
printf("Ingrese su numero de telefono: \n");
scanf("%s", &demanda_p.contacto);
```

Caso conductor, módulo de conexión. Este módulo se encarga de crear el socket de conexión y escuchar a los conductores, el puerto de recepción para establecer las conexiones es 8888.

```
int sock, c, client_sock;
struct sockaddr_in server, client;

//Create socket
sock = socket(AF_INET , SOCK_STREAM , 0);
if (sock == -1)
{
    printf("Could not create socket");
}
puts("Socket created");

//Prepare the sockaddr_in structure
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( 8888 );

//Bind
if( bind(sock,(struct sockaddr *)&server , sizeof(server)) < 0)
{
    //print the error message
    perror("bind failed. Error");
    return 1;
}
puts("bind done");

//Listen
listen(sock , 3);
```

```
//Accept and incoming connection
puts("Waiting for incoming connections...");
```

Caso conductor, módulo de recepción de data del conductor. Este módulo se encarga de recibir la data del conductor, la data recibida consiste en información del conductor y el precio del viaje cifrado con Paillier Eh(precio).

```
//Receive a reply from the server
if( recv(client_sock , (void *)&oferta_c, sizeof(oferta_c) , 0) < 0)
{
    puts("recv failed");
    return 1;
}
puts("Offer received\n");


puts("Oferta del conductor\n");
printf("Identificacion del conductor:\n%s\n", oferta_c.conductor);
printf("Fecha del viaje:\n%s\n", oferta_c.fecha);
printf("Ciudad Origen:\n%s\n", oferta_c.ciudad_ori);
printf("Ciudad Destino:\n%s\n", oferta_c.ciudad_dest);
printf("Numero de asientos disponibles:\n%d\n", oferta_c.plazas);
printf("Precio cifrado con Paillier:\n%s\n", oferta_c.precio_s);
```

Caso conductor, módulo de proceso de data del conductor. Este módulo se encarga de consultar al pasajero si desea aceptar el viaje, si no lo acepta el programa finaliza. Caso contrario convierte los datos llave de Paillier y precio del viaje (enviados por el conductor) de Char a Bignum, para luego consultar al pasajero por el precio límite a pagar por el viaje.

Luego de recibir el precio límite por entrada de teclado por parte del pasajero, se procede con el cifrado del precio límite a pagar por el viaje. Con Eh(límite) se realiza la operación homomórfica de la resta del Eh(límite) – Eh(precio) con el criptosistema de Paillier. Finalmente se envía el dato Eh(límite – precio) al conductor.

```
int option2 = 0;

while ( option2 < 1 || option2 > 2 ) {
printf("\nDesea aceptar la oferta de viaje\n\n");
printf("Ingrese 1 para aceptar o 2 para rechazar: \n");
scanf("%d", &option2);
}

if ( option2 == 2 )
        printf("No se acepto la oferta de viaje del conductor. Fin
 MPEM\n\n"); return (EXIT_SUCCESS);
BN_dec2bn(&precio, oferta_c.precio_s);
BN_dec2bn(&n, oferta_c.n);
BN_dec2bn(&n2, oferta_c.n2);
BN_dec2bn(&g, oferta_c.g);

pub.g = BN_dup(g);
pub.n = BN_dup(n);
pub.n2 = BN_dup(n2);

printf("Demanda del cliente\n\n");
printf("Ingrese precio limite a pagar: \n");
scanf("%lli", &limite_plano);

if (encryptll(demanda_p.limite, limite_plano, &pub, ctx) != 0)
    return 1;

char *result_str_lim = BN_bn2dec(demanda_p.limite);  // convert the
 BIGNUM back to string
printf("Limite cifrado con Paillier:\n%s\n", result_str_lim);

if (sub(result_sub, demanda_p.limite, precio, pub.n2, ctx) != 0)
    return 1;

char *result_str_sub = BN_bn2dec(result_sub);  // convert the BIGNUM
 back to string
printf("Resta, precio limite del pasajero menos precio del viaje del
 conductor, operacion computada con el criptosistema homomorfico de
 Paillier:\n%s\n", result_str_sub);
strcpy(demanda_p.limite_precio,result_str_sub);

//Send some data
if( send(client_sock , (void *)&demanda_p , sizeof(demanda_p) , 0) <
 0)
{
    puts("Send failed");
```

```
    return 1;
}

printf("\nLimite a pagar por el pasajero menos precio del viaje del
 conductor enviado al conductor, si el limite se ajusta a la oferta,
 el conductor se comunicara con usted. Fin MPEM\n\n");
```

Caso pasajero, módulo de declaración de variables. En este módulo declaramos las variables que utilizaremos para la aplicación pasajero caso pasajero.

```
long long precio_plano = 0;
int socket_desc[10];
struct sockaddr_in server;
int read_size;

int key_len = KEY_LEN;
paillierKeys K = {0};

oferta_p.precio = BN_CTX_get(ctx);
BIGNUM *n;
BIGNUM *n2;
BIGNUM *g;
n = NULL;
n2 = NULL;
g = NULL;

int len = 2;
int cont = 0;
int i = 0;
FILE *fp;
char buff[16];
char ip_clients[3][16];
```

Caso pasajero, módulo de lectura de archivo con lista de conductores. Este módulo permite leer un archivo en formato txt, el cual contiene la lista de conductores del MPEM.

```
fp = fopen("conductores.txt", "r");

printf("MPEM - App Pasajero - Caso Pasajero\n" );

printf("Conductores registrados en el MPEM\n\n" );

while( len > 1 ) {
```

```
    fgets(buff, 16, (FILE*)fp);
    printf("%s", buff );
    len = strlen(buff);
    strcpy(ip_clients[cont],buff);
    buff[0] = '\0';
    cont = cont+1;
}
```

Caso pasajero, módulo de ingreso de información del pasajero y precio límite a pagar por el viaje. Este módulo permite el ingreso por teclado de los datos del pasajero y el precio límite a pagar por el viaje.

```
printf("Informacion del Pasajero\n\n");
printf("Ingrese su numero de identificacion: \n");
scanf("%s", &oferta_p.conductor);
printf("Ingrese ciudad origen: \n");
scanf("%s", &oferta_p.ciudad_ori);
printf("Ingrese ciudad destino: \n");
scanf("%s", &oferta_p.ciudad_dest);
printf("Ingrese fecha del viaje: \n");
scanf("%s", &oferta_p.fecha);
printf("Ingrese el limite de precio a pagar por del viaje: \n");
scanf("%lli", &precio_plano);
printf("Ingrese el numero de asientos que solicita: \n");
scanf("%d", &oferta_p.plazas);
```

Caso pasajero, módulo de generación de llaves de Paillier y cifrado. En este módulo se generan las llaves pública y privada del criptosistema homomórfico de Paillier, además con la llave pública procedemos a cifrar el valor del precio límite a pagar demandado por el pasajero.

```
if (generateRandomKeys(&K, &key_len, ctx) != 0)
return -1;

if (encryptll(oferta_p.precio, precio_plano, &K.pub, ctx) != 0)
    return 1;
```

Caso pasajero, módulo de conversión de Bignum a Char. Es necesario que todos los datos enviados a través de una conexión TCP sean de tipo de dato Char, caso contrario no llegarán de manera íntegra al host destino. Por lo tanto este módulo se encarga de convertir los datos Bignum de la llave de Paillier y del precio límite cifrado del pasajero a datos tipo Char.

```
char *result_str_p = BN_bn2dec(oferta_p.precio);  // convert the
BIGNUM back to string
printf("Precio limite cifrado con Paillier:\n%s\n", result_str_p);
strcpy(oferta_p.precio_s,result_str_p);

char *result_str_g = BN_bn2dec(K.pub.g);  // convert the BIGNUM back
to string
//printf("Clave publica g:\n%s\n", result_str_g);
strcpy(oferta_p.g,result_str_g);


char *result_str_n = BN_bn2dec(K.pub.n);  // convert the BIGNUM back
to string
//printf("Clave publica n:\n%s\n", result_str_n);
strcpy(oferta_p.n,result_str_n);


char *result_str_n2 = BN_bn2dec(K.pub.n2);  // convert the BIGNUM
back to string
//printf("Clave publica n2:\n%s\n", result_str_n2);
strcpy(oferta_p.n2,result_str_n2);
```

Caso pasajero, módulo de creación de conexión y envío de data. Este módulo se encarga de establecer una conexión TCP para enviar los datos del pasajero y del viaje a todos los hosts conductores registrados en la lista de conductores, en este proceso se envía la identificación del pasajero, fecha del viaje, origen, destino del viaje, precio límite cifrado con Paillier Eh(límite) y número de plazas

solicitadas. El puerto utilizado por las aplicaciones del MPEM es el 8.8.8.8.

```
for(i = 0; i < cont; i++)
{
    //printf("cont: %d, i: %d\n",cont,i);
    //Create socket
    socket_desc[i] = socket(AF_INET , SOCK_STREAM , 0);
    if (socket_desc[i] == -1)
    {
        printf("Could not create socket");
    }
    puts("Socket created");

    //Prepare the sockaddr_in structure
    printf("%s", ip_clients[i]);
    server.sin_addr.s_addr = inet_addr(ip_clients[i]);
    server.sin_family = AF_INET;
    server.sin_port = htons( 8888 );

    //Connect to remote server
    if (connect(socket_desc[i] , (struct sockaddr *)&server ,
sizeof(server)) < 0)
    {
        perror("connect failed. Error");
        return 1;
    }

    puts("Connected");

    //Send some messages to the client
    if( send(socket_desc[i] , (void *)&oferta_p , sizeof(oferta_p) ,
0) < 0)
    {
        puts("Send failed");
        return 1;
    }
    puts("Sent offer");
}
```

Caso pasajero, módulo de recepción y proceso de información del conductor. Este módulo se encarga de recibir la información y precio del viaje de todos los conductores registrados en la lista de conductores.

El proceso de la información consiste en convertir de Char a Bignum al dato Eh(límite – precio) enviado por el conductor para luego descifrarlo con la llave privada de Paillier, luego se consulta si el valor descifrado es menor a 0, si lo es, se muestra el mensaje “El precio del viaje es mayor al valor límite, operación computada con el criptosistema homomórfico de Paillier, oferta del conductor no aceptado”, caso contrario se muestra el mensaje “El precio del viaje es menor/igual al valor límite, operación computada con el criptosistema homomórfico de Paillier, oferta del conductor aceptada” y se almacena el valor de (límite – precio) e identificación del conductor, para luego evaluar el precio más bajo por el viaje e identificar al conductor que ofrece dicho precio, aparecerá el siguiente mensaje " El conductor que ofrece el valor más bajo por el viaje es: xxx, su número telefónico es: xx, diferencia de valor:. Es xxx” Finalizando el programa. Por otro lado si ninguna oferta de los conductores fue aceptada se mostrara el mensaje “Todas las ofertas de los conductores superaron el precio límite a pagar, ofertas de conductores no aceptadas” y el MPEM finalizará.

```
while( (read_size = recv(socket_desc[k] , (void *)&demanda_tmp ,
 sizeof(demanda_tmp), 0)) > 0 )
{
    //Proccess
    demanda demanda_c = demanda_tmp;
    BIGNUM *result_sub;
    result_sub = NULL;
    long long limite_precio;

    puts("Oferta del conductor \n");
```

```
   printf("Identificacion del conductor:\n%s\n",
 demanda_c.pasajero);
   printf("Numero de telefono del conductor:\n%s\n",
 demanda_c.contacto);
   printf("Limite menos precio cifrado con Paillier:\n%s\n",
 demanda_c.limite_precio);

   BN_dec2bn(&result_sub, demanda_c.limite_precio);
   if (decryptll(&limite_precio, result_sub, &K.priv, ctx) != 0)
       exit(1);

   puts("\nProceso de la informacion cifrada \n");
   if (limite_precio < 0)
       printf("El precio del viaje es mayor al valor limite,
 operacion computada con el criptosistema homomorfico de Paillier,
 oferta del conductor no aceptado\n");
   else{
       printf("El precio del viaje es menor/igual al valor limite,
 operacion computada con el criptosistema homomorfico de Paillier,
 oferta del conductor aceptada\n");
       ofertas[cont2]=limite_precio;
       strcpy(conductores[cont2],demanda_c.pasajero);
       strcpy(contacto_condt[cont2],demanda_c.contacto);
       cont2 = cont2 + 1;
   }
   if (k < i)
       k = k+1;
   else
       k = 0;

if(read_size == 0)
{
    puts("Client disconnected");
    fflush(stdout);
}
else if(read_size == -1)
{
    long long max;
    int c, location = 1;
    max = ofertas[0];

    for ( c = 1 ; c < cont2 ; c++ )
    {
        if ( ofertas[c] > max )
        {
           max = ofertas[c];
           location = c+1;
        }
    }

    for ( c = 0 ; c < cont2 ; c++ )
    {
        if ( ofertas[c] == max )
        {
            id_conductor = c;
        }
    }

    if (cont2 != 0)
```

```
      printf("El conductor que ofrece el valor mas bajo por el
viaje es: %s, su numero telefonico es: %s, diferencia de valor:
%lli\n\n", conductores[id_conductor], contacto_condt[id_conductor],
ofertas[id_conductor]);
   else
            printf("Todas las ofertas de los conductores superaron
       el precio limite a pagar, ofertas de conductores no
       aceptadas\n\n");
     }
```

5.4. Pruebas de la implementación

Las pruebas de la implementación del MPEM consisten en dos escenarios, la primera prueba se basa en el caso de uso del conductor, donde un conductor que está dispuesto a viajar envía información del viaje que va a realizar incluyendo fecha, ciudad de origen, ciudad de destino, identificación del conductor, un precio a todos los pasajeros registrados en el MPEM y una llave publica generada por el criptosistema de Paillier. Los pasajeros responderán a la oferta del conductor con su información de contacto y la resta homomórfica de Paillier Eh(límite – precio), finalmente esta información llega al conductor, para que su aplicación escoja a los clientes que cumplen con la oferta de precio por el viaje y reserve los asientos demandados.

La segunda prueba corresponde al caso de uso del pasajero, donde un pasajero necesita viajar en una fecha estimada, de una ciudad a otra, y está dispuesto a pagar hasta un valor límite por el viaje, entonces el pasajero deberá hacer uso de su aplicación en caso pasajero para enviar

su información, una llave publica generada con el criptosistema de Paillier y la información del viaje antes mencionado. Esta demanda de viaje se enviará a todos los conductores registrados en el MPEM, los conductores responderán a la demanda con su información de contacto y la resta homomórfica de Paillier Eh(límite – precio). Una vez que el mensaje alcanza al pasajero, este verifica que el precio del viaje ofertado por el conductor no alcance el límite a pagar, de darse el caso la aplicación del pasajero guardará la identificación de los conductores que no superan los límites de precio para luego determinar quién tiene el precio más bajo por el viaje, finalizando el MPEM. A continuación mostramos figuras donde se puede ver la secuencia de ambos casos en ambas aplicaciones.

5.4.1. Ejecución del MPEM en caso conductor

Para dar inicio al MPEM en el caso conductor, todos los pasajeros que estén interesados en recibir ofertas de viajes deberán ejecutar su aplicación pasajero en caso conductor, como se puede observar en la Figura 5.8 – MPEM ejecución de aplicación pasajero en caso conductor. Donde dos pasajeros esperan por ofertas.

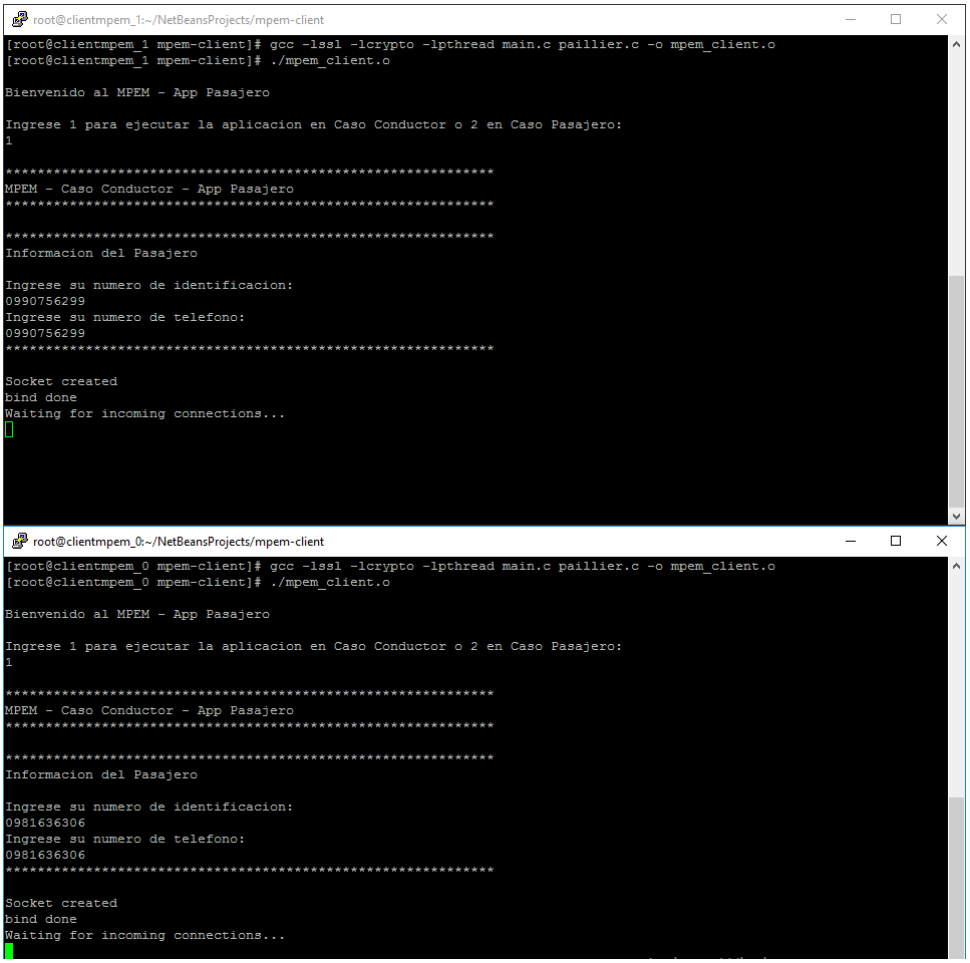


Figura 5.8 - MPEM ejecución de aplicación pasajero en caso conductor.

Cuando un conductor esté interesado en compartir su automóvil para viajar, ejecutará la aplicación conductor del MPEM en el caso conductor, como se muestra en la Figura 5.9 - MPEM ejecución de la aplicación conductor en caso conductor. Ahí los conductores ingresarán información personal como su identificación e información del viaje como ciudad origen, destino, fecha del viaje, número de

asientos disponibles y el precio del viaje cifrado con la llave publica de Paillier. Luego de ingresar los datos, se enviará un mensaje con la información antes mencionada a todos los pasajeros registrados en el MPEM.

La información del viaje y del conductor alcanzarán a los pasajeros que estén en línea como se puede observar en la Figura 5.10 – MPEM caso conductor, pasajeros reciben oferta de viaje. El pasajero tiene la opción de descartar la oferta o aceptarla y enviar un precio límite por el viaje conjunto a su información de contacto, este proceso se lo puede observar en la Figura 5.11 - MPEM caso conductor, pasajeros responden a la oferta de viaje.

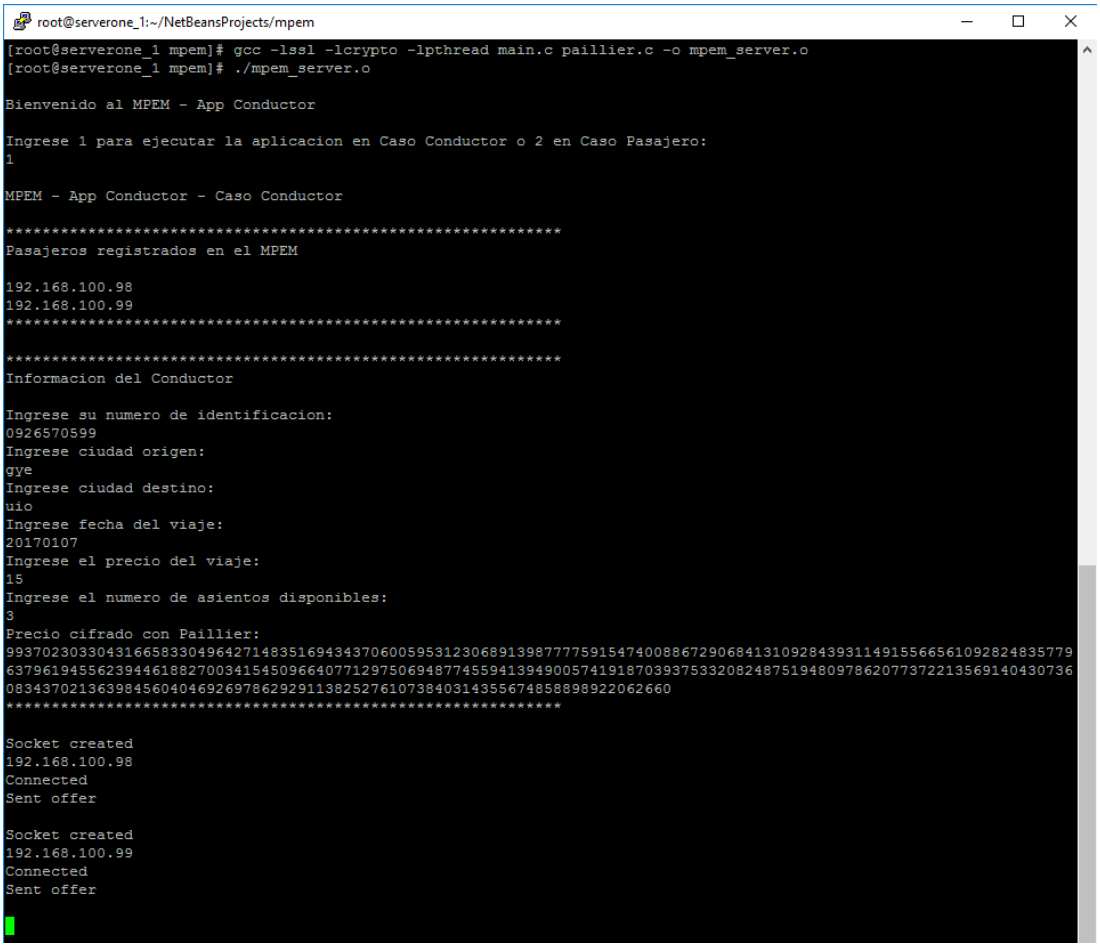


Figura 5.9 - MPEM ejecución de la aplicación conductor en caso conductor.

Finalmente las respuestas de los pasajeros llegarán al conductor, y la aplicación conductor evaluará que límite a pagar se ajusta al precio del viaje, se descartará a los valores que estén por debajo del precio del viaje y se aceptarán a los valores iguales o mayores al precio del viaje, reduciendo una plaza o asiento en el automóvil del conductor, este resultado se muestra en la Figura 5.12 – MPEM aplicación conductor,

evaluación de precios límites de los pasajeros. Finalmente el conductor podrá llamar al contacto del pasajero que salió favorecido.

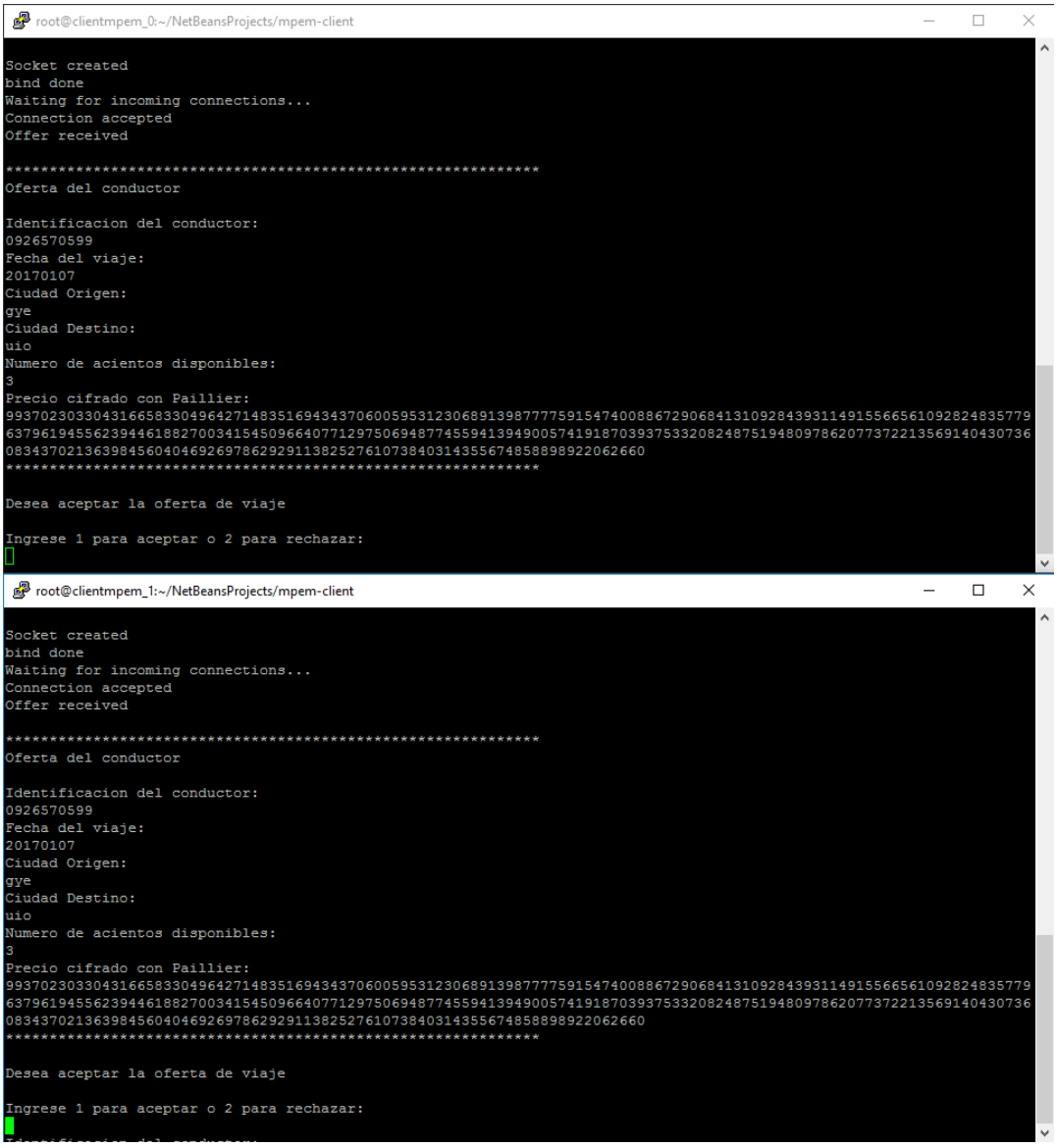


Figura 5.10 - MPEM caso conductor, pasajeros reciben oferta de viaje.

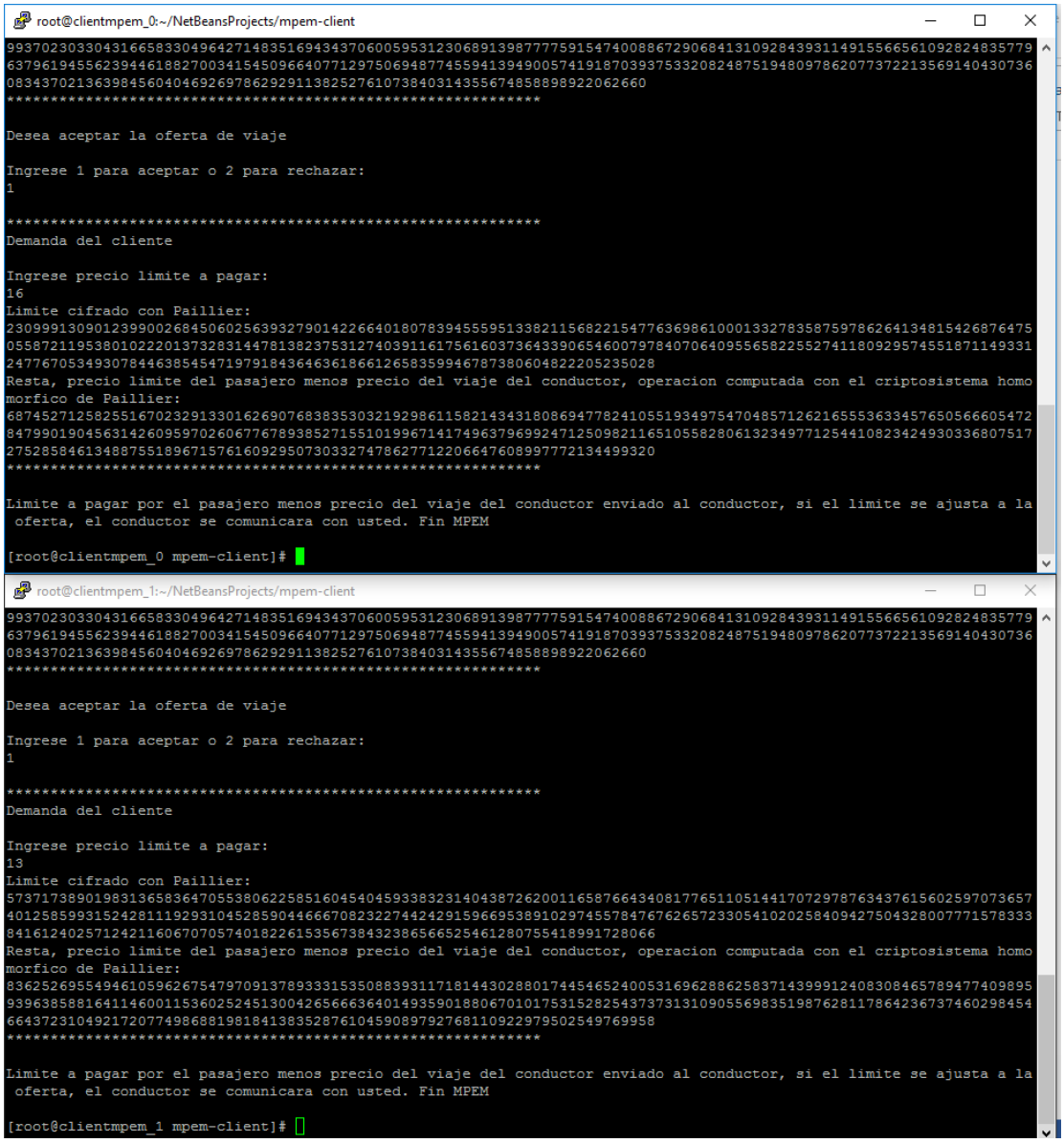


Figura 5.11 - MPEM caso conductor, pasajeros responden a la oferta de viaje.

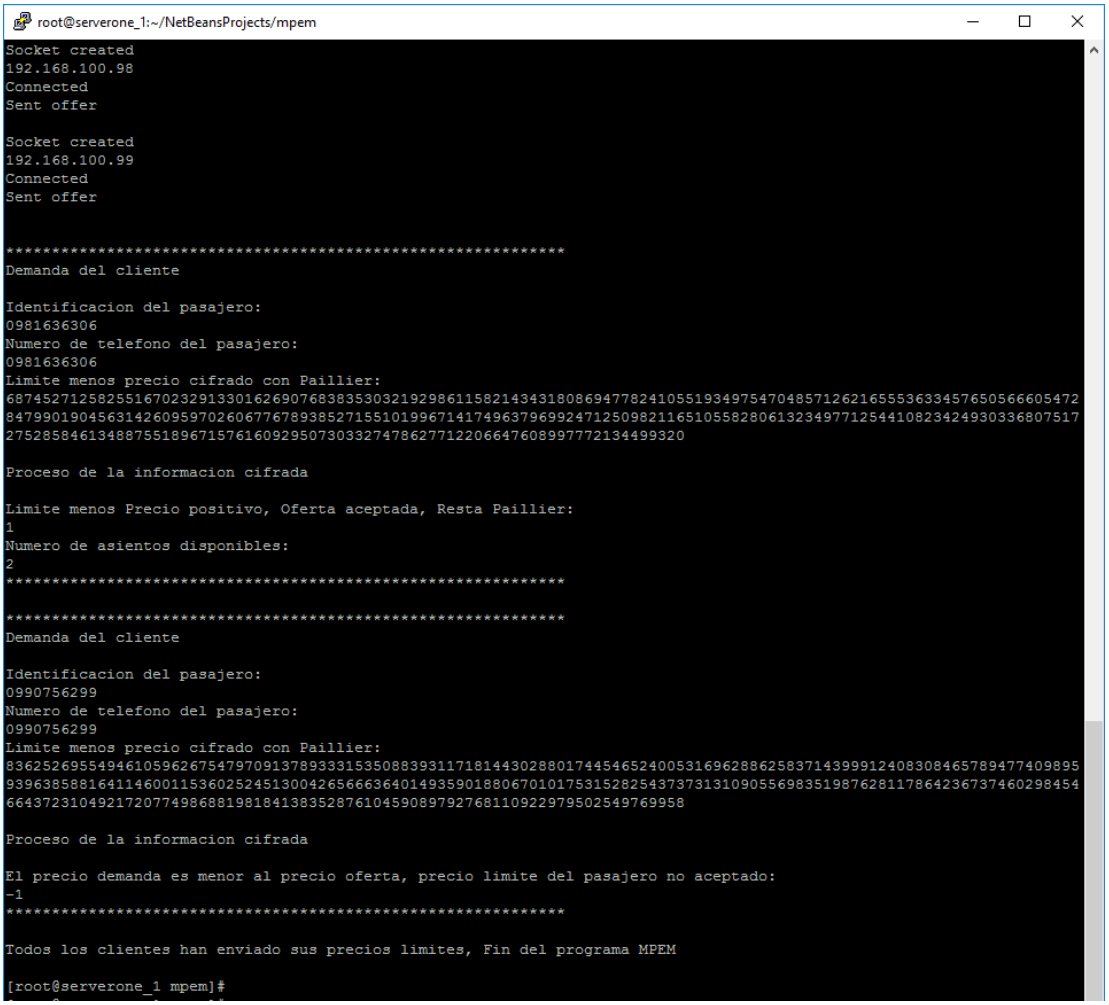


Figura 5.12 - MPEM aplicación conductor, evaluación de precios límites de los pasajeros.

5.4.2. Ejecución del MPEM en caso pasajero

Para dar inicio al MPEM en el caso pasajero, todos los conductores que estén interesados en recibir demandas de viajes deberán ejecutar su aplicación conductor en caso pasajero, como se puede observar en

la Figura 5.13 – MPEM ejecución de aplicación conductor en caso pasajero. Donde dos conductores esperan por demandas.

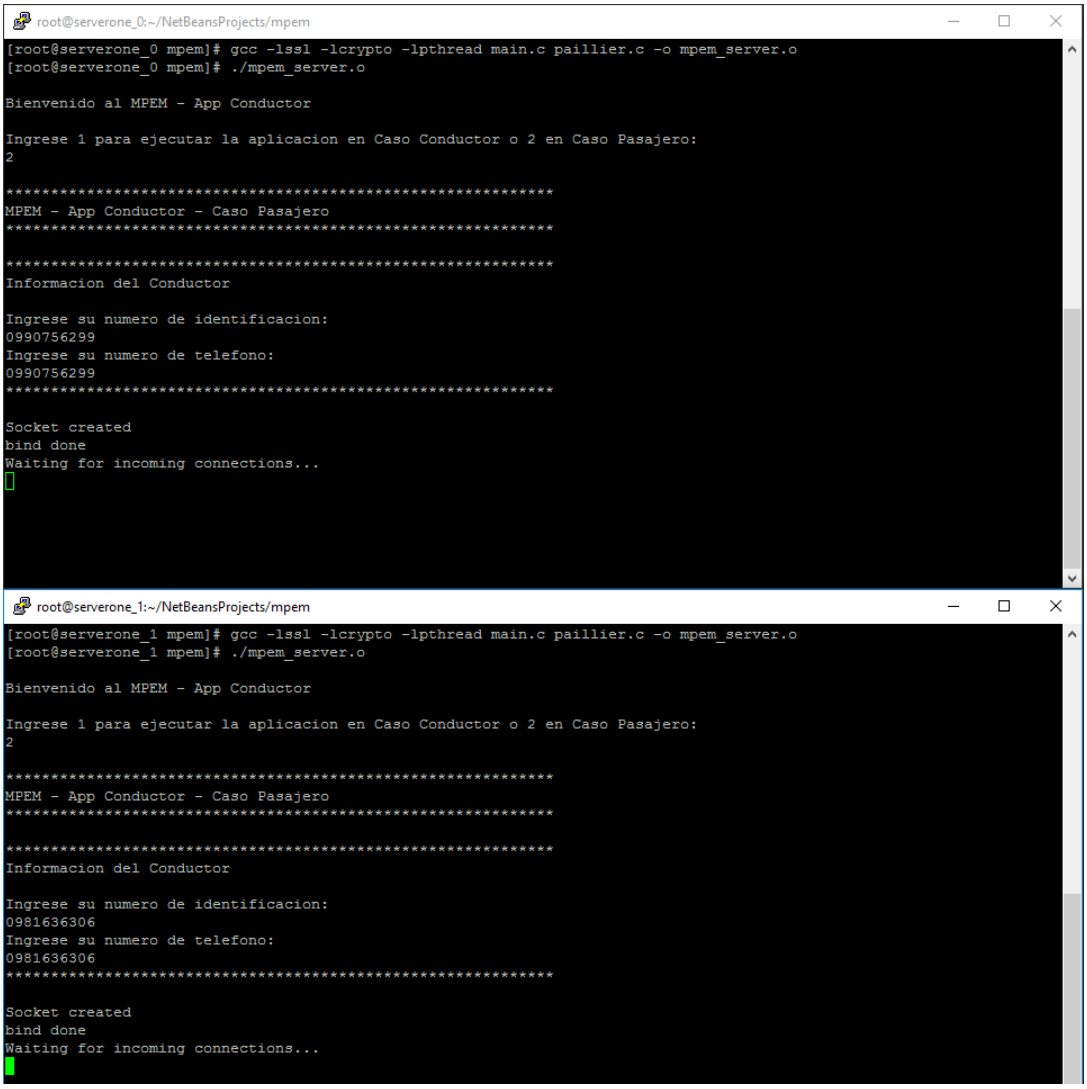


Figura 5.13 - MPEM ejecución de aplicación conductor en caso pasajero.

Cuando un pasajero desee viajar, deberá hacer uso de su aplicación pasajero en caso pasajero, donde podrá ingresar su información como identificación, además información del viaje que desea realizar, ciudad origen, destino, fecha del viaje, asientos que desea reservar, y el límite a pagar por el viaje cifrado con la llave publica de Paillier. Estos datos se enviarán a todos los pasajeros que estén registrados en el MPEM, la aplicación del pasajero estará a la espera de la respuesta de los conductores, para visualizar este proceso podemos observar la Figura 5.14 - MPEM ejecución de la aplicación pasajero en caso pasajero.

La información de la demanda del viaje y del pasajero llegarán a los conductores, estos podrán rechazar la demanda o aceptarla para luego enviar su precio a cobrar por el viaje, Observar la Figura 5.15 - MPEM caso pasajero, conductores reciben demanda de viaje y la Figura 5.16 - MPEM caso pasajero, conductores responden a la demanda de viaje. Cuando los conductores colocan un precio, la aplicación del conductor cifrará este precio y realizará la operación de resta bajo el esquema homomórfico de Paillier. El pasajero recibirá el producto cifrado.

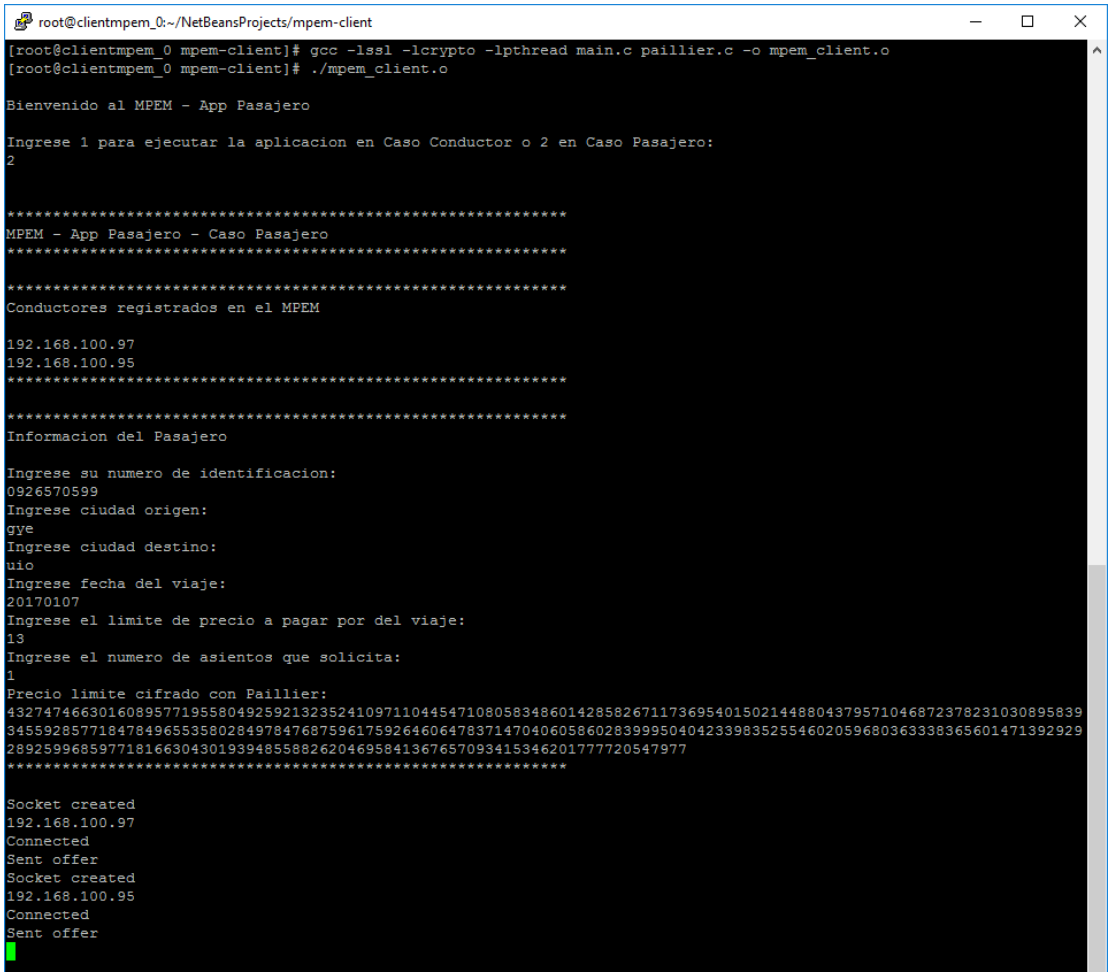


Figura 5.14 - MPEM ejecución de la aplicación pasajero en caso pasajero.

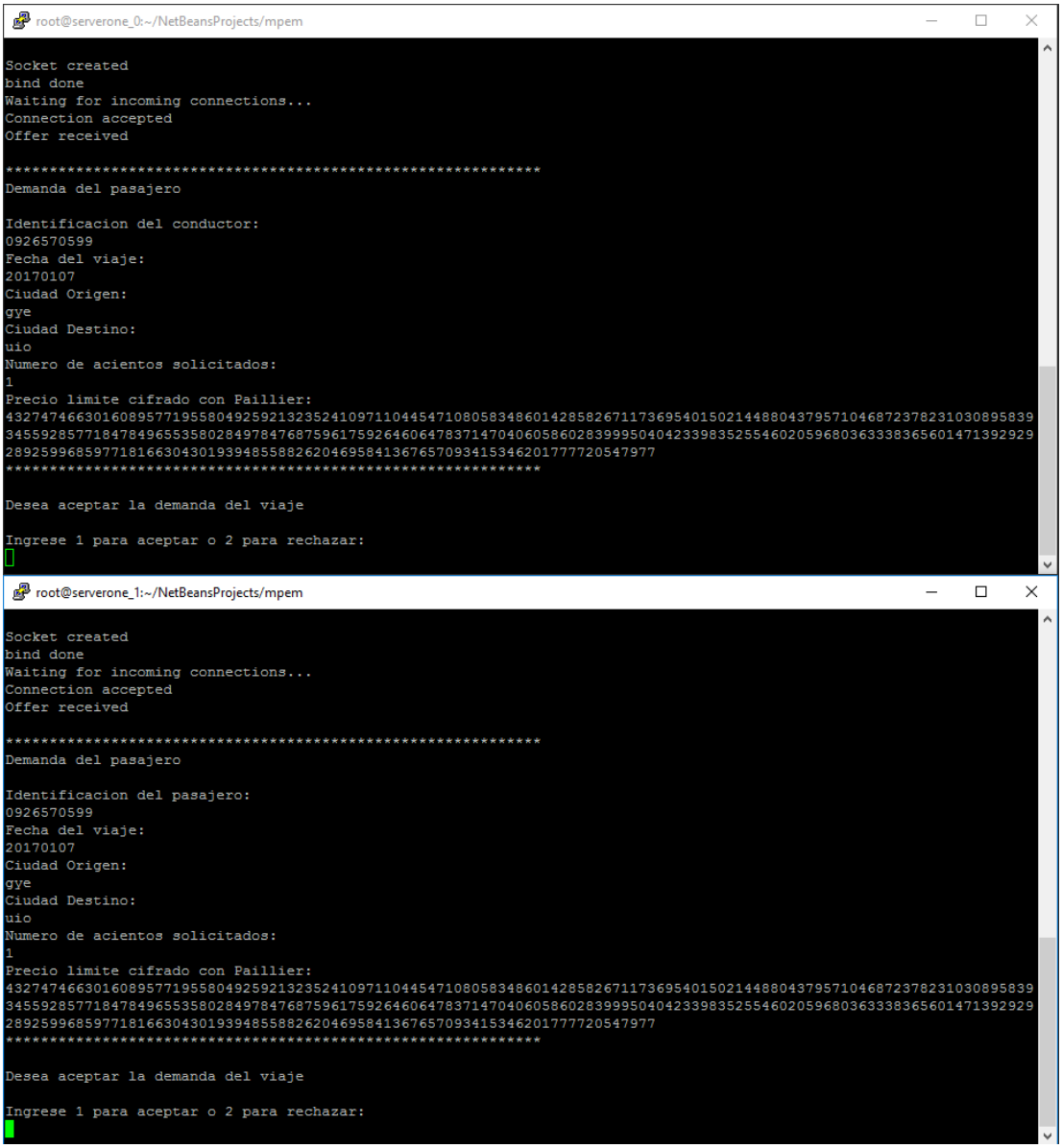


Figura 5.15 - MPEM caso pasajero, conductores reciben demanda de viaje.

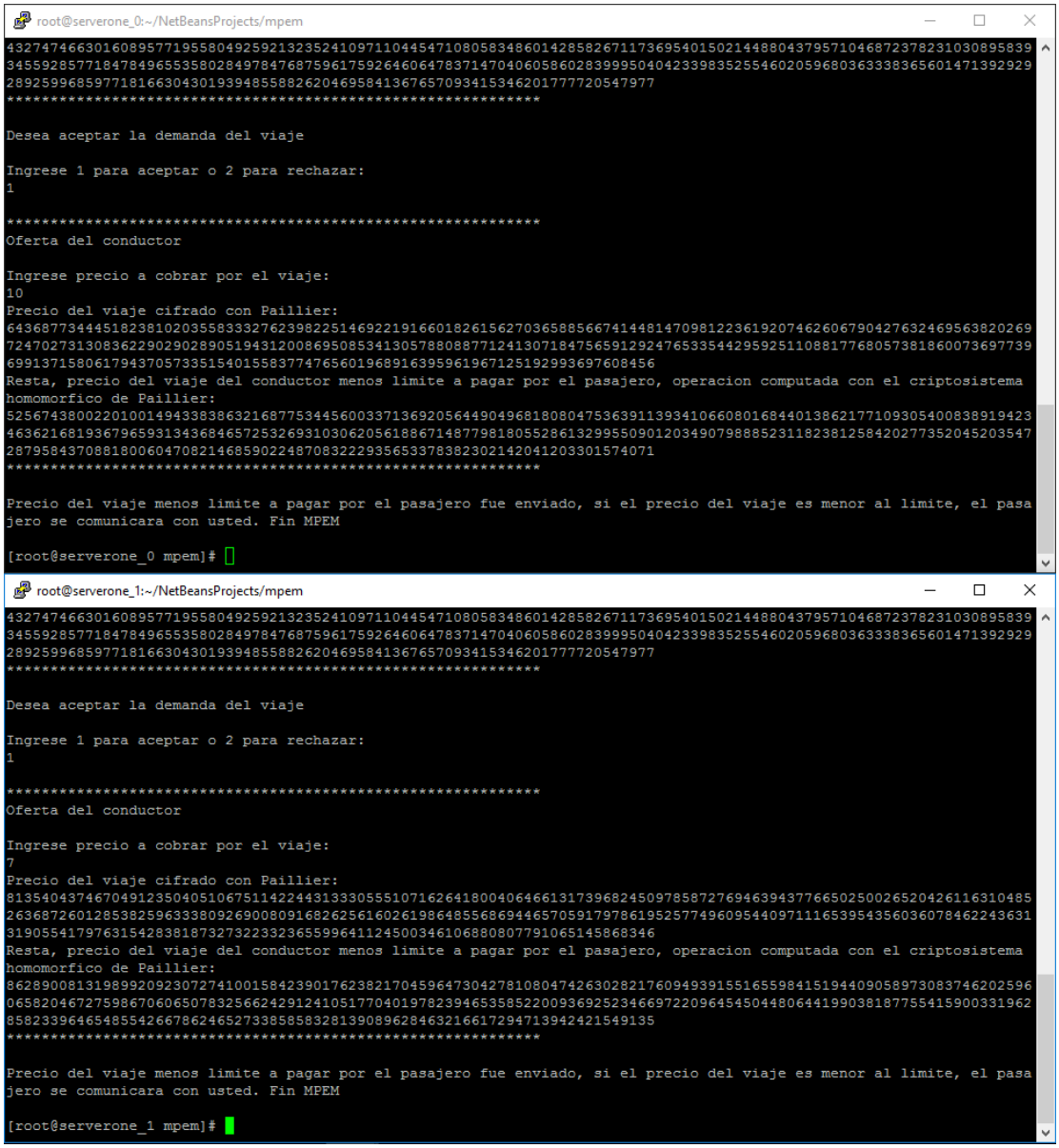


Figura 5.16 - MPEM caso pasajero, conductores responden a la demanda de viaje.

Finalmente las respuestas de los conductores llegaran al pasajero, y la aplicación pasajero evaluará el precio del viaje que se ajusta al límite a pagar, se descartará a los valores que estén por encima del límite y

se aceptarán los valores iguales o menores al límite a pagar por el viaje, cada vez que una oferta es aceptada la aplicación del pasajero guardará la oferta y la identificación del conductor, para que luego pueda evaluar el precio más bajo e identificar al conductor que la ofrece. El resultado de este proceso lo podemos observar en la Figura 5.17 – MPEM aplicación pasajero, evaluación de precio más bajo. Finalmente el pasajero podrá llamar al contacto del conductor cuyo precio por el viaje sea el más bajo.

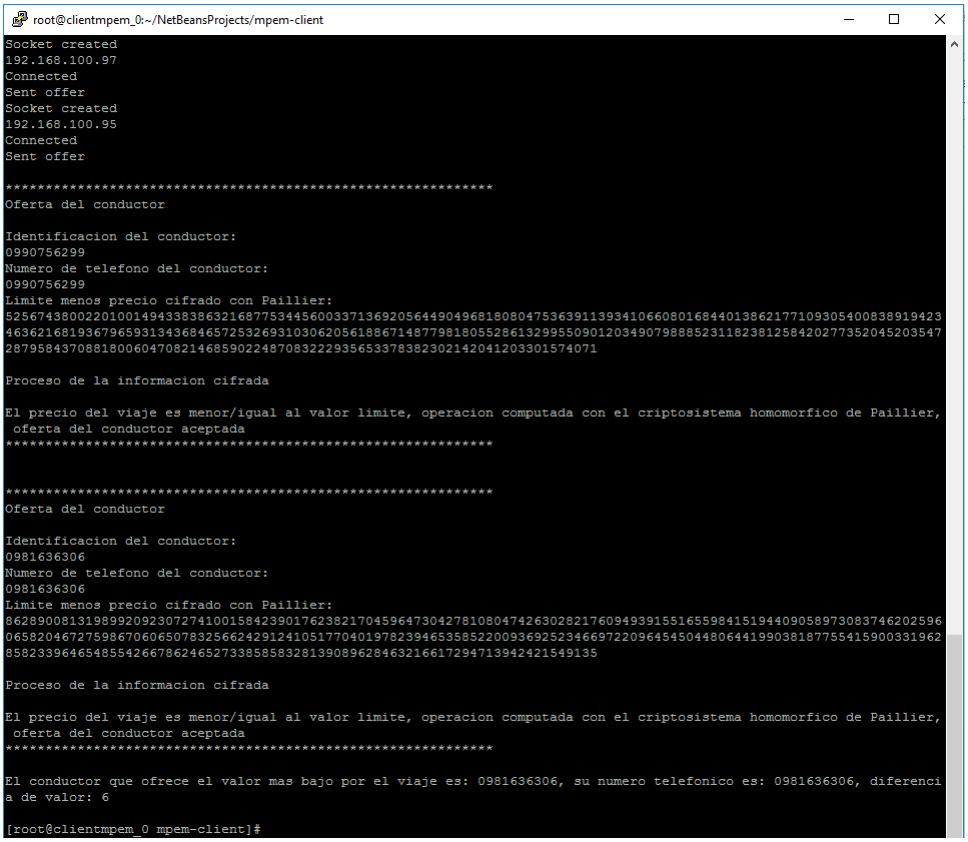


Figura 5.17 - MPEM aplicación pasajero, evaluación de precio más bajo.

CAPÍTULO 6

6. ANÁLISIS DE RESULTADOS

En el siguiente capítulo analizaremos los resultados de las pruebas de la simulación del MPEM en una Red oportunista sobre la ciudad de Guayaquil, y las pruebas de implementación del cifrado homomórfico para el MPEM donde se ejecutan los dos casos de uso en las aplicaciones conductor y pasajero. Finalmente presentaremos los resultados de la implementación del mecanismo de protección de la equidad de mercado MPEM utilizando cifrado homomórfico con el criptosistema de Paillier, sobre una Red tolerante a retardos Oportunista.

6.1. Análisis e interpretación de la simulación de la Red oportunista

Los parámetros de rendimiento a evaluar son los siguientes:

- **ping/pong success prob.** Muestra la probabilidad de entrega de mensajes ping y pong.

- **overhead_ratio.** Es el número de copias de un mensaje por cada mensaje creado, refleja el costo de transmisión en la Red.

- **latency_avg.** Es el tiempo en segundos en el que un mensaje pasa del estado creado ha entregado.

- **hopcount_avg.** Indica el número de nodos por los que un mensaje ha cruzado para llegar a su destino, sin contar el nodo origen.

- **buffertime_avg.** Indica el tiempo en segundos en el que un mensaje ha permanecido en el almacenamiento Buffer de un nodo.

Analizaremos los resultados de las simulaciones comparando las estadísticas de ping y rendimiento al usar los protocolos DTN Epidemic y SprayAndWait, en comunidades de conductores y pasajeros de 5:20 y de 20:80. Para los casos conductor y pasajero.

6.1.1. Análisis de resultados de la simulación Caso Conductor

Caso Conductor comunidad 5:20. La Tabla 19 – Estadísticas de simulaciones con Epidemic y SAW en 5:20 Caso Conductor, nos muestra una comparación del rendimiento de ambos protocolos de enrutamiento DTN.

En ambas simulaciones se generaron 100 pings durante los primeros 209 segundos en 12 horas de simulación. El uso del protocolo DTN Epidemic presenta una probabilidad de entrega de pings del 100%, en comparación al 85% que se produce con el uso del protocolo SprayAndWait. Por otro lado, SprayAndWait mejora en su probabilidad de entrega de pongs con un 54.12% de entregas satisfactorias en comparación al protocolo Epidemic que entregó pongs con una probabilidad del 50%. La tabla 19 nos indica que la probabilidad de entrega de pings y pongs satisfactorios utilizando el protocolo de enrutamiento DTN Epidemic es 50%, y utilizando el protocolo SprayAndWait es 46%. Por lo que podemos observar que en el campo de entregas satisfactorias el protocolo Epidemic supera por poco a SprayAndWait.

Tabla 19 - Estadísticas de simulaciones con Epidemic y SAW en 5:20 Caso Conductor.

Ping stats for scenario MPEM_GYE_CASO_CONDUCTOR_EPIDEMIC_5_20	**Ping stats for scenario MPEM_GYE_CASO_CONDUCTOR_SAW_5_20**
sim_time: 43200.1000	sim_time: 43200.1000
pings sent: 100	pings sent: 100
pings received: 100	pings received: 85
pongs sent: 100	pongs sent: 85
pongs received: 50	pongs received: 46
ping delivery prob: 1.0000	ping delivery prob: 0.8500
pong delivery prob: 0.5000	pong delivery prob: 0.5412
ping/pong success prob: 0.5000	ping/pong success prob: 0.4600
Message stats for scenario MPEM_GYE_CASO_CONDUCTOR_EPIDEMIC_5_20	**Message stats for scenario MPEM_GYE_CASO_CONDUCTOR_SAW_5_20**
overhead_ratio: 48.0000	overhead_ratio: 9.5802
latency_avg: 3605.7984	latency_avg: 4882.0086
hopcount_avg: 4.0895	hopcount_avg: 2.3642
buffertime_avg: 14341.6849	buffertime_avg: 15966.5109

Por otro lado la tasa de sobrecarga o inundación de la Red utilizando el protocolo Epidemic es de 48 copias por mensaje creado, este valor es mucho mayor que SprayAndWait con 9.58 transmisiones, otro dato que no beneficia al protocolo Epidemic es el promedio de saltos que debe tener un mensaje para llegar a su destino, el protocolo Epidemic con un promedio de saltos de 4 es mayor al promedio de saltos que presenta el protocolo SprayAndWait con 2.36 saltos; esto es justificable debido a que el protocolo Epidemic inunda la Red oportunista con más conexiones que SprayAndWait.

El protocolo de enrutamiento DTN Epidemic presenta un mejor promedio de latencia con 3065.79s contra el protocolo SprayAndWait con 4882.0s, lo que indica que los mensajes que se transmiten con el protocolo Epidemic llegan más rápido a su destino. El promedio de tiempo de almacenamiento de data en Buffer es menor al utilizar el protocolo Epidemic con 14341.68s contra SprayAndWait con 15966.51s.

Por lo tanto, podemos determinar que para el escenario 5:20, el protocolo de enrutamiento DTN recomendado es Epidemic debido a que posee mayor probabilidad de entrega de mensajes, sobre todo en la primera fase del esquema de intercambio de mensajes, permitiendo a los conductores entregar su oferta a todos los pasajeros registrados. Además es necesario considerar que la tasa de sobrecarga o inundación de la Red utilizando el protocolo Epidemic generó mayor consumo de recursos en la Red, por lo que no se considera óptimo el uso del protocolo Epidemic en implementaciones del MPEM en comunidades con mayor número de usuarios. Esta deducción la verificaremos en los siguientes párrafos.

Caso Conductor comunidad 20:80. Las estadísticas de rendimiento y entrega de mensajes ping de las simulaciones en comunidades de usuarios 20:80 se pueden observar en la Tabla 20 - Estadísticas de simulaciones con Epidemic y SAW en 20:80 Caso Conductor. Ambas simulaciones generaron 1600 pings durante los primeros 809 segundos en 12 horas de simulación.

Tabla 20 - Estadísticas de simulaciones con Epidemic y SAW en 20:80 Caso Conductor.

Ping stats for scenario	**Ping stats for scenario**
MPEM_GYE_CASO_CONDUCTOR_EPIDEMIC_20_80	**MPEM_GYE_CASO_CONDUCTOR_SAW_20_80**
sim_time: 43200.1000	sim_time: 43200.1000
pings sent: 1600	pings sent: 1600
pings received: 1600	pings received: 1476
pongs sent: 1600	pongs sent: 1476
pongs received: 673	pongs received: 848
ping delivery prob: 1.0000	ping delivery prob: 0.9225
pong delivery prob: 0.4206	pong delivery prob: 0.5745
ping/pong success prob: 0.4206	ping/pong success prob: 0.5300
Message stats for scenario	**Message stats for scenario**
MPEM_GYE_CASO_CONDUCTOR_EPIDEMIC_20_80	**MPEM_GYE_CASO_CONDUCTOR_SAW_20_80**
overhead_ratio: 123.0000	overhead_ratio: 9.0806
latency_avg: 2428.5147	latency_avg: 4572.9764
hopcount_avg: 5.6354	hopcount_avg: 2.6922
buffertime_avg: 15373.3085	buffertime_avg: 17278.1216

Los resultados demuestran que al utilizar el protocolo Epidemic se genera una sobrecarga elevada de transmisión de mensajes por

mensaje creado, en total 123 transmisiones contra 9 transmisiones bajo el protocolo SprayAndWait, esto se refleja en el promedio de saltos donde un mensaje bajo el protocolo Epidemic atraviesa 5.6 nodos para llegar a su destino, contra 2.69 saltos de nodos bajo el protocolo SprayAndWait. Por otro lado, con el protocolo SprayAndWait un mensaje permanece en promedio 17278.12s en el almacenamiento Buffer de un nodo, es un tiempo mayor comparado a los 15373.30s de buffertime_avg experimentado bajo el protocolo Epidemic. La latencia también nos indica que un mensaje creado y transmitido bajo el protocolo Epidemic llega más rápido que si fuese transmitido bajo el protocolo SprayAndWait, con 2428.51s contra 4572.97s. Lo que nos indica que un conductor debe esperar por una respuesta del pasajero en promedio 40 minutos bajo el protocolo Epidemic y 1 hora con 16 minutos bajo el protocolo SprayAndWait.

Finalmente analizamos las estadísticas de entrega de ping que nos indican la probabilidad de entrega de mensajes en ambos protocolos de enrutamiento DTN. Con el protocolo Epidemic observamos que la probabilidad de entrega de pings es del 100%, pero la probabilidad de entregar pongs es de 42%. Por otro lado observamos que los mensajes enviados bajo el protocolo SprayAndWait experimentan una probabilidad del 92.25% de entrega de pings y superan al protocolo

Epidemic en la entrega de pongs con una probabilidad de 57,45%. Estos resultados indican que la probabilidad de entrega satisfactoria de pings y pongs bajo el protocolo de enrutamiento tolerante a retardo Epidemic es del 42% y bajo el protocolo SprayAndWait es del 53%.

Por lo tanto, se determina que para este escenario de 20 conductores y 80 pasajeros en el caso conductor, el protocolo recomendado para la Red oportunista es SprayAndWait, debido que posee una probabilidad de entrega de mensajes mayor que el protocolo Epidemic, además de que la tasa de sobrecarga de la Red es mucho menor, así como los saltos que debe realizar un mensaje para llegar a su destino, de esta forma optimizamos los recursos de la Red oportunista y evitamos su inundación.

6.1.2. Análisis de resultados de la simulación Caso Pasajero

Caso Pasajero comunidad 5:20. La Tabla 21 – Estadísticas de simulaciones con Epidemic y SAW en 5:20 Caso Pasajero, nos muestra el resultado estadístico de rendimiento y entrega de pings, que nos sirven para analizar y determinar el protocolo de enrutamiento óptimo o recomendado para el MPEM en el caso pasajero.

Tabla 21 - Estadísticas de simulaciones con Epidemic y SAW en 5:20 Caso Pasajero.

Ping stats for scenario MPEM_GYE_CASO_PASAJERO_EPIDEMIC_5_20	**Ping stats for scenario MPEM_GYE_CASO_PASAJERO_SAW_5_20**
sim_time: 43200.1000	sim_time: 43200.1000
pings sent: 100	pings sent: 100
pings received: 100	pings received: 93
pongs sent: 100	pongs sent: 93
pongs received: 40	pongs received: 39
ping delivery prob: 1.0000	ping delivery prob: 0.9300
pong delivery prob: 0.4000	pong delivery prob: 0.4194
ping/pong success prob: 0.4000	ping/pong success prob: 0.3900
Message stats for scenario MPEM_GYE_CASO_PASAJERO_EPIDEMIC_5_20	**Message stats for scenario MPEM_GYE_CASO_PASAJERO_SAW_5_20**
overhead_ratio: 48.0000	overhead_ratio: 9.0500
latency_avg: 3664.0940	latency_avg: 5192.1507
hopcount_avg: 3.8267	hopcount_avg: 2.6571
buffertime_avg: 14312.2841	buffertime_avg: 15851.8563

En este caso, los 20 pasajeros envían una demanda de viaje a cada uno de los 5 conductores, por lo que se generan 20 mensajes cada 10 segundos durante los primeros 59 segundos en 12 horas de simulación. Podemos observar en la tabla 21 que la probabilidad de entrega de pings (Fase 1 del esquema de intercambio de mensajes) es de 100% con el protocolo Epidemic, y 93% con el protocolo SprayAndWait. Para la Fase 2 del esquema de intercambio de mensajes, la probabilidad de entrega de pongs es de 40% con el protocolo Epidemic y 41.94% con el protocolo SprayAndWait. En base a estas probabilidades se determina que la probabilidad de completar

el esquema de intercambio de mensajes del MPEM representado por pings y pongs es de 40% con el protocolo Epidemic y 39% con el protocolo SprayAndWait.

Por otro lado tenemos a las estadísticas de rendimiento que nos demuestran que la sobrecarga en la Red generada por el protocolo Epidemic es mayor al protocolo SprayAndWait con 48 trasmisiones contra 9. Además el tiempo de latencia que se experimenta al usar el protocolo Epidemic es menor comparado con el protocolo SprayAndWait con 3664s contra 5192s. El parámetro de número de saltos en promedio nos indica que un mensaje trasmitido con el protocolo Epidemic atraviesa por 3.8 hosts para llegar a su destino, mientras que con el protocolo SprayAndWait un mensaje pasaría por 2.6 hosts. Finalmente el tiempo de almacenamiento de datos en Buffer con el protocolo Epidemic es de 14312s contra 15851s experimentado con el protocolo SprayAndWait.

Con los resultados estadísticos de entregas de ping y rendimiento podemos determinar que el protocolo recomendado para el MPEM en una comunidad 5:20 de conductores y pasajeros en el caso pasajero es SprayAndWait, el cual posee un porcentaje menos de probabilidad

de entrega satisfactoria de pings y pongs que el protocolo Epidemic, mayor tiempo en entregar mensajes al destino, datos estadísticos similares en el tiempo de almacenamiento de data en Buffer, pero genera menos sobrecarga en la Red y menos hosts involucrados en la transmisión de mensajes lo que lo vuelve un factor determinante para obtener un esquema de intercambio de mensajes óptimo y un mejor uso de recursos de la Red distribuida, evitando inundarla.

Tabla 22 - Estadísticas de simulaciones con Epidemic y SAW en 20:80 Caso Pasajero.

Ping stats for scenario MPEM_GYE_CASO_PASAJERO_EPIDEMIC_20_80	**Ping stats for scenario MPEM_GYE_CASO_PASAJERO_SAW_20_80**
sim_time: 43200.1000	sim_time: 43200.1000
pings sent: 1600	pings sent: 1600
pings received: 1600	pings received: 1539
pongs sent: 1600	pongs sent: 1539
pongs received: 386	pongs received: 449
ping delivery prob: 1.0000	ping delivery prob: 0.9619
pong delivery prob: 0.2413	pong delivery prob: 0.2917
ping/pong success prob: 0.2413	ping/pong success prob: 0.2806
Message stats for scenario MPEM_GYE_CASO_PASAJERO_EPIDEMIC_20_80	**Message stats for scenario MPEM_GYE_CASO_PASAJERO_SAW_20_80**
overhead_ratio: 122.9403	overhead_ratio: 9.0264
latency_avg: 2281.5175	latency_avg: 3987.9358
hopcount_avg: 5.8006	hopcount_avg: 2.6796
buffertime_avg: 15460.9512	buffertime_avg: 17230.8625

6.2. Análisis e interpretación de la implementación del cifrado homomórfico

La implementación del MPEM para los casos conductor y pasajero fue exitosa, el esquema de intercambio de mensajes se realizó con el protocolo TCP, mediante las librerías de comunicación sys/socket.h y arpa/inet.h. Las estructuras de datos y todos los tipos de variables fueron transformadas a Char para que puedan ser transportadas desde un conductor a un pasajero y viceversa. Por lo tanto, los precios cifrados así como la llave pública de Paillier que se codifican con tipos de datos Bignum fueron transformados a Char y enviados a través de la Red de datos; este proceso fue únicamente de transformación y no de descifrado, por lo que la información viajó de manera íntegra y confiable. Los precios transformados de Bignum a Char se pueden observar en las Figuras 5.9, 5.10, 5.11, 5.12, 5.14, 5.15, 5.16, 5.17.

El MPEM es una aplicación sencilla que utiliza las propiedades homomórficas del criptosistema de Paillier para realizar operaciones de resta con el objetivo de determinar si una oferta de viaje se encuentra dentro de los límites de precio y además calcular el valor de la oferta más baja. La aplicación conductor y pasajero cubren los dos casos de uso posibles desde el punto de vista del conductor y

pasajero, además, se validó que su diseño y esquema de intercambio de mensajes se acopla a un sistema distribuido como una Red tolerante a retardos oportunista, donde todos los nodos funcionan como servidor o cliente, y son capaces de reenviar paquetes a sus destinos.

Las funciones utilizadas en el MPEM que involucran al criptosistema homomórfico de Paillier fueron la creación de llave pública y privada, el cifrado y descifrado de valores, y la resta de dos valores cifrados. Los resultados obtenidos en la ejecución de funciones que utilizan el criptosistema homomórfico de Paillier fueron exitosos, esperados y precisos.

Por lo tanto podemos determinar que es factible el uso de cifrado homomórfico para realizar operaciones matemáticas básicas como la suma, resta y multiplicación, en funciones o módulos que integren una aplicación que tenga como objetivo proteger la privacidad de los datos en la entrada, proceso y salida de la información. Lo que abre a la posibilidad de realizar programas confiables, seguros y útiles, como es en este caso un mecanismo de protección de equidad de mercado para un sistema de automóvil compartido.

6.3. Presentación de resultados de la implementación del MPEM

De los resultados expuestos en los Subcapítulos 6.1 y 6.2 se puede determinar que el protocolo de enrutamiento tolerante a retardo óptimo para el MPEM tanto en comunidades pequeñas y grandes, y para los casos conductor y pasajero es SprayAndWait, las comparaciones de rendimiento contra Epidemic se pueden observar en las siguientes Figuras.

La Figura 6.1 – Estadísticas de sobrecarga de mensajes en la Red oportunista, nos muestra que el protocolo Epidemic no es eficiente para el esquema distribuido sobre el cual trabaja el MPEM, este protocolo produce un alto uso de recursos de la Red comparado con SprayAndWait. El valor de tasa de sobrecarga de SprayAndWait es aproximado a 9 en todas las simulaciones, esto se debe a que la configuración de número de copias por mensaje creado es de 10, este valor se definió en la Tabla 10 - Configuración por defecto de enrutamiento SprayAndWait.

La tasa de sobrecarga de la Red se refleja en el número de saltos que debe dar un mensaje para llegar a su destino. La Figura 6.2 – Número promedio de saltos de un mensaje en la Red oportunista, nos

demuestra que los mensajes enviados bajo el protocolo SprayAndWait pasan en promedio por 2.6 nodos en cambio el protocolo Epidemic refleja mayor uso de nodos para que un mensaje pueda llegar a su destino.

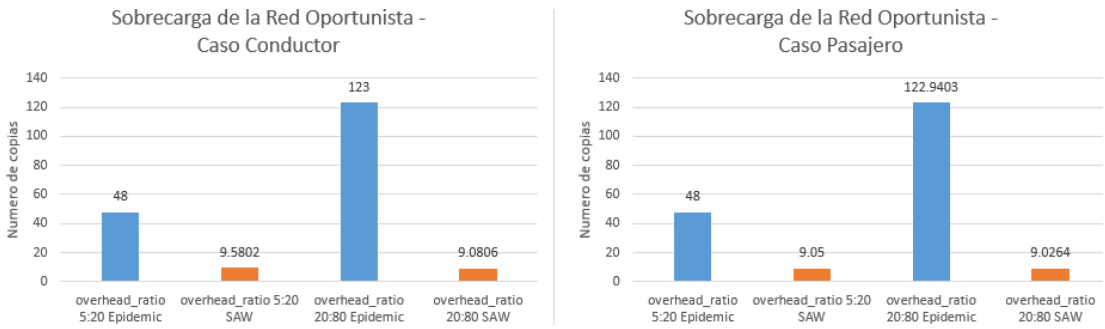


Figura 6.1 - Estadísticas de sobrecarga de mensajes en la Red oportunista.

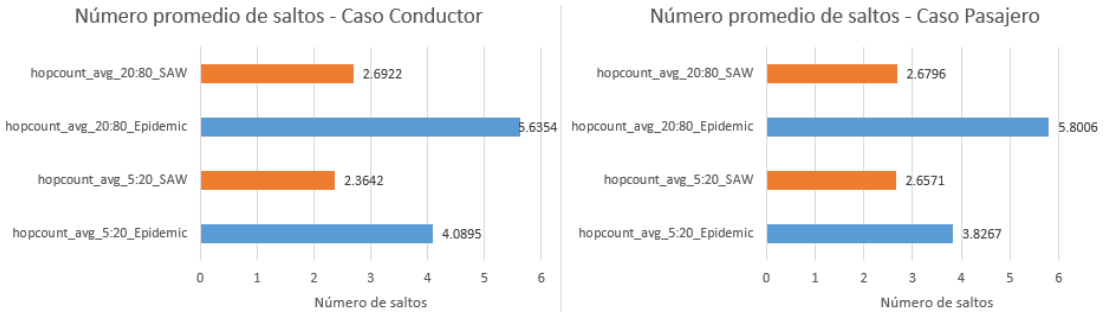


Figura 6.2 - Número promedio de saltos de un mensaje en la Red oportunista.

El tiempo en que un mensaje es creado y entregado es representado por el parámetro latency_avg. Los resultados de las simulaciones

indican que los mensajes enviados con el protocolo Epidemic llegan más rápido a su destino que con el protocolo SprayAndWait. La comparación del tiempo promedio de entrega de mensajes con ambos protocolos se puede observar en la Figura 6.3 - Tiempo promedio de entrega de mensajes en la Red oportunista.

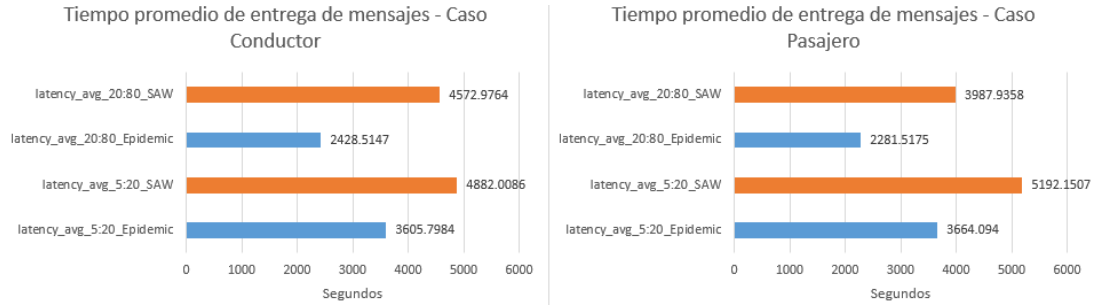


Figura 6.3 - Tiempo promedio de entrega de mensajes en la Red oportunista.

El tiempo promedio en que un mensaje se almacena en el Buffer de un nodo también favorece al protocolo Epidemic, ya que bajo este protocolo los mensajes duran menos en el Buffer que al usar el protocolo SprayAndWait. Al observar la Figura 6.4 – Tiempo promedio de almacenamiento de mensajes en Buffer, podemos determinar que SprayAndWait retiene más tiempo los mensajes en Buffer que al usar el protocolo Epidemic, para los casos conductor y pasajero y en comunidades de 5:20 y 20:80.

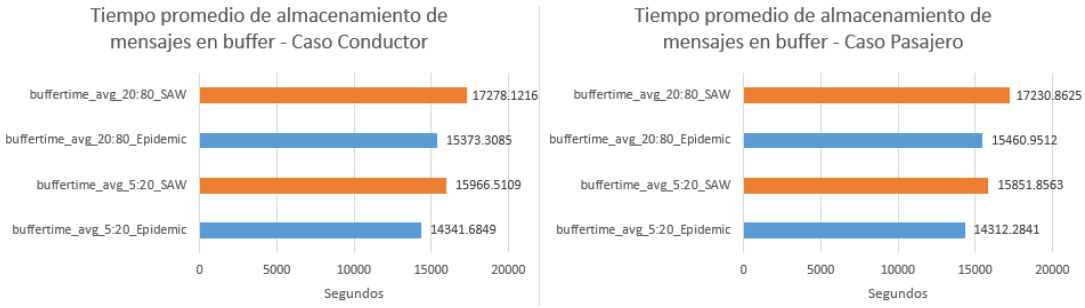


Figura 6.4 - Tiempo promedio de almacenamiento de mensajes en Buffer.

Finalmente presentamos los resultados de probabilidad de entrega de pings y pongs en ambos casos de uso y ambas comunidades, en la Figura 6.5 – Probabilidad de entrega ping/pong en la Red oportunista. La Figura 6.5 demuestra que SprayAndWait produce una probabilidad de entrega satisfactoria de ping y pong superior al protocolo Epidemic en comunidades grandes, en nuestro caso 20 conductores y 80 pasajeros. Pero en comunidades pequeñas de 5 conductores y 20 pasajeros la probabilidad de entrega satisfactoria de pings y pongs es ligeramente inferior o igual al usar el protocolo Epidemic. Debido al costo de uso de recursos de la Red oportunista se concluye que el protocolo óptimo y eficiente para la implementación MPEM en una Red distribuida tolerante a retardos oportunista es SprayAndWait.

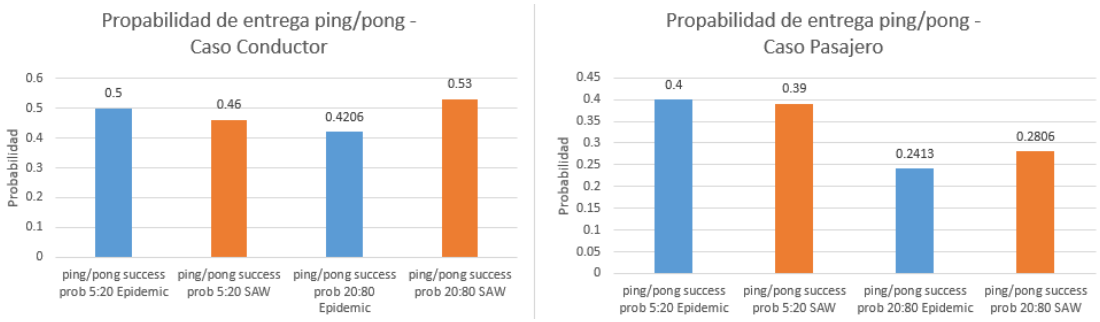


Figura 6.5 - Probabilidad de entrega ping/pong en la Red oportunista.

Por otro lado podemos determinar que el desarrollo e implementación del MPEM con el criptosistema homomórfico de Paillier bajo un esquema de intercambio de mensajes que simula nuestra Red distribuida, fue satisfactorio y cumple a los casos de uso planteados en el Subcapítulo 4.4 y siguen el flujo de procesos que se diseñaron y expusieron en las Figuras 4.4, 4.5, 4.6, 4.7. La ejecución del MPEM donde se representa el esquema de intercambio de mensajes para una Red oportunista se pueden observar en las Figuras del 5.8 a la 5.17.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Se ha investigado y estudiado a las Redes tolerantes a retardos al igual que a los sistemas de cifrado homomórfico en especial al criptosistema de Paillier; además se ha repasado la base teórica de las actuales y potenciales aplicaciones de las Redes tolerantes a retardos y criptosistemas de cifrado homomórfico, así como técnicas relacionas al mecanismo de protección de equidad de mercado de un sistema de automóvil compartido.

2. Se ha realizado un levantamiento de información que data desde la historia, aplicaciones y uso de los sistemas de automóviles compartidos en Ecuador y el resto del mundo, hasta obtener información de rutas de la ciudad de Guayaquil y precios de los viajes interprovinciales más frecuentes realizados desde el Terminal Terrestre de Guayaquil. Por otro lado se ha investigado, analizado y escogido las herramientas para la simulación de una Red oportunista y las librerías para la implementación de criptografía homomórfica para un mecanismo de protección de equidad de mercado para sistemas de automóvil compartido. Se eligió el uso The ONE como simulador DTN y las librerías Habcast del proyecto PrivHab+, para realizar operaciones homomórficas en el desarrollo de las aplicaciones del MPEM.

3. Se ha analizado los posibles protocolos de enrutamiento tolerantes a retardos que puedan aplicarse al MPEM, para determinar la factibilidad del uso de los protocolos Epidemic y SprayAndWait los cuales guardan la privacidad de ubicación de los usuarios. Se ha diseñado un esquema de intercambio de mensajes para el MPEM con el cual se construye el esquema distribuido basado en una Red tolerante a retardos oportunista, además de las rutas y tipos de nodos que lo conforman. Se ha analizado el uso de diferentes librerías en varios lenguajes de programación para determinar que Habcast en C soporta la resta de

dos números cifrados con el criptosistema homomórfico de Paillier utilizando funciones de mapping. Se ha diseñado casos de usos para conductores y pasajeros, y diagramas de flujos con los cuales se identifican los procesos de la información en las aplicaciones del mecanismo de protección de equidad del mercado.

4. En las simulaciones se ha configurado casos de uso para los usuarios conductor y pasajero, en dos tipos de comunidades de usuarios que comprenden 5 conductores con 20 pasajeros, y 20 conductores con 80 pasajeros. Finalmente se ha configurado dos tipos de enrutamiento DTN que involucran a los protocolos Epidemic y SprayAndWait. Por lo tanto estas configuraciones han derivado en 8 simulaciones con las cuales se ha obtenido información suficiente para determinar requisitos de análisis, escalamiento y viabilidad técnica del MPEM en una Red oportunista. Se ha logrado desarrollar el MPEM en base al diseño de los casos de uso y los diagramas de flujo de procesos, por lo cual se ha creado una aplicación para los conductores y otra para los pasajeros, con funciones de generación de llaves de Paillier y resta homomórfica. Finalmente se ha logrado implementar y ejecutar las aplicaciones del MPEM obteniendo resultados esperados.

5. Se ha logrado obtener estadísticas de entregas de mensajes y rendimiento del esquema de intercambio de mensajes del MPEM en una Red oportunista a partir de las simulaciones. El análisis e interpretación de estos reportes determinan que el protocolo de enrutamiento DTN óptimo y recomendado es SprayAndWait debido a que economiza los recursos de la Red distribuida y presenta una probabilidad de entrega satisfactoria de mensajes mayor o simular a la probabilidad de entrega de mensajes generada por el protocolo Epidemic en los dos tipos de comunidades de usuarios simuladas.

6. Se ha logrado ejecutar el MPEM en varias máquinas virtuales con Sistema Operativo Linux CentOS 7 x86_64 dedicadas para conductores y pasajeros. Las aplicaciones conductor y pasajero han logrado intercambiar precios de viajes cifrados y operar estos datos cifrados mediante la función de resta homomórfica desarrollada en este proyecto, los resultados obtenidos son exactos y precisos al igual que si fueran operados sin cifrar, lo que demuestra la factibilidad de uso de cifrado homomórfico para aplicaciones que se propongan como objetivo proteger la privacidad de los datos en la entrada, proceso y salida de la información. Se ha logrado desarrollar un programa confiable, seguro y útil, que es el mecanismo de protección de equidad de mercado para un sistema de automóvil compartido.

7. Se ha logrado diseñar e implementar un mecanismo de equidad de mercado para un sistema de automóvil compartido que se respalda en estadísticas de entrega de mensajes y rendimiento satisfactorios en la simulación del esquema de intercambio de mensajes del MPEM en una Red distribuida conformada por una Red tolerante a retardos oportunista. Además, la ejecución satisfactoria de operaciones basadas en criptografía homomórfica de Paillier han validado la capacidad del MPEM de determinar la aceptación de viajes en base a precios límites y por la determinación de cotizaciones más bajas, resolviendo operaciones con precios cifrados durante todos los procesos del esquema de intercambio de mensajes del MPEM.

Recomendaciones

1. Para la implementación real del MPEM sobre una Red oportunista se recomienda como nodo el uso de Raspberry Pi 3 model B, el cual posee integrado una interfaz WiFi 802.11n y soporte de tarjeta de almacenamiento externa MicroSDHC cuya capacidad puede ser de 2GB hasta 32GB [55]. Este dispositivo soporta un Sistema Operativo Linux básico en el cual se puede desplegar el MPEM. Además, para encapsular los paquetes TCP en paquetes Bundles para la transmisión de mensajes dentro de una Red tolerante a retardo oportunista, se

recomienda utilizar e integrar el MPEM a la plataforma aDTN la cual se encuentra desarrollada en los lenguajes de programación C y JAVA, y nos permite reenviar, almacenar mensajes en Buffer y enrutar paquetes Bundle a su destino dentro de un Red tolerante a retardos [59].

BIBLIOGRAFÍA

[1] Grupo El Comercio, En Quito hay una aplicación para compartir el carro, http://www.elcomercio.com/actualidad/quito/hay-aplicacion-compartir-carro.html, fecha de consulta junio 2016.

[2] Ministerio de Turismo Ecuador, Feriados 2015, http://servicios.turismo.gob.ec/index.php/categoria-feriados/carnaval-2015/143. fecha de consulta junio 2016.

[3] Procuraduría Metropolitana de Quito, Reformatoria Ordenanza Metropolitana No. 0305, http://www7.quito.gob.ec/mdmq_ordenanzas/Sesiones%20del%20Concejo/2014/Sesi%C3%B3n%20Ordinaria%202014-09-04/III.%20%20Primer%20Debate%20-%20Proyecto%20de%20Ordenanza/IC-O-2014-062%20%20%20%20%20Reformatoria%20Ordenanza%20Metropolitana%20No.%200305.pdf, fecha de consulta junio 2016.

[4] Grupo El Comercio, Una red en línea para compartir vehículos, http://www.revistalideres.ec/lideres/red-linea-compartir-vehiculos.html, fecha de consulta junio 2016.

[5] Amey, Andrew y Oliphant, Marc, Dynamic Ridesharing: Carpooling Meets the Information Age, http://ridesharechoices.scripts.mit.edu/home/wp-content/papers/APA_TPD_Webinar_Aug2010.pdf, fecha de consulta junio 2016.

[6] Wauters, Robin, Ride-sharing has arrived in Europe, and the race is on between BlaBlaCar and Carpooling.com, http://tech.eu/features/481/ride-sharing-europe-carpooling-blablacar/, fecha de consulta junio 2016.

[7] BlaBlaCar, About as, https://www.blablacar.co.uk/about-us, fecha de consulta junio 2016.

[8] Naone, Erica, TR10: Cifrado homomórfico, https://www.technologyreview.es/informatica/37710/tr10-cifrado-homomorfico/, fecha de consulta junio 2016.

[9] Riggio, Roberto y Sicari, Sabrina, Secure Aggregation in Hybrid Mesh/Sensor Networks, http://disi.unitn.it/~riggio/lib/exe/fetch.php?media=publications:sasn2009.pdf, fecha de consulta junio 2016.

[10] Gentry, Craig y Halevi Shai, A Working Implementation of Fully Homomorphic Encryption,

http://eurocrypt2010rump.cr.yp.to/9854ad3cab48983f7c2c5a2258e27717.pdf, fecha de consulta junio 2016.

[11] Gupta, C P y Sharma, Iti, A fully homomorphic encryption scheme with symmetric keys with application to private data processing in clouds, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6724526, fecha de consulta junio 2016.

[12] Brakerski, Zvika y Gentry, Craig, Fully Homomorphic Encryption without Bootstrapping, http://eprint.iacr.org/2011/277.pdf, fecha de consulta junio 2016.

[13] Shai, Halevi, An Implementation of homomorphic encryption, https://github.com/shaih/HElib, fecha de consulta junio 2016.

[14] Huang, Chung-Ming, Lan, Kun-chan y Tsai, Chang-Zhou, A Survey of Opportunistic Networks, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4483161&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F4482830%2F4482831%2F04483161, fecha de consulta junio 2016.

[15] Sánchez-Carmona, Adrián, Robles, Sergi, Borrego, Carlos, PrivHab+: A secure geographic routing protocol for DTN,

http://www.sciencedirect.com/science/article/pii/S0140366415003825, fecha de consulta junio 2016.

[16] BlaBlaCar, Como funciona compartir viaje coche, https://www.blablacar.es/como-funciona-compartir-viaje-coche, fecha de consulta junio 2016.

[17] Keränen, Ari, The ONE, https://akeranen.github.io/the-one/, fecha de consulta junio 2016.

[18] Pelusi, Luciana, Opportunistic networking: data forwarding in disconnected mobile ad hoc networks, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4014485&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4014485, fecha de consulta junio 2016.

[19] Stuntz, Craig, What is Homomorphic Encryption, and Why Should I Care?, http://community.embarcadero.com/blogs/entry/what-is-homomorphic-encryption-and-why-should-i-care-38566, fecha de consulta junio 2016.

[20] Sen, Jaydip, Homomorphic Encryption: Theory & Application, https://arxiv.org/ftp/arxiv/papers/1305/1305.5886.pdf, fecha de consulta junio 2016.

[21] Wikipedia, Cifrado homomórfico, https://es.wikipedia.org/wiki/Cifrado_homom%C3%B3rfico#cite_note-1, fecha de consulta junio 2016.

[22] Google Code, The Homomorphic Encryption Project, https://code.google.com/archive/p/thep/, fecha de consulta junio 2016.

[23] GitHub, Inc, javallier, https://github.com/NICTA/javallier, fecha de consulta junio 2016.

[24] Warthman, Forrest, Delay- and Disruption-Tolerant Networks (DTNs), http://ipnsig.org/wp-content/uploads/2012/07/DTN_Tutorial_v2.05.pdf, fecha de consulta junio 2016.

[25] McMahon, Alex y Farrell, Stephen, Delay- and disruption-tolerant networking (DTN), http://www.tara.tcd.ie/bitstream/handle/2262/39153/Delay-%20and%20Disruption-Tolerant%20Networking.pdf;jsessionid=089AC5B9739EC081B5D9A419611244EB?sequence=1, fecha de consulta junio 2016.

[26] Kruse, Hans, Ivancic, Will, Scott, Keith y Ostermann, Shawn, Delay-Tolerant Networking Research Group (DTNRG), https://sites.google.com/site/dtnresgroup/, fecha de consulta julio 2016.

[27] KU Leuven ESAT/COSIC, Homomorphic Encryption Applications and Technology, https://heat-project.eu/index.html, fecha de consulta julio 2016.

[28] IPNSIG, InterPlanetary Networking Special Interest Group, http://ipnsig.org/, fecha de consulta junio 2016.

[29] Moreira, Waldir y Mendes, Paulo, Opportunistic Networking: Extending Internet Communications Through Spontaneous Networks, http://copelabs.ulusofona.pt/scicommons/index.php/publications/show/274, fecha de consulta junio 2016.

[30] Boldrini, Chiara, Conti, Marco, Jacopini, Jacopo y Passarella, Andrea, HiBOp: a History Based Routing Protocol for Opportunistic Networks, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4351716, fecha de consulta junio 2016.

[31] Moreira, Waldir y Mendes, Paulo, «Survey on Opportunistic Routing for Delay/Disruption Tolerant Networks, http://copelabs.ulusofona.pt/scicommons/index.php/publications/show/31, fecha de consulta junio 2016.

[32] Vahdat, Amin y Becker, David, Epidemic Routing for Partially-Connected Ad Hoc Networks, issg.cs.duke.edu/epidemic/epidemic.pdf, fecha de consulta junio 2016.

[33] Rouse, Margaret, Homomorphic encryption, http://searchsecurity.techtarget.com/definition/homomorphic-encryption, fecha de consulta julio 2016.

[34] Greenberg, Andy, Hacker Lexicon: What is homomorphic encryption?, https://www.wired.com/2014/11/hacker-lexicon-homomorphic-encryption/, fecha de consulta julio 2016.

[35] Fouque , Pierre-Alain, Poupard, Guillaume y Stern, Jacques, Sharing Decryption in the Context of Voting or Lotteries, http://www.di.ens.fr/~stern/data/St85.pdf, fecha de consulta julio 2016.

[36] Sen, Jaydip y Maitra, Subhamoy, An Attack on Privacy Preserving Data Aggregation Protocol for Wireless Sensor Networks, http://link.springer.com/chapter/10.1007%2F978-3-642-29615-4_15, fecha de consulta julio 2016.

[37] Wikipedia, Paillier cryptosystem, https://en.wikipedia.org/wiki/Paillier_cryptosystem, fecha de consulta julio 2016.

[38] Choinyambuu, Sansar, Homomorphic Tallying with Paillier Cryptosystem, http://security.hsr.ch/msevote/seminar-papers/HS09_Homomorphic_Tallying_with_Paillier.pdf, fecha de consulta julio 2016.

[39] Sánchez-Carmona, Adrián, Robles, Sergi, Borrego, Carlos, PrivHab: a Privacy Preserving Georouting Protocol based on a Multiagent System for Podcast Distribution based on a Multiagent System for Podcast Distribution based on a Multiagent System for Podcast Distribution on Disconnected Areas, Elsevier, 2016.

[40] Terminal Terrestre, Terminal Terrestre Guayaquil, http://ttg.ec/, fecha de consulta julio 2016.

[41] Google, Google Maps, https://www.google.com/maps, fecha de consulta julio 2016.

[42] Cheng, Yu, Eun, Do Young, Qin, Zhiguang, Song, Min y Kai, Xing, Wireless Algorithms, Systems, and Applications, Springer, 2011.

[43] Keränen, J Ari, Ott, Jörg y Kärkkäinen, Teemu, The ONE Simulator for DTN Protocol Evaluation, https://www.netlab.tkk.fi/tutkimus/dtn/theone/pub/the_one_simutools.pdf , fecha de consulta julio 2016.

[44] openjump.project, OpenJUMP, http://www.openjump.org/, fecha de consulta Octubre 2016.

[45] GitHub, Inc, kunerd/jpaillier, https://github.com/kunerd/jpaillier, fecha de consulta julio 2016.

[46] GitHub, Inc, qstokkink/PHENet, https://github.com/qstokkink/PHENet, fecha de consulta julio 2016.

[47] GitHub, Inc, amillevuillaume/Paillier-GMP, https://github.com/camillevuillaume/Paillier-GMP, fecha de consulta julio 2016.

[48] GitHub, Inc, GerardGarcia/paillier-c, https://github.com/GerardGarcia/paillier-c, fecha de consulta julio 2016.

[49] GitHub, Inc, mikeivanov/paillier, https://github.com/mikeivanov/paillier, fecha de consulta julio 2016.

[50] GitHub, Inc, NICTA/python-paillier, https://github.com/NICTA/python-paillier, fecha de consulta julio 2016.

[51] Wikipedia, Routing in delay-tolerant networking, https://en.wikipedia.org/wiki/Routing_in_delay-tolerant_networking, fecha de consulta octubre 2016.

[52] Mehto, Anjula y Chawla, Meenu, Comparing Delay Tolerant Network Routing Protocols for Optimizing L-Copies in Spray and Wait Routing for Minimum Delay, Atlantis Press, 2016.

[53] Wikipedia, IEEE 802.11, https://en.wikipedia.org/wiki/IEEE_802.11#802.11n, fecha de consulta octubre 2016.

[54] Raspberry PI Fundation, Raspberry PI USB WIFI DONGLE, https://www.raspberrypi.org/products/usb-wifi-dongle/, fecha de consulta octubre 2016.

[55] Raspberry PI Fundation, Raspberry PI, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, fecha de consulta diceimbre 2016.

[56] CentOS, The CentOS Project, https://www.centos.org/, fecha de consulta noviembre 2016.

[57] Samaras, Christos y Tsaoussidis, Vassilis, Adjusting transport segmentation policy of DTN Bundle Protocol under synergy with lower layers, http://www.sciencedirect.com/science/article/pii/S0164121210002657, fecha de consulta octubre 2016.

[58] Graphviz, Graphviz - Graph Visualization Software, http://www.graphviz.org/, fecha de consulta enero 2017.

[59] SeNDA, Security of Networks and Distributed Applications, http://senda.uab.es/adtn, fecha de consulta enero 2017.

GLOSARIO

Criptografía: Arte de escribir con clave secreta o de un modo enigmático.

Homomorfismo: Es la correspondencia entre las operaciones con los datos sin cifrar y las operaciones que se realizarán con los datos cifrados. Describe la transformación de un conjunto de datos en otro preservando las relaciones entre los elementos en ambos conjuntos, el término es derivado del griego “misma estructura”.

Criptografía homomórfica: Es un criptosistema que permite realizar cálculos u operaciones matemáticas complejas en datos cifrados sin descifrarlos, donde las mismas operaciones matemáticas - ya sea que se realizan en los datos cifrados o descifrados - darán resultados equivalentes.

Redes oportunistas: Es un tipo de Red que resuelve desafíos de conexión, donde los contactos de la Red son intermitentes o cuando el rendimiento del enlace es muy variable o extremo. En una Red de este tipo, no existe una ruta completa desde el origen al destino la mayor parte del tiempo. Los nodos intermedios pueden tomar la custodia de los datos durante el apagón para

luego transmitirlos cuando la conectividad vuelve a habilitarse, así sucesivamente a la espera de un futuro contacto oportunista con otro nodo para reenviar la información hasta llegar a su destino.

Pico y placa: Medida de restricción que regula el tránsito prohibiendo la circulación vehicular por unas horas cada día, de acuerdo con el último número de su placa, en un área determinada de Quito.

Cómputo multipartidista: Esquema donde las partes interesadas en cómputo común utilizan funciones públicas manteniendo la privacidad de las entradas, por lo tanto la función que debe ser calculada es de conocimiento público, mientras que las entradas son datos cifrados y privados.

Estado del Arte: Es una modalidad de la investigación documental que permite el estudio del conocimiento acumulado (escrito en textos) o técnicas relacionas dentro de un área específica.

Asintóticamente: Dicho de una curva que se acerca de continuo a una recta o a otra curva sin llegar nunca a encontrarla.

Store-and-forward: Mensajes enteros (bloques enteros de datos de usuario) -o- piezas (fragmentos) de tales mensajes que se mueven - (reenvían) desde un lugar de almacenamiento en un nodo a un lugar de almacenamiento en otro nodo, a lo largo de un camino que finalmente llegan al destino.

Telemetría: Conjunto de técnicas para la medida a distancia de magnitudes físicas.

Taxonomía: Ciencia que trata de los principios, métodos y fines de la clasificación. Se aplica en particular, dentro de la biología, para la ordenación jerarquizada y sistemática, con sus nombres, de los grupos de animales y de vegetales.

PrivHab+: Protocolo de enrutamiento geográfico seguro que aprende acerca de los hábitos de movilidad de los nodos de la Red y utiliza esta información de una manera segura.

Bootstrapping: Descodificación que puede ser expresada como un polinomio de grado bajo. Una vez que el grado de polinomios que se puede evaluar por el esquema supera el grado del polinomio descifrado por un factor de dos.

Bundle: Es un protocolo que permite el almacenamiento y envió de paquetes Bundles sobre los nodos, conformando una Red tolerante a retardos, DTN. El protocolo Bundle consiste de tres cosas, 1) una cabecera Bundle, la cual consiste en uno o más bloques DTN insertados por el agente Bundle, 2) data de usuario de la capa de Aplicación, incluyendo cómo manejar la data del usuario e información de control de como procesar, almacenar y disponer.

Epidemic: Protocolo de enrutamiento para DTN basado en técnicas de inundación, cada nodo constantemente replica y transmite mensajes a nuevos contactos que aún no posean una copia del mensaje, hasta que el mensaje llegue a su destino.

PRoPHET: Protocolo de enrutamiento para DTN diseñado para mejorar la probabilidad de entrega de Bundles y reducir la sobrecarga y consumo de recursos de la Red. Si un nodo ha visitado una misma ubicación en varias ocasiones y ha generado encuentros con otros nodos en esta misma ubicación, es probable que este patrón se repita en el futuro, por lo tanto el protocolo guarda y aprende esta información de encuentros pasados para optimizar la entrega de Bundles.

SprayAndWait: Protocolo de enrutamiento basado en replicación, este protocolo limita la estrategia de envío de mensajes del enrutamiento Epidémico agregando un número máximo de copias permitidas de cada mensaje. Este protocolo consiste en dos fases, la primera, Spray, en la cual el nodo emisor envía un mensaje con un número límite N de copias permitidas en la Red, el emisor es el encargado de enviar una copia del mensaje a cada distinto retransmisor. La segunda fase, Wait, los nodos de retrasmisión receptan las copias, y esperan la transmisión directa del mensaje al nodo destino.

Bignum: Tipo de variable, precisión arbitraria es un método que permite la representación, en un programa informático, de números ya sean enteros o racionales con tantos dígitos de precisión como cuanto sea deseado y además posibilita la realización de operaciones aritméticas sobre dichos números.

Char: Tipo de variable, unidad direccionable de la máquina más pequeña que puede contener un juego de caracteres básico. Es un tipo entero.

ANEXO A – CÓDIGO FUENTE DEL MPEM

MPEM - Aplicación Conductor

```c
/*
 * File:   main.c
 * Author: Miguel Carpio
 *
 * Created on August 21, 2016, 11:08 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include <openssl/bn.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/rand.h>

#include "paillier.h"

#include<sys/socket.h>
#include<arpa/inet.h> //inet_addr
#include<unistd.h>    //write
#include<pthread.h> //for threading , link with lpthread

#define KEY_LEN 512

/* Globals */
BN_CTX *ctx;
/**/

typedef struct oferta_s {
    char n[310];
    char n2[310];
    char g[310];
    char precio_s[310];
    char conductor[11];
    char fecha[9];
    char ciudad_ori[4];
    char ciudad_dest[4];
    int plazas;
    BIGNUM *precio;
} oferta;

typedef struct demanda_s {
    char pasajero[11];
    char contacto[11];
    char limite_precio[310];
    BIGNUM *limite;
} demanda;

paillierKeys K = {0};

int main(int argc, char** argv) {

    oferta oferta_c;
    demanda demanda_c;
```

```c
    int socket_desc[10];
    struct sockaddr_in server;
    int read_size;

    srand(time(NULL));
    ctx = BN_CTX_new();

    BN_CTX_start(ctx);
    int key_len = KEY_LEN;

    long long precio_plano = 0;

    oferta_c.precio = BN_CTX_get(ctx);
    demanda_c.limite = BN_CTX_get(ctx);

    BIGNUM *inv_map;
    inv_map = BN_CTX_get(ctx);

    BIGNUM *n;
    BIGNUM *n2;
    BIGNUM *g;
    n = NULL;
    n2 = NULL;
    g = NULL;

    int option = 0;

    while ( option < 1 || option > 2 ) {
    printf("\nBienvenido al MPEM - App Conductor\n\n");
    printf("Ingrese 1 para ejecutar la aplicacion en Caso Conductor o 2 en Caso Pasajero: \n");
    scanf("%d", &option);
    //printf("Numero ingresado: %d \n", option);
    }
```

```
92      if ( option == 1) {
93          int len = 2;
94          int cont = 0;
95          int i = 0;
96          FILE *fp;
97          char buff[16];
98          char ip_clients[3][16];
99
100
101         fp = fopen("pasajeros.txt", "r");
102         printf("\n*****************************************************************\n");
103         printf("MPEM - Caso Conductor - App Conductor\n" );
104         printf("*****************************************************************\n");
105         printf("\n*****************************************************************\n");
106         printf("Pasajeros registrados en el MPEM\n\n" );
107
108         while( len > 1 ) {
109             fgets(buff, 16, (FILE*)fp);
110             printf("%s", buff );
111             len = strlen(buff);
112             strcpy(ip_clients[cont],buff);
113             buff[0] = '\0';
114             cont = cont+1;
115         }
116
117         fclose(fp);
118
119         cont = cont - 1;
120         printf("*****************************************************************\n");
121
122         printf("\n*****************************************************************\n");
123         printf("Informacion del Conductor \n\n");
124         printf("Ingrese su numero de identificacion: \n");
125         scanf("%s", &oferta_c.conductor);
126         printf("Ingrese ciudad origen: \n");
127         scanf("%s", &oferta_c.ciudad_ori);
128         printf("Ingrese ciudad destino: \n");
129         scanf("%s", &oferta_c.ciudad_dest);
130         printf("Ingrese fecha del viaje: \n");
131         scanf("%s", &oferta_c.fecha);
132         printf("Ingrese el precio del viaje: \n");
133         scanf("%lli", &precio_plano);
134         printf("Ingrese el numero de asientos disponibles: \n");
135         scanf("%d", &oferta_c.plazas);
136
137         if (generateRandomKeys(&K, &key_len, ctx) != 0)
138             return -1;
139
140         if (encryptll(oferta_c.precio, precio_plano, &K.pub, ctx) != 0)
141             return -1;
142
143         char *result_str_p = BN_bn2dec(oferta_c.precio);  // convert the BIGNUM back to string
144         printf("Precio cifrado con Paillier:\n%s\n", result_str_p);
145         strcpy(oferta_c.precio_s,result_str_p);
146         printf("*****************************************************************\n\n");
147
148         char *result_str_g = BN_bn2dec(K.pub.g);  // convert the BIGNUM back to string
```

```
149         //printf("Clave publica g:\n%s\n", result_str_g);
150         strcpy(oferta_c.g,result_str_g);
151
152
153         char *result_str_n = BN_bn2dec(K.pub.n);  // convert the BIGNUM back to string
154         //printf("Clave publica n:\n%s\n", result_str_n);
155         strcpy(oferta_c.n,result_str_n);
156
157
158         char *result_str_n2 = BN_bn2dec(K.pub.n2);  // convert the BIGNUM back to string
159         //printf("Clave publica n2:\n%s\n", result_str_n2);
160         strcpy(oferta_c.n2,result_str_n2);
161
162         for(i = 0; i < cont; i++)
163         {
164             //printf("cont: %d, i: %d\n",cont,i);
165             //Create socket
166             socket_desc[i] = socket(AF_INET , SOCK_STREAM , 0);
167             if (socket_desc[i] == -1)
168             {
169                 printf("Could not create socket");
170             }
171             puts("Socket created");
172
173             //Prepare the sockaddr_in structure
174             printf("%s", ip_clients[i]);
175             server.sin_addr.s_addr = inet_addr(ip_clients[i]);
176             server.sin_family = AF_INET;
177             server.sin_port = htons( 8888 );
178
```

```
            //Connect to remote server
            if (connect(socket_desc[i] , (struct sockaddr *)&server , sizeof(server)) < 0)
            {
                perror("connect failed. Error");
                return 1;
            }

            puts("Connected");

            //Send some messages to the client
            if( send(socket_desc[i] , (void *)&oferta_c , sizeof(oferta_c) , 0) < 0)
            {
                puts("Send failed");
                return 1;
            }
            puts("Sent offer\n");

        }

        demanda demanda_tmp;
        int k = 0;

        while( (read_size = recv(socket_desc[k] , (void *)&demanda_tmp , sizeof(demanda_tmp), 0)) > 0 )
        {
            //Proccess
            demanda demanda_p = demanda_tmp;
            BIGNUM *result_sub;
            result_sub = NULL;
            long long limite_precio;
            printf("\n*****************************************************************\n");
            puts("Demanda del cliente \n");
            printf("Identificacion del pasajero:\n%s\n", demanda_p.pasajero);
            printf("Numero de telefono del pasajero:\n%s\n", demanda_p.contacto);
            printf("Limite menos precio cifrado con Paillier:\n%s\n", demanda_p.limite_precio);

            BN_dec2bn(&result_sub, demanda_p.limite_precio);

            if (decrypt(inv_map, result_sub, &K.priv, ctx) != 0)
                return -1;

            if (inv_mapping(&limite_precio,inv_map,K.pub.n,ctx))
                return -1;

            puts("\nProceso de la informacion cifrada \n");
            if (limite_precio < 0)
            printf("El precio limite del pasajero es menor al precio del viaje, precio limite del pasajero no aceptado\n");
                else
                {
                    printf("Precio limite del pasajero menos Precio del viaje del conductor positivo, Oferta aceptada, Resta computada c
                    oferta_c.plazas=oferta_c.plazas-1;
                    if (oferta_c.plazas > 0)
                        printf("Numero de asientos disponibles:\n%d\n", oferta_c.plazas);
                    else{
                        printf("*****************************************************************\n\n");
                        printf("Las plazas se han llenado, no existe asientos disponibles, programa MPEM finalizado, Gracias...\n\n");
                        return (EXIT_SUCCESS);
                    }
```

```
                }
            printf("*****************************************************************\n");
            if (k < i)
                k = k+1;
            else
                k = 0;
        }

        if(read_size == 0)
        {
            puts("Client disconnected");
            fflush(stdout);
        }
        else if(read_size == -1)
        {
            puts("\nTodos los clientes han enviado sus precios limites por el viaje, Fin del programa MPEM\n");
            return (EXIT_SUCCESS);
        }
    }
    else{
        oferta oferta_p;
        BIGNUM *n;
        BIGNUM *n2;
        BIGNUM *g;
        BIGNUM *precio;
        n = NULL;
        n2 = NULL;
        g = NULL;
        precio = NULL;
        long long limite_plano = 0;

        BIGNUM *result_sum;
        BIGNUM *result_sub;
```

```
269         result_sum = BN_CTX_get(ctx);
270         result_sub = BN_CTX_get(ctx);
271         struct _pubKey pub;
272 
273         demanda demanda_c;
274         demanda_c.limite = BN_CTX_get(ctx);
275 
276         printf("\n*************************************************************\n");
277         printf("MPEM - App Conductor - Caso Pasajero\n" );
278         printf("*************************************************************\n");
279         printf("\n*************************************************************\n");
280         printf("Informacion del Conductor\n\n");
281         printf("Ingrese su numero de identificacion: \n");
282         scanf("%s", &demanda_c.pasajero);
283         printf("Ingrese su numero de telefono: \n");
284         scanf("%s", &demanda_c.contacto);
285         printf("*************************************************************\n\n");
286 
287         int sock, c, client_sock;
288         struct sockaddr_in server, client;
289 
290         //Create socket
291         sock = socket(AF_INET , SOCK_STREAM , 0);
292         if (sock == -1)
293         {
294             printf("Could not create socket");
295         }
296         puts("Socket created");
297 
298         //Prepare the sockaddr_in structure
299         server.sin_family = AF_INET;
300         server.sin_addr.s_addr = INADDR_ANY;
301         server.sin_port = htons( 8888 );
302 
303         //Bind
304         if( bind(sock,(struct sockaddr *)&server , sizeof(server)) < 0)
305         {
306             //print the error message
307             perror("bind failed. Error");
308             return 1;
309         }
310         puts("bind done");
311 
312         //Listen
313         listen(sock , 3);
314 
315         //Accept and incoming connection
316         puts("Waiting for incoming connections...");
317         c = sizeof(struct sockaddr_in);
318 
319         client_sock = accept(sock, (struct sockaddr *)&client, (socklen_t*)&c);
320         if (client_sock < 0)
321         {
322             perror("accept failed");
323             return 1;
324         }
325         puts("Connection accepted");
```

```
326 
327         //Receive a reply from the server
328         if( recv(client_sock , (void *)&oferta_p, sizeof(oferta_p) , 0) < 0)
329         {
330             puts("recv failed");
331             return 1;
332         }
333         puts("Offer received\n");
334 
335         printf("*************************************************************\n");
336         puts("Demanda del pasajero\n");
337         printf("Identificacion del conductor:\n%s\n", oferta_p.conductor);
338         printf("Fecha del viaje:\n%s\n", oferta_p.fecha);
339         printf("Ciudad Origen:\n%s\n", oferta_p.ciudad_ori);
340         printf("Ciudad Destino:\n%s\n", oferta_p.ciudad_dest);
341         printf("Numero de acientos solicitados:\n%d\n", oferta_p.plazas);
342         printf("Precio limite cifrado con Paillier:\n%s\n", oferta_p.precio_s);
343         printf("*************************************************************\n");
344 
345         int option2 = 0;
346 
347         while ( option2 < 1 || option2 > 2 ) {
348         printf("\nDesea aceptar la demanda del viaje\n\n");
349         printf("Ingrese 1 para aceptar o 2 para rechazar: \n");
350         scanf("%d", &option2);
351         }
352 
353         if ( option2 == 2 ){
354                 printf("No se acepto la demanda de viaje del pasajero. Fin MPEM\n\n");
355                 return (EXIT_SUCCESS);
356         }
357 
358         BN_dec2bn(&precio, oferta_p.precio_s);
```

```
359             BN_dec2bn(&n, oferta_p.n);
360             BN_dec2bn(&n2, oferta_p.n2);
361             BN_dec2bn(&g, oferta_p.g);
362 
363             pub.g = BN_dup(g);
364             pub.n = BN_dup(n);
365             pub.n2 = BN_dup(n2);
366 
367             printf("\n************************************************************\n");
368             printf("Oferta del conductor\n\n");
369             printf("Ingrese precio a cobrar por el viaje: \n");
370             scanf("%lli", &limite_plano);
371 
372             if (encryptll(demanda_c.limite, limite_plano, &pub, ctx) != 0)
373                 return 1;
374 
375             char *result_str_lim = BN_bn2dec(demanda_c.limite);  // convert the BIGNUM back to string
376             printf("Precio del viaje cifrado con Paillier:\n%s\n", result_str_lim);
377 
378             if (add(result_sub, precio, demanda_c.limite, pub.n2, ctx) != 0)
379                 return 1;
380 
381             char *result_str_sub = BN_bn2dec(result_sub);  // convert the BIGNUM back to string
382             printf("Resta, precio del viaje del conductor menos limite a pagar por el pasajero, operacion computada con el criptosistema homom
383             strcpy(demanda_c.limite_precio,result_str_sub);
384             printf("************************************************************\n");
385 
386             //Send some data
387             if( send(client_sock , (void *)&demanda_c , sizeof(demanda_c) , 0) < 0)
388             {
389                 puts("Send failed");
390                 return 1;
391             }
392 
393             printf("\nPrecio del viaje menos limite a pagar por el pasajero fue enviado, si el precio del viaje es menor al limite, el pasajer
394 
395             close(sock);
396         }
397 
398         BN_CTX_end(ctx);
399         BN_CTX_free(ctx);
400 
401         return (EXIT_SUCCESS);
402     }
403 
```

MPEM - Aplicación Pasajero

```c
/*
 * File:   main.c
 * Author: Miguel carpio
 *
 * Created on 10 de septiembre de 2016, 22:22
 */

#include<stdio.h> //printf
#include<string.h>    //strlen
#include<sys/socket.h>    //socket
#include<arpa/inet.h> //inet_addr

#include <openssl/bn.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/rand.h>

#include "paillier.h"

#define KEY_LEN 512

/* Globals */
BN_CTX *ctx;
/**/

typedef struct oferta_s {
    char n[310];
    char n2[310];
    char g[310];
    char precio_s[310];
    char conductor[11];
    char fecha[9];
    char ciudad_ori[4];
    char ciudad_dest[4];
    int plazas;
    BIGNUM *precio;
} oferta;


typedef struct demanda_s {
    char pasajero[11];
    char contacto[11];
    char limite_precio[310];
    BIGNUM *limite;
} demanda;


int main(int argc, char** argv) {

    srand(time(NULL));
    ctx = BN_CTX_new();
    BN_CTX_start(ctx);

    BIGNUM *result_sum;
    BIGNUM *result_sub;
    result_sum = BN_CTX_get(ctx);
    result_sub = BN_CTX_get(ctx);
```

```c
    struct _pubKey pub;

    demanda demanda_p;
    demanda_p.limite = BN_CTX_get(ctx);

    oferta oferta_c;
    oferta oferta_p;
    long long limite_plano = 0;


    BIGNUM *n;
    BIGNUM *n2;
    BIGNUM *g;
    BIGNUM *precio;
    n = NULL;
    n2 = NULL;
    g = NULL;
    precio = NULL;

    int option = 0;

    while ( option < 1 || option > 2 ) {
    printf("\nBienvenido al MPEM - App Pasajero\n\n");
```

```
81      printf("Ingrese 1 para ejecutar la aplicacion en Caso Conductor o 2 en Caso Pasajero: \n");
82      scanf("%d", &option);
83      //printf("Numero ingresado: %d \n", option);
84      }
85
86      if ( option == 1) {
87
88          BIGNUM *map;
89          map = BN_CTX_get(ctx);
90
91          printf("\n*************************************************************\n");
92          printf("MPEM - Caso Conductor - App Pasajero\n");
93  printf("*************************************************************\n");
94
95          printf("\n*************************************************************\n");
96          printf("Informacion del Pasajero\n\n");
97          printf("Ingrese su numero de identificacion: \n");
98          scanf("%s", &demanda_p.pasajero);
99          printf("Ingrese su numero de telefono: \n");
100         scanf("%s", &demanda_p.contacto);
101         printf("*************************************************************\n\n");
102
103         int sock, c, client_sock;
104         struct sockaddr_in server, client;
105
106         //Create socket
107         sock = socket(AF_INET , SOCK_STREAM , 0);
108         if (sock == -1)
109         {
110             printf("Could not create socket");
111         }
112         puts("Socket created");
113
114         //Prepare the sockaddr_in structure
115         server.sin_family = AF_INET;
116         server.sin_addr.s_addr = INADDR_ANY;
117         server.sin_port = htons( 8888 );
118
119         //Bind
120         if( bind(sock,(struct sockaddr *)&server , sizeof(server)) < 0)
121         {
122             //print the error message
123             perror("bind failed. Error");
124             return 1;
125         }
126         puts("bind done");
127
128         //Listen
129         listen(sock , 3);
130
131         //Accept and incoming connection
132         puts("Waiting for incoming connections...");
133         c = sizeof(struct sockaddr_in);
134
135         client_sock = accept(sock, (struct sockaddr *)&client, (socklen_t*)&c);
136         if (client_sock < 0)
137         {
```

```
137         {
138             perror("accept failed");
139             return 1;
140         }
141         puts("Connection accepted");
142
143         //Receive a reply from the server
144         if( recv(client_sock , (void *)&oferta_c, sizeof(oferta_c) , 0) < 0)
145         {
146             puts("recv failed");
147             return 1;
148         }
149         puts("Offer received\n");
150
151         printf("*************************************************************\n");
152         puts("Oferta del conductor\n");
153         printf("Identificacion del conductor:\n%s\n", oferta_c.conductor);
154         printf("Fecha del viaje:\n%s\n", oferta_c.fecha);
155         printf("Ciudad Origen:\n%s\n", oferta_c.ciudad_ori);
156         printf("Ciudad Destino:\n%s\n", oferta_c.ciudad_dest);
157         printf("Numero de acientos disponibles:\n%d\n", oferta_c.plazas);
158         printf("Precio cifrado con Paillier:\n%s\n", oferta_c.precio_s);
159         printf("*************************************************************\n");
160
161         int option2 = 0;
162
163         while ( option2 < 1 || option2 > 2 ) {
164         printf("\nDesea aceptar la oferta de viaje\n\n");
```

```
165         printf("Ingrese 1 para aceptar o 2 para rechazar: \n");
166         scanf("%d", &option2);
167         }
168
169         if ( option2 == 2 ) {
170                 printf("No se acepto la oferta de viaje del conductor. Fin MPEM\n\n");
171                 return (EXIT_SUCCESS);
172         }
173
174         BN_dec2bn(&precio, oferta_c.precio_s);
175         BN_dec2bn(&n, oferta_c.n);
176         BN_dec2bn(&n2, oferta_c.n2);
177         BN_dec2bn(&g, oferta_c.g);
178
179         pub.g = BN_dup(g);
180         pub.n = BN_dup(n);
181         pub.n2 = BN_dup(n2);
182
183         printf("\n*****************************************************************\n");
184         printf("Demanda del cliente\n\n");
185         printf("Ingrese precio limite a pagar: \n");
186         scanf("%lli", &limite_plano);
187
188         if (mapping(map,limite_plano,pub.n,ctx))
189             return -1;
190
191         if (encrypt(demanda_p.limite, map, &pub, ctx) != 0)
192             return -1;
193
194         char *result_str_lim = BN_bn2dec(demanda_p.limite);  // convert the BIGNUM back to string
195         printf("Limite cifrado con Paillier:\n%s\n", result_str_lim);
196
197         if (add(result_sub, demanda_p.limite, precio, pub.n2, ctx) != 0)
198             return 1;
199
200         char *result_str_sub = BN_bn2dec(result_sub);  // convert the BIGNUM back to string
201         printf("Resta, precio limite del pasajero menos precio del viaje del conductor, operacion computada con el criptosistema
202         strcpy(demanda_p.limite_precio,result_str_sub);
203         printf("*****************************************************************\n");
204
205         //Send some data
206         if( send(client_sock , (void *)&demanda_p , sizeof(demanda_p) , 0) < 0)
207         {
208             puts("Send failed");
209             return 1;
210         }
211
212         printf("\nLimite a pagar por el pasajero menos precio del viaje del conductor enviado al conductor, si el limite se ajus
213
214         close(sock);
215     }
216     else {
217         long long precio_plano = 0;
218         int socket_desc[10];
219         struct sockaddr_in server;
220         int read_size;
221
```

```
222         int key_len = KEY_LEN;
223         paillierKeys K = {0};
224
225         BIGNUM *map;
226         map = BN_CTX_get(ctx);
227
228         BIGNUM *inv_map;
229         inv_map = BN_CTX_get(ctx);
230
231         oferta_p.precio = BN_CTX_get(ctx);
232         BIGNUM *n;
233         BIGNUM *n2;
234         BIGNUM *g;
235         n = NULL;
236         n2 = NULL;
237         g = NULL;
238
239         int len = 2;
240         int cont = 0;
241         int i = 0;
242         FILE *fp;
243         char buff[16];
244         char ip_clients[3][16];
```

```
245
246         fp = fopen("conductores.txt", "r");
247         printf("\n\n*************************************************************\n");
248         printf("MPEM - App Pasajero - Caso Pasajero\n" );
249         printf("*************************************************************\n");
250         printf("\n*************************************************************\n");
251         printf("Conductores registrados en el MPEM\n\n" );
252
253         while( len > 1 ) {
254             fgets(buff, 16, (FILE*)fp);
255             printf("%s", buff );
256             len = strlen(buff);
257             strcpy(ip_clients[cont],buff);
258             buff[0] = '\0';
259             cont = cont+1;
260         }
261         printf("*************************************************************\n");
262         printf("\n*************************************************************\n");
263         printf("Informacion del Pasajero\n\n");
264         printf("Ingrese su numero de identificacion: \n");
265         scanf("%s", &oferta_p.conductor);
266         printf("Ingrese ciudad origen: \n");
267         scanf("%s", &oferta_p.ciudad_ori);
268         printf("Ingrese ciudad destino: \n");
269         scanf("%s", &oferta_p.ciudad_dest);
270         printf("Ingrese fecha del viaje: \n");
271         scanf("%s", &oferta_p.fecha);
272         printf("Ingrese el limite de precio a pagar por del viaje: \n");
273         scanf("%lli", &precio_plano);
274         printf("Ingrese el numero de asientos que solicita: \n");
275         scanf("%d", &oferta_p.plazas);
276
277         if (generateRandomKeys(&K, &key_len, ctx) != 0)
278         return -1;
279
280         if (mapping(map,precio_plano,K.pub.n,ctx))
281             return -1;
282
283         if (encrypt(oferta_p.precio, map, &K.pub, ctx) != 0)
284             return -1;
285
286         char *result_str_p = BN_bn2dec(oferta_p.precio);  // convert the BIGNUM back to string
287         printf("Precio limite cifrado con Paillier:\n%s\n", result_str_p);
288         strcpy(oferta_p.precio_s,result_str_p);
289         printf("*************************************************************\n\n");
290
291         char *result_str_g = BN_bn2dec(K.pub.g);  // convert the BIGNUM back to string
292         //printf("Clave publica g:\n%s\n", result_str_g);
293         strcpy(oferta_p.g,result_str_g);
294
295
296         char *result_str_n = BN_bn2dec(K.pub.n);  // convert the BIGNUM back to string
297         //printf("Clave publica n:\n%s\n", result_str_n);
298         strcpy(oferta_p.n,result_str_n);
299
300
301         char *result_str_n2 = BN_bn2dec(K.pub.n2);  // convert the BIGNUM back to string
```

```
302         //printf("Clave publica n2:\n%s\n", result_str_n2);
303         strcpy(oferta_p.n2,result_str_n2);
304
305         cont = cont - 1;
306
307         for(i = 0; i < cont; i++)
308         {
309             //printf("cont: %d, i: %d\n",cont,i);
310             //Create socket
311             socket_desc[i] = socket(AF_INET , SOCK_STREAM , 0);
312             if (socket_desc[i] == -1)
313             {
314                 printf("Could not create socket");
315             }
316             puts("Socket created");
317
318             //Prepare the sockaddr_in structure
319             printf("%s", ip_clients[i]);
320             server.sin_addr.s_addr = inet_addr(ip_clients[i]);
321             server.sin_family = AF_INET;
322             server.sin_port = htons( 8888 );
323
324             //Connect to remote server
325             if (connect(socket_desc[i] , (struct sockaddr *)&server , sizeof(server)) < 0)
326             {
327                 perror("connect failed. Error");
328                 return 1;
329             }
330
```

```
            puts("Connected");

            //Send some messages to the client
            if( send(socket_desc[i] , (void *)&oferta_p , sizeof(oferta_p) , 0) < 0)
            {
                puts("Send failed");
                return 1;
            }
            puts("Sent offer");
        }

        demanda demanda_tmp;
        long long ofertas[3] = {0,0,0};
        char conductores[3][11];
        char contacto_condt[3][11];
        int id_conductor;
        int k = 0;
        int cont2 = 0;

        while( (read_size = recv(socket_desc[k] , (void *)&demanda_tmp , sizeof(demanda_tmp), 0)) > 0 )
        {
            //Proccess
            demanda demanda_c = demanda_tmp;
            BIGNUM *result_sub;
            result_sub = NULL;
            long long limite_precio;

            printf("\n*************************************************************\n");
            puts("Oferta del conductor \n");
            printf("Identificacion del conductor:\n%s\n", demanda_c.pasajero);
            printf("Numero de telefono del conductor:\n%s\n", demanda_c.contacto);
            printf("Limite menos precio cifrado con Paillier:\n%s\n", demanda_c.limite_precio);

            BN_dec2bn(&result_sub, demanda_c.limite_precio);

            if (decrypt(inv_map, result_sub, &K.priv, ctx) != 0)
                return -1;

            if (inv_mapping(&limite_precio,inv_map,K.pub.n,ctx))
                return -1;

            puts("\nProceso de la informacion cifrada \n");
            if (limite_precio < 0)
                printf("El precio del viaje es mayor al valor limite, operacion computada con el criptosistema homomorfico de Pa
            else{
                printf("El precio del viaje es menor/igual al valor limite, operacion computada con el criptosistema homomorfico
                ofertas[cont2]=limite_precio;
                strcpy(conductores[cont2],demanda_c.pasajero);
                strcpy(contacto_condt[cont2],demanda_c.contacto);
                cont2 = cont2 + 1;
            }
            if (k < i)
                k = k+1;
            else
                k = 0;

            printf("*************************************************************\n\n");
```

```
        }

        if(read_size == 0)
        {
            puts("Client disconnected");
            fflush(stdout);
        }
        else if(read_size == -1)
        {
            long long max;
            int c, location = 1;
            max = ofertas[0];

            for ( c = 1 ; c < cont2 ; c++ )
            {
                if ( ofertas[c] > max )
                {
                   max = ofertas[c];
                   location = c+1;
                }
            }

            for ( c = 0 ; c < cont2 ; c++ )
            {
                if ( ofertas[c] == max )
                {
                    id_conductor = c;
                }
```

```
416             }
417
418             if (cont2 != 0)
419                 printf("El conductor que ofrece el valor mas bajo por el viaje es: %s, su numero telefonico es: %s, diferencia de
420             else
421                 printf("Todas las ofertas de los conductores superaron el precio limite a pagar, ofertas de conductores no acepta
422         }
423     }
424
425
426     BN_CTX_end(ctx);
427     BN_CTX_free(ctx);
428
429     return (EXIT_SUCCESS);
430 }
```

ANEXO B – CONFIGURACIÓN DE SIMULACIONES DEL MPEM SOBRE UNA RED OPORTUNISTA

Simulación 1 - MPEM_GYE_CASO_CONDUCTOR_PINPONG_EPIDEMIC_5_20

```
Scenario.name = MPEM_GYE_CASO_CONDUCTOR_PINPONG_EPIDEMIC_5_20
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type = SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250

Scenario.nrofHostGroups = 6

Group.movementModel = MapBasedMovement
Group.router = EpidemicRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 50

Group1.groupID = c
Group1.movementModel = MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 5

Group2.groupID = pp
Group2.movementModel = ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 5

Group3.groupID = pc
Group3.movementModel = MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 15

Group4.groupID = mr
Group4.movementModel = MapRouteMovement

Events.nrof = 1

Events1.hosts = 0,0
Events1.prefix = MH1_
Events1.time = 0,209
Events1.class = MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 5,25
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 5,25
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 209.0000

pingAppResponse.type = PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 = pingApp

Group2.nrofApplications = 1
Group2.application1 = pingAppResponse

Group3.nrofApplications = 1
Group3.application1 = pingAppResponse

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000, 90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 = data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 = data/wkt/wkt_sim_gye/metro_riodaule.wkt
```

```
Group4.routeFile =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
MapBasedMovement.mapFile3 =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
MapBasedMovement.mapFile4 =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt

Report.nrofReports = 3
Report.warmup = 0
Report.reportDir =
reports/caso_conductor/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 10
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 2 - MPEM_GYE_CASO_CONDUCTOR_PINGPONG_EPIDEMIC_20_80

```
Scenario.name =
MPEM_GYE_CASO_CONDUCTOR_PINGPONG_EPI
DEMIC_20_80
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type =
SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250

Scenario.nrofHostGroups = 6

Group.movementModel =
MapBasedMovement
Group.router = EpidemicRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300
```

```
Events.nrof = 1

Events1.hosts = 0,0
Events1.prefix = MH1_
Events1.time = 0,809
Events1.class =
MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 20,100
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 20,100
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 809.0000

pingAppResponse.type =
PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true
```

```
Group.nrofHosts = 125

Group1.groupID = c
Group1.movementModel =
MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 20

Group2.groupID = pp
Group2.movementModel =
ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 20

Group3.groupID = pc
Group3.movementModel =
MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 60

Group4.groupID = mr
Group4.movementModel =
MapRouteMovement
Group4.routeFile =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
Group1.nrofApplications = 1
Group1.application1 = pingApp

Group2.nrofApplications = 1
Group2.application1 =
pingAppResponse

Group3.nrofApplications = 1
Group3.application1 =
pingAppResponse

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000,
90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 =
data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
MapBasedMovement.mapFile3 =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
MapBasedMovement.mapFile4 =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt

Report.nrofReports = 3
Report.warmup = 0
Report.reportDir =
reports/caso_conductor/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 10
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 3 - MPEM_GYE_CASO_CONDUCTOR_PINPONG_SAW_5_20

```
Scenario.name = MPEM_GYE_CASO_CONDUCTOR_PINPONG_SAW_5_20
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type = SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250

Scenario.nrofHostGroups = 6

Group.movementModel = MapBasedMovement
Group.router = SprayAndWaitRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 50

Group1.groupID = c
Group1.movementModel = MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 5

Group2.groupID = pp
Group2.movementModel = ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 5

Group3.groupID = pc
Group3.movementModel = MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 15

Group4.groupID = mr
Group4.movementModel = MapRouteMovement
Group4.routeFile = data/wkt/wkt_sim_gye/metro_riodaule.wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel = MapRouteMovement
Events.nrof = 1

Events1.hosts = 0,0
Events1.prefix = MH1_
Events1.time = 0,209
Events1.class = MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 5,25
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 5,25
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 209.0000

pingAppResponse.type = PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 = pingApp

Group2.nrofApplications = 1
Group2.application1 = pingAppResponse

Group3.nrofApplications = 1
Group3.application1 = pingAppResponse

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000, 90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 = data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 = data/wkt/wkt_sim_gye/metro_riodaule.wkt
MapBasedMovement.mapFile3 = data/wkt/wkt_sim_gye/metro_25julio.wkt
MapBasedMovement.mapFile4 = data/wkt/wkt_sim_gye/buses_interprovinciales.wkt

Report.nrofReports = 3
Report.warmup = 0
```

```
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
Report.reportDir =
reports/caso_conductor/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter


SprayAndWaitRouter.nrofCopies = 10
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 4 - MPEM_GYE_CASO_CONDUCTOR_PINGPONG_SAW_20_80

```
Scenario.name =
MPEM_GYE_CASO_CONDUCTOR_PINGPONG_SAW
_20_80
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type =
SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250


Scenario.nrofHostGroups = 6

Group.movementModel =
MapBasedMovement
Group.router = SprayAndWaitRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 125

Group1.groupID = c
Group1.movementModel =
MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 20
```

```
Events.nrof = 1

Events1.hosts = 0,0
Events1.prefix = MH1_
Events1.time = 0,809
Events1.class =
MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 20,100
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 20,100
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 809.0000


pingAppResponse.type =
PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 = pingApp

Group2.nrofApplications = 1
Group2.application1 =
pingAppResponse
```

```
Group2.groupID = pp
Group2.movementModel =
ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 20

Group3.groupID = pc
Group3.movementModel =
MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 60

Group4.groupID = mr
Group4.movementModel =
MapRouteMovement
Group4.routeFile =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
Group3.nrofApplications = 1
Group3.application1 =
pingAppResponse


MovementModel.rngSeed = 1
MovementModel.worldSize = 90000,
90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 =
data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
MapBasedMovement.mapFile3 =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
MapBasedMovement.mapFile4 =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt

Report.nrofReports = 3
Report.warmup = 0
Report.reportDir =
reports/caso_conductor/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 10
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 5 - MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_5_20

```
Scenario.name = MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_5_20
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type = SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250

Scenario.nrofHostGroups = 6

Group.movementModel = MapBasedMovement
Group.router = EpidemicRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 50

Group1.groupID = c
Group1.movementModel = MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 5

Group2.groupID = pp
Group2.movementModel = ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 5

Group3.groupID = pc
Group3.movementModel = MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 15

Group4.groupID = mr
Group4.movementModel = MapRouteMovement
Group4.routeFile = data/wkt/wkt_sim_gye/metro_riodaule.wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
```

```
Events.nrof = 1

Events1.hosts = 5,5
Events1.prefix = MH5_
Events1.time = 0,59
Events1.class = MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 0,5
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 0,5
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 59.0000

pingAppResponse.type = PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 = pingAppResponse

Group2.nrofApplications = 1
Group2.application1 = pingApp

Group3.nrofApplications = 1
Group3.application1 = pingApp

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000, 90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 = data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 = data/wkt/wkt_sim_gye/metro_riodaule.wkt
MapBasedMovement.mapFile3 = data/wkt/wkt_sim_gye/metro_25julio.wkt
MapBasedMovement.mapFile4 = data/wkt/wkt_sim_gye/buses_interprovinciales.wkt

Report.nrofReports = 3
Report.warmup = 0
```

```
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
Report.reportDir =
reports/caso_pasajero/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 6
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 6 - MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PINGPONG_20_80

```
Scenario.name =
MPEM_GYE_CASO_PASAJERO_EPIDEMIC_PING
PONG_20_80
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type =
SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250

Scenario.nrofHostGroups = 6

Group.movementModel =
MapBasedMovement
Group.router = EpidemicRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 125

Group1.groupID = c
Group1.movementModel =
MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 20

Group2.groupID = pp
```

```
Events.nrof = 1

Events1.hosts = 20,20
Events1.prefix = MH1_
Events1.time = 0,209
Events1.class =
MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 0,20
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 0,20
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 209.0000

pingAppResponse.type =
PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 =
pingAppResponse

Group2.nrofApplications = 1
Group2.application1 = pingApp

Group3.nrofApplications = 1
```

```
Group2.movementModel =
ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 20

Group3.groupID = pc
Group3.movementModel =
MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 60

Group4.groupID = mr
Group4.movementModel =
MapRouteMovement
Group4.routeFile =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
Group3.application1 = pingApp

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000,
90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 =
data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
MapBasedMovement.mapFile3 =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
MapBasedMovement.mapFile4 =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt

Report.nrofReports = 3
Report.warmup = 0
Report.reportDir =
reports/caso_pasajero/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 6
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 7 - MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_5_20

```
Scenario.name =
MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_
5_20
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type =
SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250
```

```
Events.nrof = 1

Events1.hosts = 5,5
Events1.prefix = MH5_
Events1.time = 0,59
Events1.class =
MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 0,5
Events1.responseSize = 435
```

```
Scenario.nrofHostGroups = 6

Group.movementModel =
MapBasedMovement
Group.router = SprayAndWaitRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 50

Group1.groupID = c
Group1.movementModel =
MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 5

Group2.groupID = pp
Group2.movementModel =
ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 5

Group3.groupID = pc
Group3.movementModel =
MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 15

Group4.groupID = mr
Group4.movementModel =
MapRouteMovement
Group4.routeFile =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
```

```
pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 0,5
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 59.0000


pingAppResponse.type =
PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 =
pingAppResponse

Group2.nrofApplications = 1
Group2.application1 = pingApp

Group3.nrofApplications = 1
Group3.application1 = pingApp

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000,
90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 =
data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
MapBasedMovement.mapFile3 =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
MapBasedMovement.mapFile4 =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt

Report.nrofReports = 3
Report.warmup = 0
Report.reportDir =
reports/caso_pasajero/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 10
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true
```

```
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

```
GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

Simulación 8 - MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_20_80

```
Scenario.name =
MPEM_GYE_CASO_PASAJERO_SAW_PINGPONG_
20_80
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
Scenario.endTime = 43200

WiFi80211.type =
SimpleBroadcastInterface
WiFi80211.transmitSpeed = 18750k
WiFi80211.transmitRange = 250

Scenario.nrofHostGroups = 6

Group.movementModel =
MapBasedMovement
Group.router = SprayAndWaitRouter
Group.bufferSize = 1024M
Group.waitTime = 10, 30
Group.nrofInterfaces = 1
Group.interface1 = WiFi80211
Group.speed = 2.7, 13.9
Group.msgTtl = 300

Group.nrofHosts = 125

Group1.groupID = c
Group1.movementModel =
MapBasedMovement
Group1.speed = 2.7, 13.9
Group1.nrofHosts = 20

Group2.groupID = pp
Group2.movementModel =
ShortestPathMapBasedMovement
Group2.speed = 0.5, 1.5
Group2.nrofHosts = 20

Group3.groupID = pc
Group3.movementModel =
MapBasedMovement
Group3.speed = 2.7, 13.9
Group3.nrofHosts = 60

Group4.groupID = mr
Group4.movementModel =
MapRouteMovement
```

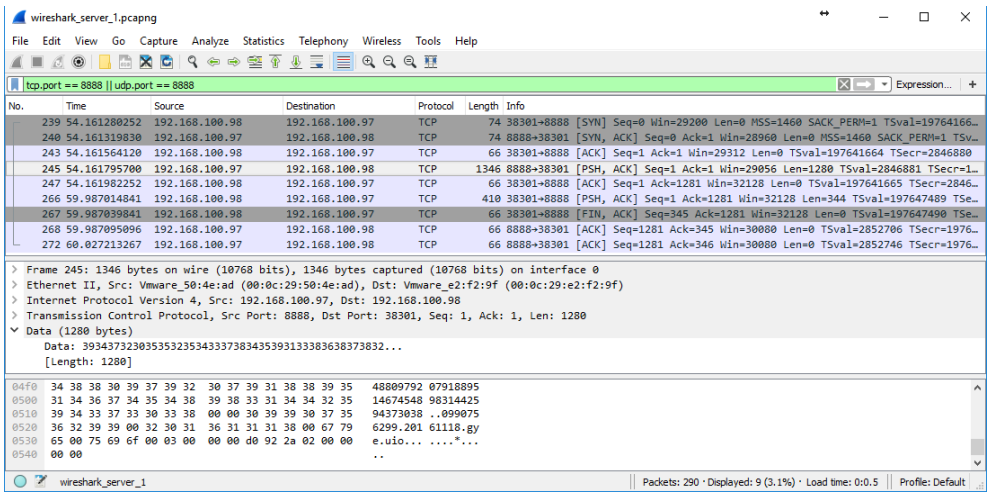
```
Events.nrof = 1

Events1.hosts = 20,20
Events1.prefix = MH1_
Events1.time = 0,209
Events1.class =
MessageEventGenerator
Events1.interval = 10
Events1.size = 1371
Events1.tohosts = 0,20
Events1.responseSize = 435

pingApp.type = PingApplication
pingApp.interval = 10
pingApp.destinationRange = 0,20
pingApp.pingSize = 1371
pingApp.pongSize = 435
pingApp.passive = false
pingApp.pingTime = 209.0000

pingAppResponse.type =
PingApplication
pingAppResponse.pongSize = 435
pingAppResponse.passive = true

Group1.nrofApplications = 1
Group1.application1 =
pingAppResponse

Group2.nrofApplications = 1
Group2.application1 = pingApp

Group3.nrofApplications = 1
Group3.application1 = pingApp

MovementModel.rngSeed = 1
MovementModel.worldSize = 90000,
90000
MovementModel.warmup = 1000

MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 =
data/wkt/wkt_sim_gye/calles_gye.wkt
MapBasedMovement.mapFile2 =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
```

```
Group4.routeFile =
data/wkt/wkt_sim_gye/metro_riodaule.
wkt
Group4.routeType = 2
Group4.speed = 5.4, 17.55
Group4.nrofHosts = 10

Group5.groupID = mj
Group5.movementModel =
MapRouteMovement
Group5.routeFile =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
Group5.routeType = 2
Group5.speed = 5.4, 17.55
Group5.nrofHosts = 10

Group6.groupID = b
Group6.movementModel =
MapRouteMovement
Group6.routeFile =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt
Group6.routeType = 2
Group6.speed = 8.1, 18.9
Group6.nrofHosts = 5
```

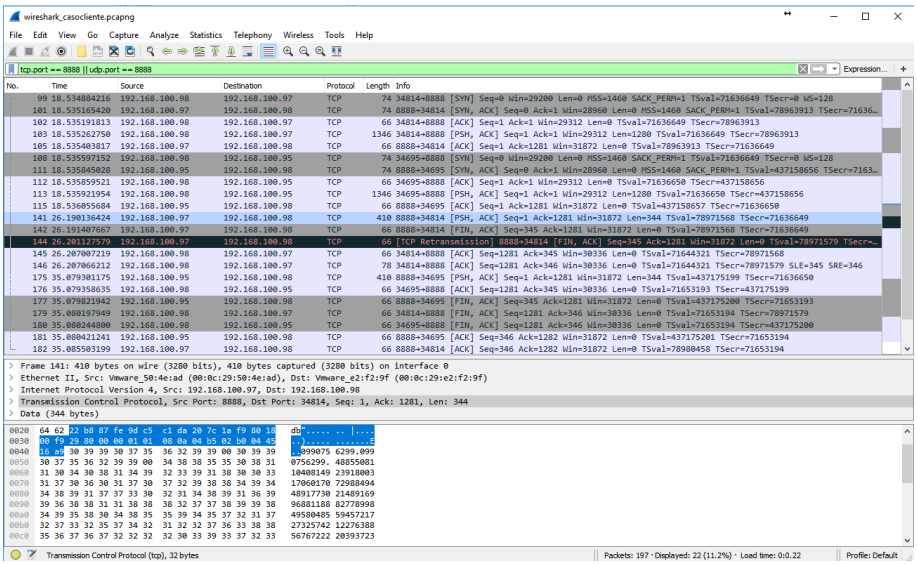
```
MapBasedMovement.mapFile3 =
data/wkt/wkt_sim_gye/metro_25julio.w
kt
MapBasedMovement.mapFile4 =
data/wkt/wkt_sim_gye/buses_interprov
inciales.wkt

Report.nrofReports = 3
Report.warmup = 0
Report.reportDir =
reports/caso_pasajero/simulaciones
Report.report1 = MessageStatsReport
Report.report2 =
MessageGraphvizReport
Report.report3 = PingAppReporter

SprayAndWaitRouter.nrofCopies = 10
SprayAndWaitRouter.binaryMode = true

Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder =
true

GUI.UnderlayImage.fileName =
data/images/guayaquil.png
GUI.UnderlayImage.offset = -8, 2
GUI.UnderlayImage.scale = 27.365
GUI.UnderlayImage.rotate = 0
GUI.EventLogPanel.nrofEvents = 100
```

ANEXO C – CAPTURA DE TRAFICO TCP ENTRE LAS APLICACIONES CONDUCTOR Y PASAJERO DEL MPEM

Caso Conductor, captura de tráfico del esquema de intercambio de mensajes

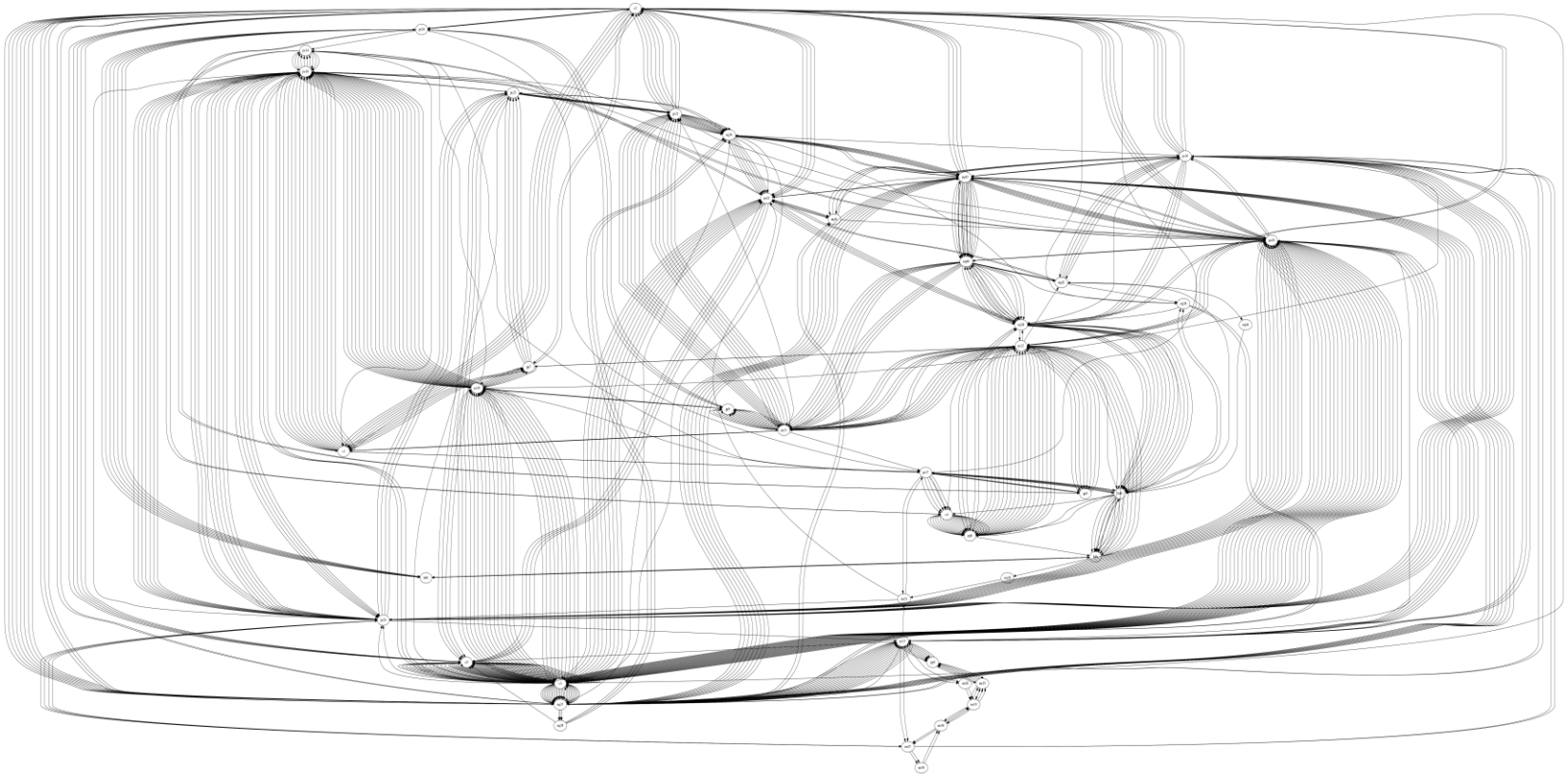


Caso Pasajero, captura de tráfico del esquema de intercambio de mensajes

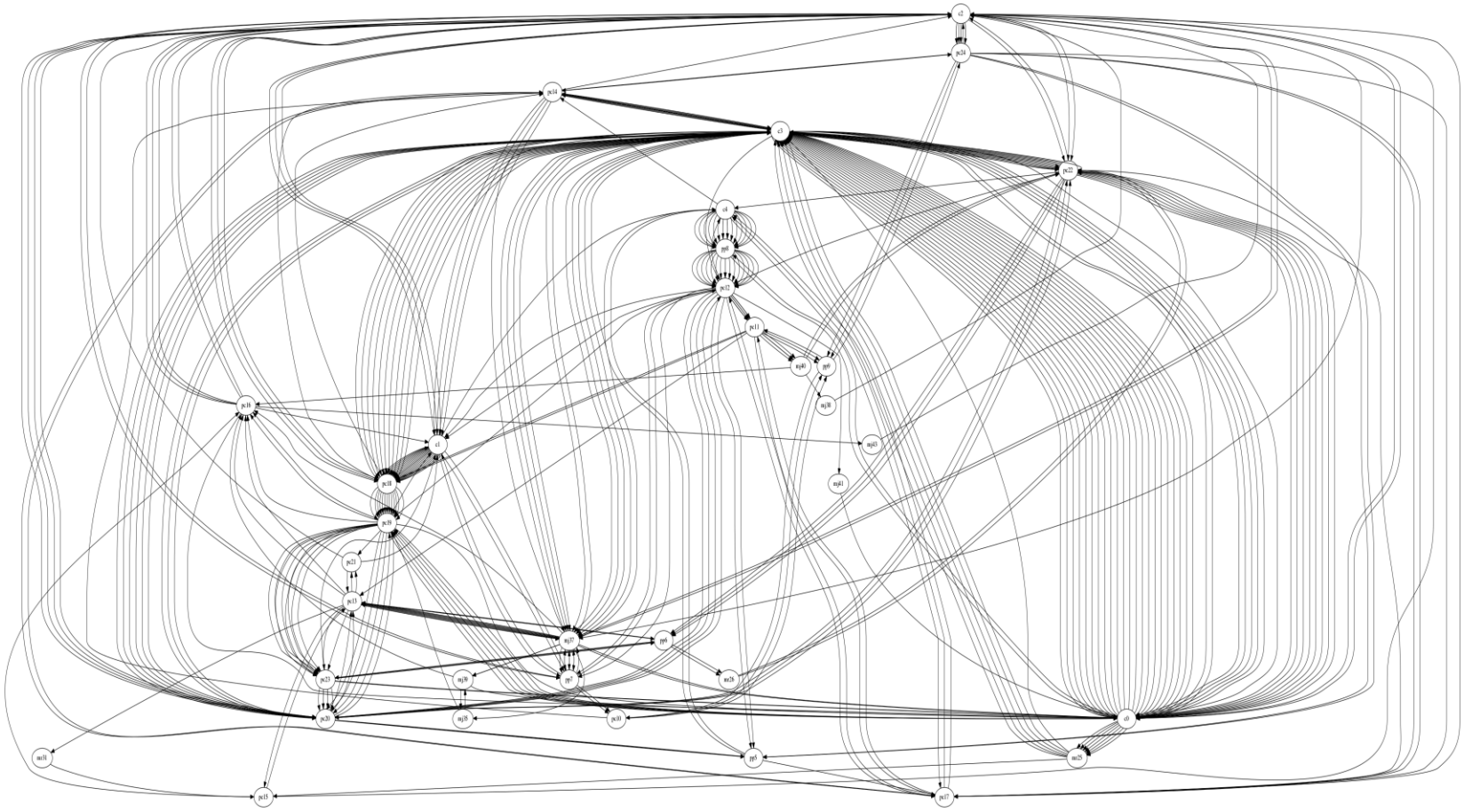


ANEXO D – GRAFOS DE LAS SIMULACIONES

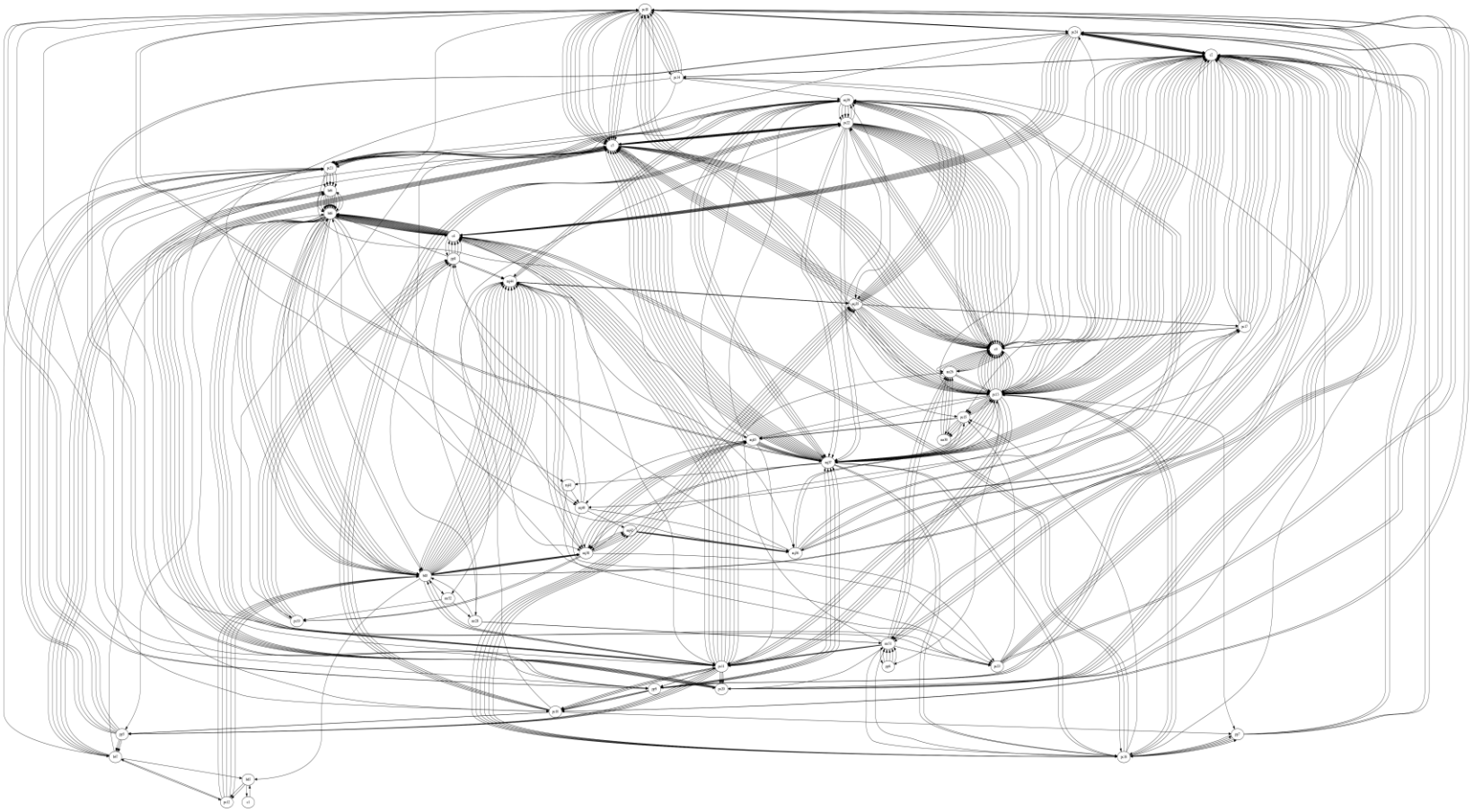
Simulación 1 Caso Conductor Epidemic 5:20



Simulación 3 Caso Conductor SprayAndWait 5:20



Simulación 5 Caso Pasajero Epidemic 5:20



Simulación 7 Caso Pasajero SprayAndWait 5:20