Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Doctorado en Ciencias en Ciencias de la Computación

Diseño de algoritmos para resolver el problema de distribución máxima y homogénea de mensajes

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Doctor en Ciencias

Presenta:

Héctor Zatarain Aceves

Ensenada, Baja California, México

2018

Tesis defendida por

Héctor Zatarain Aceves

y aprobada por el siguiente Comité

Dr. José Alberto Fernández Zepeda
Codirector de tesis

Dr. Carlos Alberto Brizuela Rodríguez
Codirector de tesis

Dr. José Antonio García Macías

Dr. Salvador Villarreal Reyes

Dr. Rolando Menchaca Méndez

Dr. Jesús Favela Vara
Coordinador del Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta Héctor Zatarain Aceves como requisito parcial para la obtención del grado de Doctor en Ciencias en Ciencias de la Computación.

Diseño de algoritmos para resolver el problema de distribución máxima y homogénea de mensajes

Resumen aprobado por:

Dr. José Alberto Fernández Zepeda	Dr. Carlos Alberto Brizuela Rodríguez
Codirector de tesis	Codirector de tesis

En este trabajo se propone el problema de Distribución Máxima y Homogénea de Mensajes (DMHM), el cual puede surgir en redes de comunicación tolerantes a fallas donde el destino de los mensajes no está presente en la red. Este tipo de comportamientos se pueden presentar en escenarios donde ocurren desastres naturales o conflictos sociales y donde una red de comunicación global no está disponible o se encuentra temporalmente fuera de servicio. En estos escenarios, las personas dentro del área afectada utilizan sus dispositivos móviles para comunicarse con las demás de forma oportunista. De tal forma, que cuando un dispositivo alcanza la cobertura de una red de comunicación global es capaz de enviar todos los mensajes en su memoria, entonces se dice que estos mensajes fueron entregados satisfactoriamente. En este trabajo se demuestra que a mayor uniformidad en la distribución de los mensajes en la red, mayor la probabilidad de entregas exitosas. Se modeló al DMHM como un problema computacional y se analizó la efectividad de algunos algoritmos de enrutamiento para redes oportunistas para este problema. Además, se diseñó un algoritmo para el DMHM y se realizaron simulaciones computacionales para compararlo con algoritmos de enrutamiento para redes oportunistas existentes.

Palabras clave: Algoritmos de redes oportunistas, redes tolerantes a retardos, problema de distribución máxima y homogénea de mensajes.

Abstract of the thesis presented by Héctor Zatarain Aceves as a partial requirement to obtain the Doctor of Science degree in Computer Science.

Algorithm design to solve the maximum uniform message distribution problem

Abstract approved by:

Dr. José Alberto Fernández Zepeda
Thesis Co-Director

Dr. Carlos Alberto Brizuela Rodríguez
Thesis Co-Director

In this thesis, we introduce the Maximum Uniform Message Distribution Problem (MUMD), that can be present in delay-tolerant communication networks where the destination of the messages is not present in the network. This type of behavior arises in scenarios of disaster or social conflicts where a global communication network is not available. In these scenarios, the people inside the affected area use their mobile devices to communicate in an opportunistic manner. Such that, when a device reaches a global communication network it sends all the messages in its memory, then we say that these messages were successfully delivered. We showed that the more uniformly distributed the messages are through the network, the higher the probability of successful deliveries. We model the MUMD as a computational problem and analyze the effectiveness of the existing opportunistic routing algorithms for the MUMD. Furthermore, we design an algorithm for the MUMD and perform computational experiments to compare it against existing opportunistic routing algorithms.

Keywords: Opportunistic routing algorithms, delay-tolerant networks, maximum uniform message distribution problem.

Dedicatoria

A mis padres Héctor y Cecilia.

A mi esposa Karina.

Agradecimientos

A mi esposa Karina por darme ánimos cuando más lo necesitaba. Gracias por tu cariño, paciencia, y amor incondicional.

A mis padres, Héctor y Cecilia por su cariño y apoyo incondicional, gracias por ser una pareja ejemplar y un ejemplo de vida.

A mis codirectores de tesis, el Dr. José Alberto Fernández y el Dr. Carlos Alberto Brizuela, por su constante orientación y apoyo que ayudaron de manera inconmensurable durante la elaboración de mi tesis.

A los miembros de mi comité de tesis, el Dr. José Antonio García Macías, el Dr. Salvador Villarreal Reyes, y el Dr. Rolando Menchaca Méndez, por sus observaciones, consejos, y críticas constructivas que sirvieron para mejorar en gran manera mi trabajo de investigación.

A todas las personas que pasaron por el cubo 103 durante mi estancia del doctorado, Daniel, Ismael, Alejandro, Antonio, Julio, Sarita, Rosa, Lino, y Roberto, por hacer la convivencia de cada día más amena y sobrellevar la carga de trabajo juntos.

A todos los investigadores del posgrado en ciencias de la computación por su enseñanza académica y su aporte en mi formación.

Al personal del departamento de ciencias de la computación por hacer mi estancia en la institución lo más agradable posible.

Al Centro de Investigación Científica y de Educación Superior de Ensenada por permitirme utilizar sus instalaciones durante mis estudios.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de doctorado. No. de becario: 232521.

Tabla de contenido

Página

Tabla de contenido (continuación)

Tabla de contenido (continuación)

Lista de figuras

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de tablas

Capítulo 1. Introducción

En el ámbito de las redes de comunicación, un algoritmo de enrutamiento define las reglas que sigue cada participante en la red para enviar mensajes entre una fuente y un destino (Tanenbaum y Wetherall, 2011). Sin embargo, en algunos escenarios particulares, el destino no está dentro de la red, no está disponible por tiempo indefinido o simplemente no se conoce su ubicación. Para estos escenarios, es necesario cambiar el paradigma de un enrutamiento clásico a uno que incremente la probabilidad de que un mensaje eventualmente alcance su destino.

Una forma de lograr lo anterior es hacer copias del mensaje y distribuirlas a través de la red, de tal forma que cada vértice en la red tenga una copia del mensaje; sin embargo, los principales aspectos que dificultan satisfacer este objetivo son: el desconocimiento de la ubicación del destino, los recursos limitados de la red y el desconocimiento total del estado de la red. A continuación, se explican las consecuencias generadas por estas dificultades.

La principal dificultad es que la ubicación del vértice destino es desconocida, por lo cual el procedimiento anterior intenta “acercar” el mensaje a su destino. Si la red pretende transmitir múltiples mensajes y muchos de sus destinos no están presentes en la red, el enfoque anterior puede generar copias excesivas, saturando los canales de comunicación y generando un decremento en el desempeño de la red.

Además, en la mayoría de los escenarios de aplicación, los recursos de la red (capacidad de almacenamiento, ancho de banda, batería, etc.) son bastante limitados. Es por ello que muchos vértices competirán por los recursos cuando traten de maximizar el número de copias de sus mensajes. Para mitigar o controlar este conflicto de interés, es deseable diseñar un mecanismo que limite la generación de copias de cada mensaje. Este enfoque puede ser útil para ayudar a evitar la sobrecarga de la red.

Finalmente, para este tipo de escenarios en sistemas distribuidos, puede ser muy costoso, o incluso inviable, limitar el número de copias utilizando un ente central. Así que, un enfoque distribuido que tome decisiones solamente con base en información local debería ser el más apropiado. Algunos de los escenarios más representativos en donde se presenta la problemática introducida se describen a continuación.

1.1. Planteamiento del problema

Las redes de comunicación son una herramienta importante para una comunicación efectiva entre personas. Durante catástrofes o conflictos sociales, las redes de comunicación están propensas a fallar o ser bloqueadas por largos periodos de tiempo. Particularmente, una de las redes que en los últimos años ha tomando mayor relevancia por su robustez, son las redes tolerantes a fallos o redes oportunistas (Lilien *et al.*, 2006; Pelusi *et al.*, 2006). Este tipo de redes son las que más se asemejan al estado de las comunicaciones en una zona de desastre.

En desastres naturales, las redes y los sistemas de sensores remotos toman una importancia crucial para sobrellevar las comunicaciones que en la práctica pueden salvar vidas. Desde el surgimiento de la Internet se ha originado un mayor interés para algoritmos que manejan la comunicación de un conjunto de computadoras interconectadas. No obstante, la mayoría de los trabajos de investigación suponen que cada participante actúa conociendo la información global del sistema como la topología y el número de nodos del sistema. Ésto es una suposición fuerte en el diseño de algoritmos en general y además pueden existir agentes egoístas en la red con información privada.

El potencial de las redes oportunistas para mitigar la falta de comunicación en todo tipo de situaciones de emergencia (incluyendo desastres naturales y humanos) es especialmente notable. Las víctimas y los daños frecuentemente se agravan por los problemas que enfrentan los socorristas debido a la falta de comunicaciones adecuadas en el área de desastre. Algunas situaciones a gran escala en donde ha ocurrido lo anterior son los ataques terroristas del 11 de septiembre de 2001 en EUA, los tsunamis del sur de Asia, los terremotos en Haití, los huracánes Irma y María, entre otros. A continuación se describen dos ejemplos puntuales, durante un desastre natural y un conflicto social, donde las redes de comunicación fallan por un largo periodo de tiempo.

Un desastre natural bien documentado es el del terremoto de Chile de 2010 (ver Figura 1a), en donde los servicios de telefonía sufrieron problemas graves durante las primeras horas del terremoto, especialmente debido a la congestión de las llamadas,

las cuales llegaron a superar 10 veces el tráfico de un día normal[1]. En el caso de la telefonía fija, no existieron daños de consideración respecto a la infraestructura. Este servicio comenzó a recuperar su normalidad en las horas posteriores una vez que se corriguieron las fallas en la red de distribución de energía eléctrica (Araneda *et al.*, 2010). Aun así la telefonía fija reaccionó mejor que la telefonía móvil que estuvo varias horas sin servicio y en algunas regiones días. Respecto al servicio de internet, éste estuvo funcionando sin problemas durante las primeras horas del terremoto. Sin embargo, en las horas posteriores, se registraron problemas en algunos servidores nacionales dejando a Chile prácticamente incomunicado con el exterior por cerca de 24 horas (Sepúlveda *et al.*, 2010).

Un ejemplo de conflicto social a gran escala fue el ocurrido durante las manifestaciones en algunos países árabes de 2010 a 2013, conocido como la primavera árabe. Durante estas manifestaciones se presentó un efecto dominó, donde la primera revolución se presento en Túnez, seguida de Egipto (ver Figura 1b), Libia, Yemen, y Siria, entre otras. En la mayoría de estos conflictos sociales, el gobierno interrumpía las telecomunicaciones (Internet, telefonía celular, etc.) durante un largo periodo de tiempo, obligando a las personas dentro de la zona incomunicada a utilizar sus dispositivos móviles de forma oportunista para comunicarse entre sí, siendo éstos cruciales para la organización de los manifestantes. Para evitar ésto, a partir del 2011 se interrumpió el Internet en varios países del norte de África, e.g., en Libia y Egipto se bloqueó el Internet durante 3 y 5 días, respectivamente (Dainotti *et al.*, 2011).

Ya sea un desastre natural o un conflicto social, una amenaza común que puede ocurrir en estos problemas es la falta de comunicaciones adecuadas en el área afectada. Por lo tanto, se debe estudiar la forma de proveer medios de comunicación alternativos confiables en emergencias. Lo anterior es un reto en las áreas de comunicación y tecnologías de la información. En esta tesis, se define y modela esta problemática como un problema computacional llamado problema de Distribución Máxima y Homogénea de Mensajes (DMHM). Además, se propone un algoritmo de enrutamiento oportunista que genera soluciones aproximadas a este problema. Finalmente, se demuestra la utilidad de este algoritmo experimentalmente en distintos escenarios y se compara con algoritmos de enrutamiento oportunista del estado del arte.

[1](Cooperativa.cl, 2011).

(a) Terremoto de Chile.

(b) Revolución de Egipto.

Figura 1. Ejemplos de dos desastres a gran escala: un desastre natural y un conflicto social. a) Imagen capturada momentos después del terremoto de Chile en 2010. b) Encuentro entre un protestante y un cerco policial en Egipto durante la primavera árabe.

1.2. Objetivos

En está sección se establece el objetivo general de este trabajo de investigación así como una serie de objetivos particulares que ayudarán a cumplir este objetivo.

1.2.1. Objetivo general

El objetivo general de esta investigación tiene como finalidad contestar la siguiente pregunta de investigación:

¿Cuál es el mecanismo más adecuado para distribuir mensajes en un modelo de red sin cobertura global, de tal manera que el número de copias de cada mensaje sea máximo y uniforme, considerando un almacenamiento limitado de cada participante de la red y distintos escenarios de movilidad, energía, y número de participantes?

1.2.2. Objetivos específicos

1. Definir y modelar formalmente el problema de distribución máxima y homogénea de mensajes como un problema computacional.
2. Seleccionar el simulador más adecuado para evaluar el desempeño de los protocolos y propuestas.

3. Determinar qué algoritmos de enrutamiento del estado del arte son los más adecuados para el problema de distribución máxima y homogénea de mensajes.
4. Diseñar algoritmos de enrutamiento que generen soluciones aproximadas al problema definido.
5. Implementar los algoritmos y mecanismos propuestos para verificar su desempeño de forma experimental.
6. Determinar las métricas más adecuadas para discriminar las soluciones candidatas al problema.
7. Analizar el impacto en el desempeño de los algoritmos al considerar diferentes escenarios, específicamente:
    - Diferentes números de dispositivos en la misma área.
    - Áreas sin restricción de movilidad y con restricciones dentro de un mapa.
    - Capacidades de almacenamiento uniformes y no uniformes.
    - Distintos niveles de energía de los dispositivos en la red.
    - Diferentes tiempos de llegadas de los participantes a la red.

1.3. Contribución al conocimiento

Las contribuciones principales de esta tesis son las siguientes:

- El planteamiento de un nuevo problema computacional denominado: el problema de distribución máxima y homogénea de mensajes (DMHM). El DMHM modela el comportamiento de un conjunto de individuos en un escenario donde las comunicaciones globales no están disponibles y donde todos desean enviar un mensaje a un destino que no está presente en el escenario. Estos individuos son capaces de comunicarse entre ellos al crear una red oportunista con sus dispositivos móviles. En esta tesis se plantean varios escenarios posibles en donde se presenta este problema.
- Un algoritmo que genera soluciones aproximadas al DMHM sin necesidad de conocer el estado de la red, como su topología, número de participantes o patrón

de contactos. Este algoritmo utiliza un mecanismo que maximiza la probabilidad de salida de cada mensaje del área afectada y administra de forma eficiente los recursos en un entorno modelado como una red oportunista.

- El análisis experimental del DMHM. Además, de una comparación empírica entre el algoritmo propuesto y otros algoritmos de ruteo del estado del arte para redes oportunistas aplicados al DMHM. Incluso, se muestra que la solución aproximada generada por nuestra propuesta es muy cercana a la solución óptima.

- La propuesta conceptual de construir una red de comunicación provisional, basada en dispositivos inalámbricos móviles, para remplazar temporalmente las redes de comunicación tradicionales cuando éstas fallan. En esta red provisional, cada dispositivo móvil ejecuta un protocolo para el DMHM, buscando maximizar la probabilidad de entrega exitosa de cualquier mensaje. El algoritmo propuesto genera soluciones aproximadas para el DMHM, por lo cual puede ser útil para implementar el protocolo de comunicación para los dispositivos inalámbricos móviles.

1.4. Organización de la tesis

El resto de este documento se organiza de la siguiente manera. El Capítulo 2 introduce algunos conceptos básicos, describe trabajo previo relacionado y se analiza la factibilidad de utilizar algoritmos de enrutamiento para redes oportunistas aplicados al DMHM. El Capítulo 3 describe de manera general y formal al DMHM como un problema computacional. El Capítulo 4 describe el algoritmo EDEN, el cual genera soluciones aproximadas al DMHM. El Capítulo 5 describe el diseño experimental. El Capítulo 6 presenta los resultados de la simulaciones experimentales. Finalmente, el Capítulo 7 presenta las conclusiones y algunas ideas de investigación para trabajo futuro.

Capítulo 2. Trabajo relacionado

En este capítulo se discuten los antecedentes desde distintos puntos de vista relacionados con el problema DMHM. Primero, se describen algunas herramientas tecnológicas que se han utilizado para intentar mitigar los problemas de comunicación que surgen durante un desastre. Posteriormente, se contextualiza el problema en el área de redes de comunicación, determinando que el concepto de red más adecuado para la problemática abordada es una del tipo oportunista. Por último, se analizan los algoritmos de enrutamiento existentes, especializados en la diseminación de datos y en redes oportunistas. Simultáneamente, se analiza la factibilidad de uso de estos algoritmos para el problema DMHM.

2.1. Herramientas tecnológicas para mitigar problemas de comunicación

En esta sección se describen ejemplos de dos enfoques distintos para resolver el problema del déficit de comunicación durante un desastre. Primero, se describen múltiples sistemas denominados EMIS, *Emergency Management Information System*, (Kwan y Lee, 2005). Segundo, se describen algunas opciones de aplicaciones de mensajería celular que no requieren de una conexión a una red de telecomunicación para funcionar.

2.1.1. Sistemas de información de gestión de emergencias

Los EMIS son sistemas basados en computadoras para apoyo durante catástrofes. El objetivo de este tipo de sistemas es, en la medida de lo posible, proveer información gráfica y en tiempo real a las personas durante un desastre. Una emergencia se divide en cuatro fases: preparación, mitigación de riesgos, respuesta, y recuperación. A continuación se describen cuatro EMIS que se enfocan principalmente en auxiliar durante las últimas dos fases.

En China, se ha tenido una gran atención a partir del año 2000 en los sistemas de estrategias de mitigación de desastre en caso de terremotos y sistemas de apoyo para la toma de decisiones (Yan *et al.*, 2010). Se han implementado sistemas de emergencias para terremotos, los cuales se enfocan principalmente a la predicción de daños.

Algunos de los problemas de este sistema es la falta de respuesta y coordinación de los cuerpos de rescate orientada a las situaciones que ocurren después del terremoto.

En México también se ha generado interés en este tipo de sistemas. Un ejemplo de ello es RieSis, un sistema de atención a contingencia sísmica creado por CIC-IPN. La idea de este sistema era visitar su página web[1] (la página web ya no es funcional), en la cual los usuarios podían reportar sitios que necesitan atención en tiempo real. Estos sitios se utilizaban para facilitar la logística de los grupos de rescate. La principal restricción de este sistema es que estaba pensado solamente para la Ciudad de México.

Otro sistema que no es exclusivo del área metropolitana, es una aplicación para teléfonos inteligentes[2] que forma parte del proyecto RESNOM (Red sísmica del noroeste de México). Está aplicación la creó un grupo de investigadores del posgrado de ciencias de la tierra del CICESE. Esta aplicación proporciona información en tiempo real sobre la magnitud y ubicación de un sismo. Además, ofrece información básica de sismos, consejos y teléfonos de emergencia. La aplicación está disponible para celulares iPhone desde el 19 de abril del 2012 y para celulares Android desde el 3 de febrero del 2013 (Vargas, 2013).

Por otra parte, en el trabajo de Ramesh *et al.* (2012) se propone una arquitectura móvil de red integrando la participación de nuevos dispositivos por medio de sensado y con la finalidad de revivir las comunicaciones en áreas de desastre; en su artículo se limitan a implementar un protocolo para celulares para intentar recuperar la red de comunicaciones sin tomar en cuenta aspectos importantes como la energía o la memoria de los dispositivos. Similarmente, en Aloi *et al.* (2017) se propone la arquitectura multicapas SENSE-ME, que permite la creación de redes provisionales con teléfonos móviles. Los autores evalúan la factibilidad de SENSE-ME en un escenario donde los dispositivos comparten información para detectar una emergencia y para definir la mejor ruta de evacuación de un edificio aplicando algoritmos basados en consenso.

Finalmente, Google crisis response (Google., 2005) consiste en un conjunto de herramientas informativas durante desastres naturales que pueden auxiliar a personas con acceso a Internet. Estas herramientas informativas incluyen alertas públicas, un

[1](CIC., 2013). http://riesis.cic.ipn.mx/RieSis/
[2](CICESE., 2013). http://resnom.cicese.mx/

buscador de personas, un mapa de crisis, entre otras. La principal limitación de este sistema es que es inútil en la ausencia de Internet.

2.1.2. Aplicaciones para comunicación alternativa

Existen múltiples intentos aislados para superar los problemas de comunicación causados por escenarios de desastres o conflictos sociales. A continuación se dan dos ejemplos puntuales de aplicaciones disponibles para teléfonos móviles que permiten vías de comunicación alternativa.

El primero es Twitmight (Hossmann *et al.*, 2011). Twitmight es un cliente de Twitter para dispositivos móviles con un modo de desastre (activado por el usuario) para comunicarse sin requerir acceso a Internet. Twitmight permite la transmisión de los tweets al usar una red entre pares (*peer-to-peer*) como medio de comunicación. Twitmight crea una red móvil ad-hoc con la suposición de que antes de la pérdida de conectividad, todos los dispositivos móviles se conectan a un servidor para obtener certificados de seguridad. Esto implica que nuevos usuarios no pueden unirse a la red durante un desastre. Otra desventaja de Twitmight es que es reactivo y no proactivo, i.e., los usuarios necesitan habilitarlo.

Por último está Firechat, una aplicación propietaria para el envío público o privado de mensajes sin utilizar Internet ni datos celulares[3]. Firechat se ha utilizado en múltiples manifestaciones y conflictos sociales donde el Internet y las redes celulares se censuraron o estaban sobrecargadas. Algunos ejemplos de su uso son: Irak y Hong Kong en el 2014, Ecuador y Cataluña en el 2015, y más recientemente, en la Convención Nacional Demócrata en Filadelfia, EUA, en Julio del 2016.

2.2. Redes de comunicación inalámbricas

En esta sección se describen las redes de comunicación inalámbricas y su clasificación desde un enfoque general hasta llegar específicamente a las redes ad hoc. Se definen los tipos de redes ad hoc y el término de red oportunista. Posteriormente se muestra una comparación detallada entre una red oportunista y una red ad hoc, además de una comparación entre una red oportunista y una red tolerante a fallos.

[3](OpenGarden., 2014). http://www.opengarden.com/firechat.html.

Las redes de comunicación permiten una comunicación efectiva entre entes o personas. Las redes inalámbricas *(WNETs - Wireless Networks)* son redes de comunicación que no necesitan tener conexiones cableadas entre sí. Las WNETs se pueden clasificar según diversos aspectos como rango de cobertura, topología, entre otros. En la Figura 2 se muestran los tipos de redes por cobertura, en donde se encuentran las redes inalámbricas de cobertura personal (WPAN), las redes inalámbricas de área local (WLAN), las redes inalámbricas ad hoc, las redes inalámbricas de área metropolitana (WMAN), las redes inalámbricas de área amplia (WWAN), las redes inalámbricas de compañías celulares (CEL. NET.), las redes inalámbricas globales, y las redes inalámbricas espaciales. En la misma Figura 2 se muestran los tipos de redes por topología, la que a su vez se divide en dos, topología centraliza y descentralizada. Las redes de topología centralizada son las que se coordinan por medio de un servidor central, en este tipo de redes no se profundiza ya que no son relevantes para el presente trabajo.

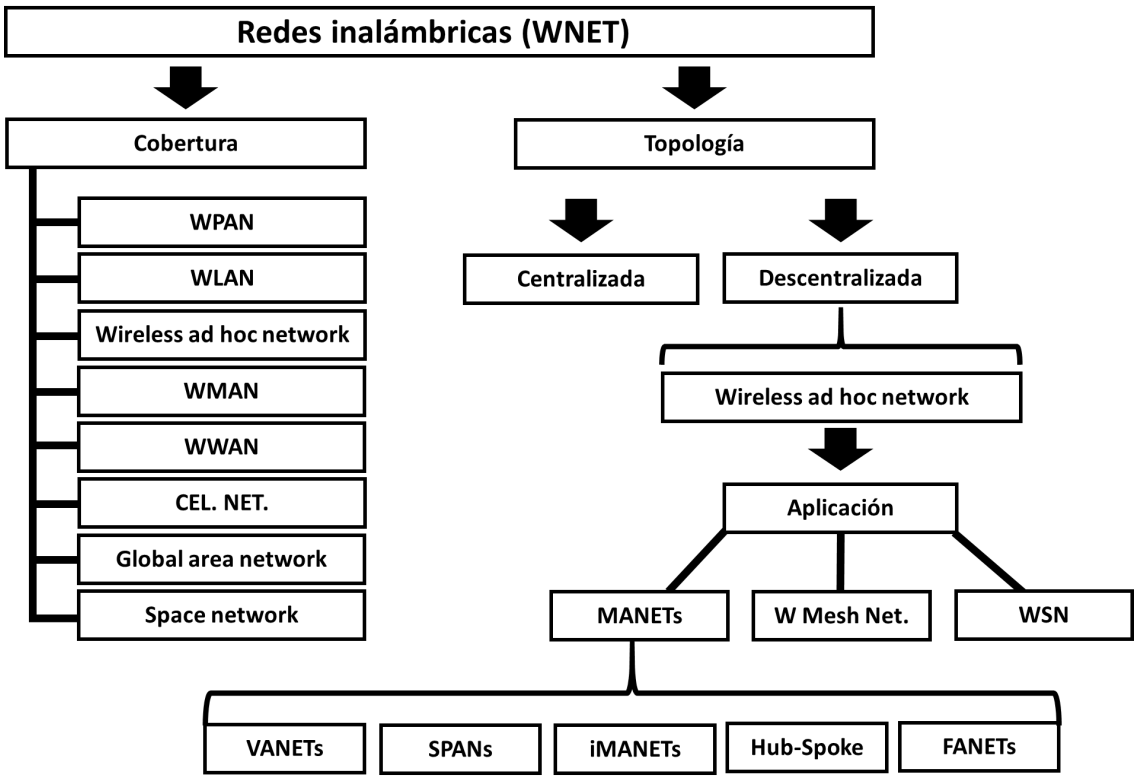


Figura 2. Clasificación general de las redes de comunicación inalámbricas por cobertura y topología.

Dentro de la clasificación de redes de topología descentralizada está la red inalámbrica ad hoc. La que se define como un conjunto de nodos que se comunican con interfaces inalámbricas y forman una red de conexión temporal. Una red ad hoc es autónoma, es decir, sus nodos se pueden configurar de forma automática y además

se crea conforme se van descubriendo los nodos, con el objetivo de llevar a cabo un propósito específico. Las redes ad hoc se clasifican dependiendo de su aplicación en:

- Red ad hoc móvil *(MANET - Mobile Ad Hoc Network)*.- Red ad hoc con nodos móviles o donde un conjunto de nodos están en plataformas móviles.
- Red inalámbrica mallada *(W. Mesh Net. - Wireless Mesh Networks)*.- Red ad hoc donde los nodos se dedican a enrutar con base en una topología fija mallada.
- Red inalámbrica de sensores *(WSN - Wireless Sensor Networks)*.- Red de sensores autónomos que se comunican entre sí para monitorear generalmente condiciones ambientales.

Una red ad hoc es una colección de dispositivos móviles inalámbricos que forman una red temporal sin la ayuda de una infraestructura establecida o un administrador centralizado. En tal escenario, es necesario que cada dispositivo solicite ayuda de los otros dispositivos para hacer llegar un paquete a su destino, debido al alcance de transmisión limitado de cada dispositivo móvil (Johnson y Maltz, 1996). Note que las MANETs son las redes que más se asemejan a una red que pudiera ser útil en el escenario del DMHM. Existen diferentes tipos de MANETs que se enlistan a continuación:

- Redes vehiculares ad hoc (*VANETs - Vehicular Ad hoc Networks*). Se utilizan para permitir la comunicación de vehículos en una carretera.
- Redes de teléfonos inteligentes ad hoc (*SPANs - Smart Phone Ad hoc Networks*). Este tipo de redes aprovechan las interfaces de comunicación (Bluetooth y WiFi) integradas en la mayoría de los teléfonos móviles para permitir la comunicación punto a punto sin requerir puntos de acceso a Internet, redes celulares, o infraestructura tradicional.
- Redes ad hoc móviles basadas en internet (*iMANETs - Internet based Mobile Ad hoc Networks*). Es un tipo de red inalámbrica ad hoc que soporta los protocolos de Internet como IP y TCP/UDP.
- Redes de nodo central parlante ad hoc (*Hub-Spoke Ad hoc Networks*). Este tipo de redes están formadas por múltiples sub-MANETs que pueden estar conectadas en un Hub-Spoke VPN clásico para crear una MANET distribuida geográficamente.

- Redes voladoras ad hoc (*FANETs - Flying Ad hoc Networks*). Están formadas por vehículos aéreos no tripulados, lo cual produce un alto grado de movilidad y permite proveer conectividad en áreas remotas.

Otro tipo de red que algunos autores consideran un tipo de red inalámbrica ad hoc y que se utiliza en esta tesis es la que se denominan 'redes oportunistas' *(OppNets - Opportunistic Network)* descrita en la sección 2.3. Una comparación entre estos dos tipos de redes se describe en la sección 2.3.1. En las siguientes secciones se profundiza en los tipos de protocolos para MANETS. Posteriormente se hace un análisis de las similitudes y diferencias entre MANETs y OppNets.

2.2.1. Clasificación de protocolos para redes ad hoc

Existen muchas clasificaciones de protocolos para redes ad hoc, en esta sección se describen algunas de las más relevantes. La Figura 3 ilustra los tipos de clasificaciones que se describen a continuación.

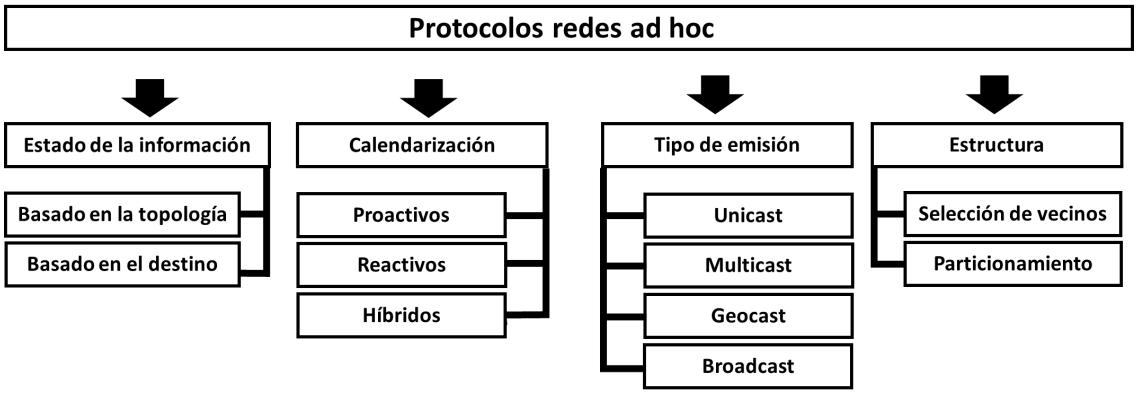


Figura 3. Tipos de clasificaciones de para redes ad hoc.

2.2.1.1. Clasificación por calendarización

Ésta es la clasificación general o más utilizada para los protocolos de redes ad hoc. En ella se distingue entre los protocolos que obtienen y actualizan la información de ruteo continuamente para cada destino y los que solamente obtienen información de ruteo bajo demanda.

- **Proactivos (*table-driven*)**.- Las rutas se calculan previamente. Cada nodo mantiene una o más tablas con información de ruteo. Los nodos intercambian infor-

mación de ruteo periódicamente y en respuesta a cambios en la topología, intentando mantener las tablas consistentes.

- **Reactivos (*on-demand*)**.- Las rutas se calculan solamente cuando un nodo fuente lo requiera. Se determina la ruta cuando todas las posibles rutas ya se han examinado.
- **Híbridos**.- Se comporta de forma proactiva para el vecindario del nodo y de forma reactiva para los demás nodos.

2.2.1.2. Clasificación por tipo de emisión

Los protocolos por tipo de emisión (*type of cast*) se refiere al número o tipos de destino para los que se emiten los mensajes. Éstos se describen a continuación:

- **Unicast**. Emisión de una fuente a un destino.
- **Multicast**.- Emisión de una fuente a múltiples destinos, para ello se construye un árbol de enrutamiento.
- **Geocast**.- Emisión de una fuente a múltiples destinos situados en un área geográfica específica.
- **Broadcast**.- Emisión a todos los destinos dentro de la red.

2.2.1.3. Clasificación por estado de la información

En esta clasificación se hace la diferencia en qué información de la topología almacena cada nodo.

- **Basado en la topología (*topology based*)**.- Cada nodo mantiene información de la topología de toda la red.
- **Basado en el destinatario (*destination based*)**.- Cada nodo solamente mantiene la información necesaria para conocer sus vecinos más cercanos.

2.2.1.4. Clasificación por estructura

En la clasificación por estructura se define si existe alguna jerarquía de nodos o si todos se consideran iguales.

- **Selección de vecino (*neighbor selection*)**.- Cada nodo se enfoca en enrutar mensajes a un subconjunto de sus vecinos.
- **Particionamiento (*partitioning*)**.- La red se particiona topológicamente. Se define un nodo dentro de cada partición con mayor jerarquía que enruta mensajes hacia afuera de la partición y los demás nodos se enfocan a enrutar dentro la partición.

Una vez descritas las clasificaciones más relevantes, es importante definir qué tipo de protocolo es el más adecuado a utilizar dependiendo de las características de la red ad hoc. Kuosmanen (1999) define qué tipo de protocolos son los más aptos y se fundamenta en suposiciones intuitivas con base en algunas características de la red. Él menciona que en el caso de la clasificación por calendarización, si se tiene una red con alta movilidad, los protocolos proactivos tienen un pobre desempeño cuando se comparan con los reactivos. Lo anterior es debido a que los protocolos reactivos aumentan el tráfico de la red al intentar mantener las tablas de ruteo actualizadas. Para la clasificación por estado de la información, los protocolos basados en topología tienen la desventaja de diseminar la información topológica por la red. Por lo que al crecer la red tienen mejor desempeño los protocolos basados en el destinatario. Por último, en la clasificación por estructura. Para redes muy grandes se pueden utilizar estructuras jerárquicas, pero éstas son difíciles de mantener dada la movilidad de la red. Por lo que los protocolos de selección de vecindario son preferibles a los protocolos de partición en redes grandes y con alta movilidad.

2.3. Redes oportunistas

En la literatura, las redes oportunistas (OppNets) (Lilien *et al.*, 2006; Pelusi *et al.*, 2006), se definen como una red donde las oportunidades de comunicación (contactos) son intermitentes y asíncronas, de tal forma que un trayecto de extremo a extremo

entre la fuente y el destino puede que nunca exista. El objetivo principal de las OppNets es la diseminación de los mensajes y la optimización de los recursos de la red (Lilien *et al.*, 2007; Huang *et al.*, 2008).

Las OppNets están formada por nodos que son pequeños dispositivos móviles, generalmente transportados por personas, y se comunican a través de un enlace inalámbrico. Estos dispositivos no requieren infraestructura para funcionar. Los nodos son capaces de descubrirse entre sí automáticamente y se comunican sin intervención del usuario. Dada la movilidad de los nodos su conexión es intermitente.

2.3.1. Comparación entre redes oportunistas y redes ad hoc móviles

Las redes oportunistas y las MANETs tienen mucho en común como menciona Heinemann (2007), las similitudes se dan en los siguientes aspectos: utilizan una arquitectura descentralizada, están formadas por dispositivos móviles, la conexión entre nodos es intermitente, los dispositivos que forman la red pueden contar con diferentes características (potencia, velocidad de transmisión, memoria). Incluso muchas técnicas de MANETs se pueden utilizar en OppNets y viceversa. Las principales diferencias entre OppNets y MANETs las mencionan Heinemann (2007) y Pelusi *et al.* (2006). Éstas se especifican a continuación.

Las OppNets utilizan comunicación asíncrona, mientras que las MANETs utilizan comunicación síncrona. Las MANETs requieren ruteo en tiempo real, por su parte las OppNets dan mayor énfasis en la diseminación de la información más que en rutear en tiempo real. Otro aspecto es la cooperación, ya que en MANETs se supone que todos los nodos quieren contribuir y están dispuestos a enrutar cualquier tráfico, lo que no sucede en OppNets. En MANETs se supone que los nodos conocen completa o parcialmente la topología y en OppNets no. Desde un punto de vista de la pila de protocolos, las MANETs están en la capa de red y las OppNets en la capa de aplicación y solicita algunas funciones a la capa de red. Por último, se dice que las OppNets son una evolución de las MANETs.

Otra diferencia relevante entre este tipo de redes se da al detallar las funcionalidades básicas de cada tipo de red. Estas funcionalidades básicas son el descubrimiento de nodos, intercambio de mensajes y administración de identidad. En MANETs el des-

cubrimiento de nodos implica que se debe estar al tanto cuándo un nodo entra o sale del rango de comunicación de la red, en OppNets cada nodo solamente debe estar informado de los nodos que están dentro de su rango de comunicación de forma directa. El intercambio de mensajes entre MANETs se realiza por medio de comunicaciones de saltos múltiples y en OppNets solamente por contacto directo, es decir comunicación de un salto. Por último, la administración de identidad en MANETs se refiere a que cada nodo y mensaje en la red deben estar identificados, mientras que en OppNets por lo general los individuos son anónimos.

2.3.2. Comparación entre redes oportunistas y redes tolerantes a retrasos

Se denomina como *red tolerante a retrasos* (*delay-tolerant networks* DTNs) a las redes inalámbricas móviles que tienen una conectividad intermitente entre sus nodos. La arquitectura de una DTN consiste en una red de nodos independientes capaces de comunicarse entre sí, pero con comunicación ocasional entre ellos. En la literatura, los conceptos de “redes tolerantes a retardos”, “redes tolerantes a fallos” y “redes oportunistas” se utilizan comúnmente de forma intercambiada. En Pelusi *et al.* (2006) establecen que la principal diferencia entre estos conceptos es que en DTNs se supone que se conoce la topología de la red, mientras que en redes oportunistas no necesariamente se conoce la topología de la red. Algunos conceptos en redes oportunistas provienen de estudios en redes tolerantes a retrasos.

Es tanta la similitud que en algunos trabajos de la literatura se contradicen con respecto a qué red es más general que otra. Huang *et al.* (2008) consideran las OppNets como una subclase de las DTNs. En donde la comunicación es intermitente y la trayectoria entre un nodo fuente y destino podría nunca existir. Por otra parte, Pelusi *et al.* (2006) mencionan que las OppNets son un concepto más general e incluyen a las redes DTNs. Estos autores afirman lo anterior argumentando que los nodos en OppNets pueden actuar como puerta de enlace, por lo que las OppNets son más flexibles que las DTNs.

En general, las DTNs evolucionaron de las MANETs al relajar algunos de los requerimientos de éstas y al permitir un alto grado de movilidad entre los nodos participantes. Las redes DTN y MANET comparten similitudes en ciertos aspectos, como: falta de infraestructura, recursos limitados, movilidad de los nodos resultando en una frecuente

partición de la red, etc. Sin embargo, tienen muchas diferencias entre si.

Las DTN tienen aplicaciones prácticas en múltiples campos y en la literatura se ha estudiado a profundidad sus propiedades, taxonomías y algunas de sus aplicaciones. En años recientes, el interés en la investigación utilizando DTN se ha reavivado ya que surgieron nuevos dominios de aplicación basados en redes emergentes que requieren el soporte de una red tolerante a retardo y por lo tanto pueden utilizar técnicas de enrutamiento y difusión específicas para este tipo de redes.

Algunos, ejemplos de estas nuevas áreas son las redes centradas en la información (*Information Centric Network*, o *ICN*), ICN móviles, red de datos con nombre (*Named Data Network*, o *NDN*), Internet de las cosas (*Internet of Things*, o *IoT*), entre otras.

2.4. Algoritmos de enrutamiento para redes oportunistas

En esta tesis, se propone la construcción de una red comunicación alternativa utilizando dispositivos móviles para remplazar temporalmente a las redes de comunicación tradicionales cuando están fuera de servicio, lo cual puede ser consecuencia de un escenario post-desastre.

Muchos esfuerzos de investigación se enfocan en analizar diferentes tipos de tecnologías de red para determinar cuál es la mejor para mitigar la falta de comunicación en escenarios post-desastre (Saha *et al.*, 2011; Reina *et al.*, 2015; Miranda *et al.*, 2016; Rosas *et al.*, 2016). En general, para estos escenarios, las OppNets son siempre una opción adecuada. Es por ello que en esta sección se describen algunos de los algoritmos de enrutamiento existentes para OppNets. Además, se analizan la factibilidad, ventajas, y desventajas de utilizar estos algoritmos para resolver el DMHM.

En la literatura, existen algunos algoritmos de enrutamiento que pueden utilizarse, con algunas modificaciones, para resolver el DMHM. Muchos de ellos están diseñados específicamente para OppNets. Una clasificación de los algoritmos para OppNets más representativos se muestran en la Figura 4. Estos protocolos utilizan un mecanismo para mejorar el caudal eficaz (*throughput*), la tasa de entrega, el uso del almacenamiento, duración de la batería, difusión de la información (Clementi *et al.*, 2015), y otras métricas bien conocidas. Sin embargo, en este contexto, y hasta donde el autor de este documento sabe, no existe un protocolo que por si solo solucione el DMHM.

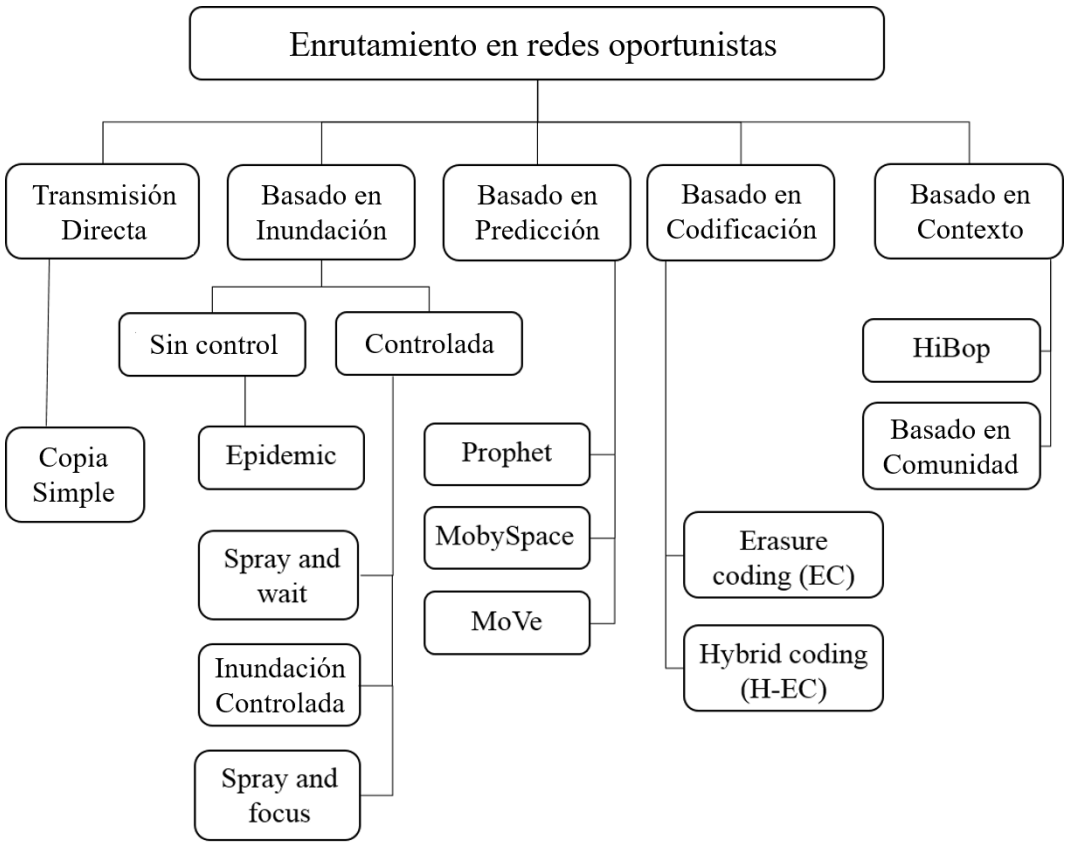


Figura 4. Clasificación de algoritmos de enrutamiento para redes oportunistas. Figura extraída de Poonguzharselvi y Vetriselvi (2013).

Existen muchos algoritmos de enrutamiento probabilísticos o predictivos para OppNets, como PROPHET (Lindgren *et al.*, 2004), MaxProp (Burgess *et al.*, 2006) y MoVe (LeBrun *et al.*, 2005). Saha *et al.* (2011) comparan múltiples algoritmos de enrutamiento para OppNets. Ellos determinan que PROPHET y MaxProp producen un mejor desempeño utilizando un modelo de movilidad de agrupamiento (*cluster mobility model*). Este modelo intenta replicar de manera fiable un escenario después de un desastre.

Por otra parte, Aschenbruck *et al.* (2009) proponen un nuevo modelo de movilidad para escenarios de desastres. Este modelo de movilidad considera obstáculos, y vértices que entran y salen de la red. Ellos comparan su modelo de movilidad contra otros modelos de movilidad tradicionales realizando simulaciones para escenarios de desastre con diferentes características. Sus resultados demuestran que los modelos de movilidad pueden impactar los resultados en el desempeño de algoritmos de enrutamiento.

Finalmente, Martín-Campillo *et al.* (2013) comparan múltiples algoritmos de enrutamiento para OppNets utilizando el modelo de movilidad propuesto por Aschenbruck *et al.* (2009). Martín-Campillo *et al.* (2013) determinan que MaxProp es el mejor algoritmo en términos de entrega de mensajes en el escenario que analizan; sin embargo, también concluyen que no existe un algoritmo de enrutamiento que sea el mejor para todos los escenarios.

Debido a que el DMHM supone que el vértice destino es desconocido o no está disponible en la red, no es posible utilizar los algoritmos de enrutamiento probabilísticos ni los que están basados en contexto. Ya que para trabajar adecuadamente, estos algoritmos requieren la presencia del vértice destino en la red.

Un primer enfoque directo e inocente para intentar resolver el DMHM es realizar una emisión de los mensajes inundando la red (*flooding*). Un ejemplo conocido de este enfoque es el protocolo de inundación epidémica propuesto en (Vahdat y Becker, 2000). Este protocolo reenvía copias de cada mensaje a cada vértice que encuentra, tratando de simular cómo se propaga una enfermedad durante una epidemia. Desafortunadamente, flooding causa ciertos problemas bien conocidos, tal como las tormentas de emisión (*broadcast storms*) (Tseng *et al.*, 2002), saturación de los canales de comunicación, ineficiente administración de la energía, y una alta redundancia de datos.

Un enfoque más factible que mitiga algunos de los problemas mencionados puede ser inundación controlada. Un ejemplo de este mecanismo es el algoritmo *Spray and Wait* (SnW) y sus variantes (Spyropoulos *et al.*, 2005, 2007; Wang *et al.*, 2015). Estos algoritmos pueden generar un número de copias controladas de cada mensaje en la red; sin embargo, estos algoritmos suponen que cada vértice conoce el número total de vértices, n, en la red, lo cual es una fuerte suposición en sistemas de cómputo distribuido (Caragiannis *et al.*, 2005).

Para ser consistentes con las observaciones previas, el algoritmo EDEN (ver Capítulo 4) no supone que cada vértice conoce el parámetro n. Además, no requiere la presencia de ningún vértice destino en la red. El algoritmo EDEN utiliza la filosofía de envió de mensajes del algoritmo SnW en modo binario. A continuación, se describe el algoritmo SnW.

2.4.1. Spray and Wait

Spyropoulos *et al.* Spyropoulos *et al.* (2005) proponen el algoritmo Spray and Wait (SnW) como un algoritmo de enrutamiento para redes oportunistas.

El algoritmo SnW utiliza dos fases, como su nombre lo indica, primero dispersar y después esperar. La fase de dispersar restringe cada vértice para generar solo L copias de sus mensajes. Esta fase tiene dos modos: el modo simple y el modo binario. En el modo simple, el vértice fuente envía una copia a L diferentes vértices. En el modo binario, el vértice fuente envía L_i copias al *i-ésimo* vértice diferente, donde $L_0 = L$ y $L_i = \lceil \frac{L_{i-1}}{2} \rceil$ para $1 \leq i \leq \lceil \log L \rceil$. Finalmente, en la fase de espera cada vértice con una copia inhibe la transmisión de este mensaje hasta que contacte con el vértice destino.

Una variante de SnW, llamada *Spray and Focus* (Spyropoulos *et al.*, 2007), remplaza la segunda fase del algoritmo SnW. Al hacer lo anterior, en lugar de inhibir la entrega de mensajes en los vértices con una copia restante, incentiva que estos vértices envíen las copias a otros que tengan mayor probabilidad de encontrar al destino.

SnW y Spray and Focus inicialmente limitan el número de copias de cada mensaje al parámetro L. El valor más adecuado para L depende del escenario de aplicación; sin embargo, Spyropoulos *et al.* (2008) sugieren que para SnW en modo binario, L debe tener un valor del 10 % al 15 % de n. De manera similar, para Spray and Focus, L debe de tener un valor del 5 % al 10 % de n.

2.5. Resumen

En este capítulo, inicialmente se analizan algunas herramientas tecnológicas que se han utilizado para mitigar los problemas de comunicación en escenarios similares al modelado por el DMHM. Después, se presentan algunas aplicaciones para celulares que se han utilizado para generar comunicación en entornos sin telecomunicaciones funcionales.

Posteriormente, se describen los tipos de redes inalámbricas. Partiendo desde un enfoque general a uno particular, con el objetivo de definir la red más adecuada para modelar el escenario del DMHM. Se acota a las OppNets o DTNs como las redes más apropiadas para este objetivo.

Finalmente, se describen los algoritmos de enrutamiento para OppNets más populares y se discute la factibilidad de utilizar estos algoritmos para generar soluciones aproximadas para el DMHM.

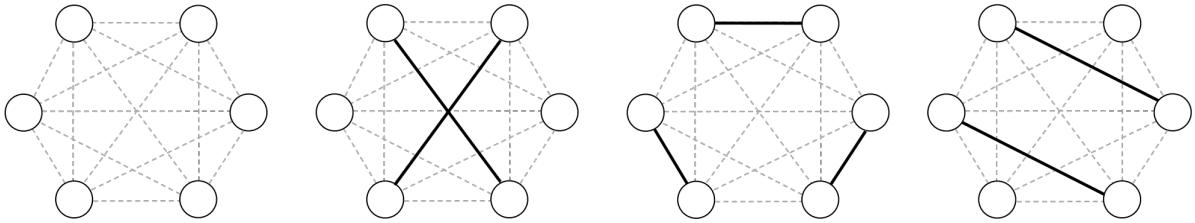
Capítulo 3. El problema de distribución máxima y homogénea de mensajes

En este capítulo se plantea una problemática presente en un escenario particular en redes oportunistas, en donde todos los participantes de la red quieren enviar un mensaje a un destino que es desconocido o no se encuentra en la red. Este tipo de problemática se ha identificado anteriormente en la literatura pero no ha sido estudiada a profundidad, a pesar de la atención que ha recibido el área de investigación en redes oportunistas.

Se propone modelar este nuevo escenario de "enrutamiento" como un problema computacional llamado problema de Distribución Máxima y Homogénea de Mensajes DMHM o MUMD por sus siglas en inglés (*Maximum Uniform Message Distribution*). En términos generales, la entrada del DMHM es un grafo dinámico con un conjunto de mensajes almacenados en sus vértices. La salida del DMHM es una distribución de copias de todos los mensajes en los vértices de la red que maximicen el número de copias de cada mensaje, sujeto a las siguientes restricciones. Primero, todos los mensajes tienen que tener el mismo número de copias en la red. Segundo, hay como máximo una copia de cada mensaje en cada vértice. El DMHM modela algunos tipos de problemáticas reales asociadas con el colapso de redes de comunicación. A continuación, se describe formalmente el DMHM, posteriormente se describe el escenario de aplicación. Finalmente, se ilustra la relación entre el DMHM y el escenario de aplicación.

3.1. Descripción formal del DMHM

En esta sección se define formalmente el DMHM como un problema computacional, i.e., como la relación entre una entrada y una salida de datos. La entrada del DMHM es un grafo dinámico (de aristas) no dirigido, definido por $\mathcal{G} = \{G_0, G_1, \ldots, G_{\phi-1}, G_\phi\}$, donde en cada grafo no dirigido, $G_j = (V, E_j)$, $|V| = n$ y $E_j \neq E_{j+1}$ para todo $j \geq 0$ (Figura 5). Adicionalmente, $|E_j| \neq 0$ para todo $j \neq 0$ (G_0 es un grafo sin aristas). Cada vértice $v \in V$ tiene un mensaje m_v y un arreglo de memoria b_v con la capacidad para almacenar κ_v mensajes.



(a) Grafo inicial G_0. **(b)** Grafo G_i. **(c)** Grafo G_j. **(d)** Grafo final G_ϕ.

Figura 5. El grafo dinámico $\mathcal{G}$ que se utiliza como entrada para el DMHM esta compuesto por un conjunto de grafos no dirigidos, los cuales representan el patrón de contactos de los vértices durante el tiempo. Esta figura ejemplifica cuatro grafos no dirigidos de un grafo dinámico de 6 vértices. a) El grafo no dirigido inicial G_0 con las aristas potenciales representadas por lineas discontinuas en gris. b) Grafo no dirigido G_i en el instante $t = i$ con las aristas activas en negro. c) Grafo no dirigido G_j en el instante $t = j$ con las aristas activas en negro. d) Grafo no dirigido final G_ϕ en el instante $t = \phi$ con las aristas activas en negro.

Durante la ranura de tiempo j, los vértices u y v se pueden comunicar si $(u, v) \in E_j$. Sea $C_{i,j}$ el número de copias del mensaje i inmediatamente después de la ranura de tiempo j, para $1 \leq i \leq n$. Sea M_j el conjunto del número de copias de todos los mensajes en los vértices de V en el grafo G_j inmediatamente después de que ocurran todas las posibles transmisiones a través de las aristas E_j, i.e., $M_j = \{C_{1,j}, C_{2,j}, \ldots, C_{n,j}\}$. Sea K el máximo número de mensajes que V puede almacenar, i.e., $K = \sum_{j=1}^{n} \kappa_j$.

Sea $\mathcal{C} = \text{mín}\{C_{i,\phi}\}$ for all $1 \leq i \leq n$, i.e., el número de copias del mensaje que tienen el mínimo número de copias en G_ϕ. La salida del DMHM es un conjunto M_ϕ que maximiza $\mathcal{C}$ sujeto a las siguientes restricciones:

1. El número de copias $C_{i,\phi}$ es el mismo para toda i.
2. Existe a lo más una copia de cada mensaje en cada vértice.

3.2. Escenario de aplicación

Como se describe en la sección 1.1, las fallas en las telecomunicaciones ocurren comúnmente durante desastres naturales y conflictos sociales. La falta de comunicación dentro del área afectada complica realizar acciones coordinadas para resolver o mitigar los efectos de estos eventos. En el caso de un desastre natural, un ejemplo de una acción coordinada sería una operación de rescate, cuyo objetivo principal es salvar vidas humanas.

Una posible solución para mitigar los problemas mencionados anteriormente es

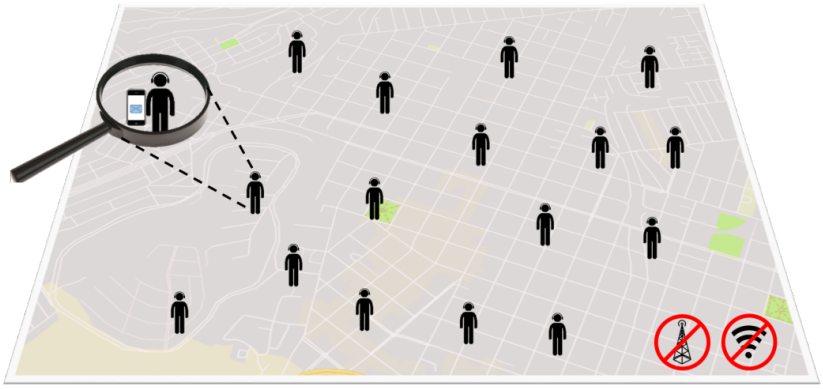
construir una red de comunicación alternativa usando dispositivos móviles. Esta red podría remplazar temporalmente las redes de comunicación tradicionales. El objetivo principal de esta red alternativa es permitir la transmisión de mensajes del área afectada al resto del mundo (i.e., a una red de comunicación global). Alcanzar este objetivo es lo que se llama en este documento una *entrega exitosa*. Un segundo objetivo más desafiante de esta red alternativa es maximizar la probabilidad de entregas exitosas para cada mensaje generado. A continuación, se proporcionan más detalles acerca del modelado de este tipo de escenarios.

Suponga que hay un grupo de personas en el área afectada sin comunicación global y que cada persona tiene un dispositivo móvil de comunicación (ver Figura 6a). Todos ellos quieren enviar un mensaje fuera del área afectada. Cualquier par de ellos pueden comunicarse de forma inalámbrica entre sí a través de sus dispositivos, si están lo suficientemente cerca (determinado por la interface de comunicación de sus dispositivos). En estos encuentros, los dispositivos intercambian copias de sus mensajes para incrementar la probabilidad de entrega exitosa. Entre mayor sea el número de copias de un mensaje, mayor es la probabilidad de entrega exitosa. Eventualmente, alguien, de alguna manera, puede salir del área afectada y acceder a una red de comunicación global con su dispositivo móvil. De esta manera, el dispositivo móvil puede entregar con éxito todos los mensajes almacenados en él.

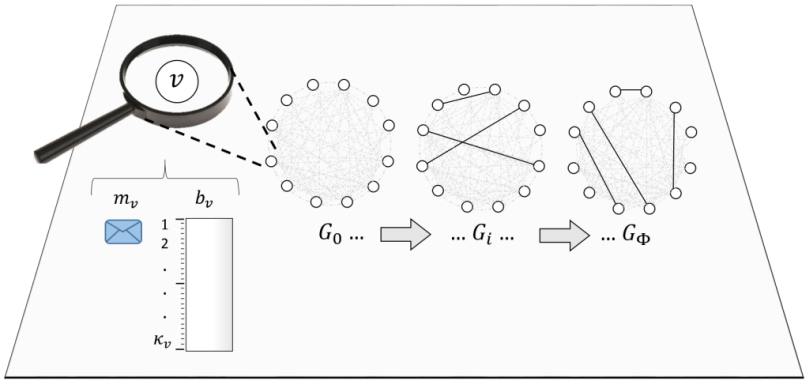
Para maximizar la probabilidad de entrega exitosa de cada mensaje, el protocolo de comunicación de esta red provisional trataría de generar tantas copias de cada mensaje como sea posible. Esta característica del protocolo es consistente con el objetivo de maximización del DMHM. Al mismo tiempo, tiene que eliminar el conflicto de interés entre los individuos. Debido a que la suma de la capacidad de almacenamiento de todos los dispositivos de la red, K, es limitada, este protocolo tiene que limitar el número de copias, L, de cada mensaje a aproximadamente $\lfloor \frac{K}{n} \rfloor$. Esta segunda característica del protocolo es congruente con la primera restricción del DMHM. Note que en este escenario, sería inútil tener más de una copia de un mensaje en un dispositivo. Por esta razón, el protocolo de la red provisional se debe encargar de esta cuestión. La tercera característica del protocolo es congruente con la segunda restricción del DMHM.

En consecuencia, es deseable un protocolo de comunicación que distribuya de ma-

nera uniforme los mensajes en toda la red y utilice el almacenamiento de los dispositivos a su máxima capacidad.



(a) Escenario de aplicación



(b) Escenario de DMHM

(c) Escenario de simulación

Figura 6. Tres diferentes niveles de abstracción del mismo escenario. (a) El escenario de aplicación del DMHM, donde hay un grupo de 16 personas que tienen dispositivos de comunicación móvil en un área sin comunicación global (como se describe en la sección 3.2). (b) Un ejemplo de entrada al problema DMHM (como se describe en la sección 3.3) que representa el patrón de todos los contactos entre los 16 vértices. (c) Un escenario de simulación en el simulador ONE utilizando un área con un mapa (como se describe en la sección 5.3.2) y 16 vértices.

3.3. Mapeando el DMHM con el escenario de aplicación

El escenario de aplicación se modela utilizando un grafo dinámico. De forma similar, otros trabajos utilizan grafos dinámicos para modelar redes dinámicas, e.g., (Clementi *et al.*, 2015; Kuhn y Oshman, 2011). Un grafo dinámico cambia su conjunto de aristas en cada instante t. Cada vértice en el grafo representa un individuo con un dispositivo móvil (ver Figura 6b). En este escenario, se supone que el número de individuos siempre es n, por lo que el número de vértices no cambia en el grafo.

Cada arista en el grafo, en el instante t, representa un contacto entre dos dispositivos móviles que pueden comunicarse (i.e., la distancia entre ellos en el escenario de aplicación es menor que el radio de comunicación dado por sus interfaces de comunicación). Cada contacto de comunicación es una consecuencia del patrón de movimientos de los individuos en el área afectada del escenario de aplicación. Por este motivo, el grafo modifica su conjunto de aristas cuando los contactos de comunicación cambian (i.e., el grafo dinámico captura el movimiento de los individuos), como se muestra en la Figura 5.

En este escenario, la capacidad de almacenar mensajes de un vértice es proporcional a la capacidad de almacenamiento del dispositivo móvil que representa. Las suposiciones del escenario de aplicación se describen a continuación:

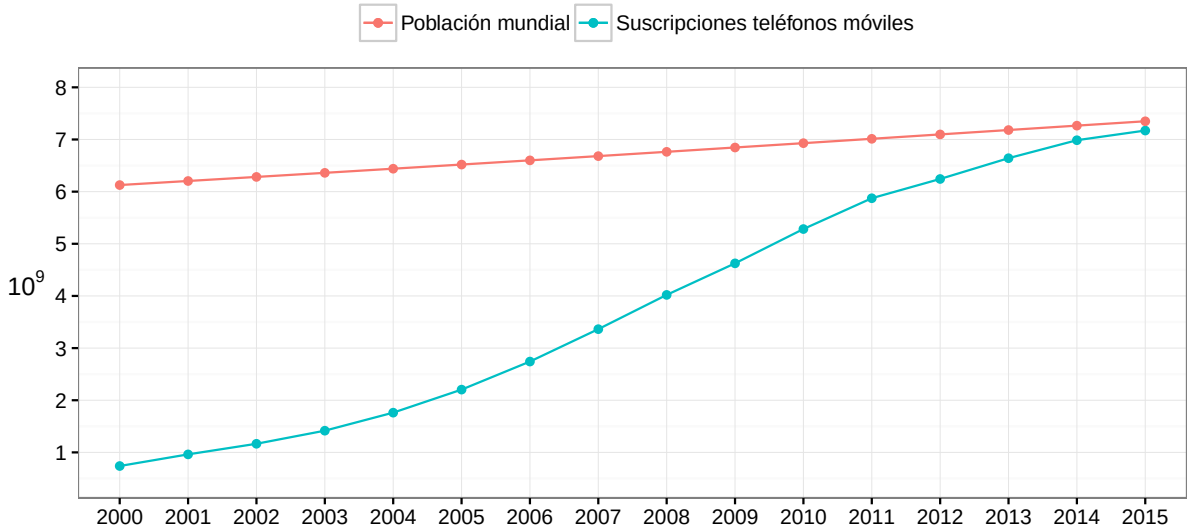


Figura 7. Comparativa del crecimiento de la población mundial y las suscripciones a teléfonos móviles durante el periodo del 2000 al 2015.

- Todos los usuarios dentro del escenario tienen un teléfono móvil (ver Figura 7)[1].
- Las comunicaciones con el exterior no funcionan o están restringidas.
- Todos los usuarios desean comunicarse con el exterior, esta suposición está asociada a la entrada del problema, ya que cada vértice quiere enviar un mensaje.
- Todos los dispositivos tienen una capacidad de almacenamiento fija, es decir no se supone que su memoria es infinita.
- Todos los dispositivos tienen una probabilidad uniforme de salir al exterior o alcanzar cobertura. No se sabe en qué lugar ni en qué momento la cobertura de una red global puede regresar, o si algún usuario puede alcanzar la suficiente altura para alcanzar una señal, es por ello que se hace está suposición.
- Cuando un dispositivo alcanza cobertura de una red global, manda todos los mensajes en su memoria, es decir, se supone que una vez que un dispositivo alcanza cobertura éstos se entregan satisfactoriamente, ya que la red global se encarga de entregar los mensajes a sus destinatarios.

Una de las principales suposiciones del DMHM es que se tiene un desconocimiento total del sistema, cada vértice solamente tiene su información local y la que puede recopilar durante los contactos subsecuentes. Por lo tanto, a continuación se establecen las suposiciones para el modelo de comunicación en el escenario de aplicación.

1. Todos los vértices tienen el mismo tipo de interface de comunicación.
2. Cada vértice tiene capacidad limitada para almacenar los mensajes.
3. Las oportunidades de transmisión son limitadas en cuanto a duración y ancho de banda (según el modelo de movilidad).
4. Los vértices no tienen control de su propio movimiento.
5. Los vértices no tienen un conocimiento a priori del tamaño de la red ni de su conectividad.
6. Los vértices no tienen un conocimiento de su localización geográfica.

[1] Esta suposición se puede justificar por medio de la Figura 7, donde se observa que en años recientes el número de suscripciones de teléfonos móviles crece mucho más rápido que el número de personas en el mundo, incluso si la tendencia continua en los próximos años superaran las suscripciones de teléfonos móviles al tamaño de la población mundial (ITU., 2016).

3.4. Resumen

En este capítulo, se describe la problemática del DMHM tomando como motivación algunos escenarios particulares donde el destino de los mensajes puede estar temporalmente no disponible. En este tipo de escenarios es necesario utilizar un algoritmo que permita distribuir el mensaje de forma homogénea en la red e intentando maximizar el número de copias de este mensaje, para maximizar la probabilidad de entrega exitosa del mensaje, i.e., cuando eventualmente alguien se encuentre con el destino entregue el mensaje al destino. En el siguiente capítulo se describe el algoritmo propuesto para esta tarea.

Capítulo 4. Algoritmo EDEN

En este capítulo, se describe el algoritmo EDEN (**E**stimar copias, **D**iseminar mensajes y **EN**mendar copias) y su mecanismo de intercambio de mensajes. Para ilustrar el proceso de intercambio se utiliza el 'árbol binario de transmisión,' un procedimiento para estimar el número de copias, y uno para actualizar el número de copias. Finalmente, se muestra el pseudocódigo del algoritmo EDEN.

El algoritmo EDEN disemina copias de los mensajes a través de la red considerando la capacidad de almacenamiento de los vértices en la red. La principal idea de este algoritmo es el de permitir un número fijo de réplicas (acotado a un número adecuado de copias) de cada mensaje de acuerdo a la memoria global de la red.

En el Apéndice A se presenta una versión preliminar de este algoritmo. El algoritmo EDEN retoma algunas ideas del algoritmo preliminar y considera las lecciones aprendidas de los experimentos preliminares al usar este algoritmo (ver sección B.3). A continuación, se explican las propiedades de cada mensaje y el mecanismo de envío de mensajes del algoritmo EDEN.

4.1. Propiedades de cada mensaje para el proceso de intercambio

Cada mensaje m_v generado por el vértice v tiene las siguientes propiedades:

- $m_v.pnc$ indica el número de copias inicialmente permitido para cada mensaje m_v. Este parámetro se define al momento de crear el mensaje.
- $m_v.sign$ es un sufijo binario común a un conjunto de copias de un mensaje utilizado en el proceso de intercambio de mensajes. A este sufijo binario se le denomina la firma (*signature*). La firma se actualiza cada vez que se transmite el mensaje.
- $m_v.updated$ bandera que indica si la variable $m_v.pnc$ se actualizó o no.
- $m_v.avIds$ es una lista ordenada de identificadores de copia disponible para el mensaje m_v. Un vértice puede generar esta lista con los valores de $m_v.sign$ y $m_v.pnc$ (ver sección 4.2).

- $m_v.id$ indica el identificador único de la copia del mensaje en formato binario.
- $m_v.anc$ indica el número de copias restantes disponibles para el mensaje $m_v.id$.

Note que el mensaje solo necesita almacenar las tres primeras propiedades descritas, ya que con éstas, un vértice puede generar fácilmente las otras. A continuación, se ilustra el proceso de intercambio de mensajes en el 'árbol binario de transmisión.'

4.2. Árbol binario de transmisión

El procedimiento para crear copias de los mensajes en el algoritmo EDEN puede considerarse como una versión modificada de la fase de dispersión en modo binario del algoritmo SnW. A continuación, se explica este proceso.

Se define el 'árbol binario de transmisión' para ilustrar gráficamente el proceso de transmisión del mensaje m_v a través del tiempo, como se muestra en la Figura 8. El *árbol binario de transmisión* (ABT) es un árbol binario que tiene a lo más (2^x-1) nodos ABT, donde x es el número de copias disponibles para el mensaje m_v.

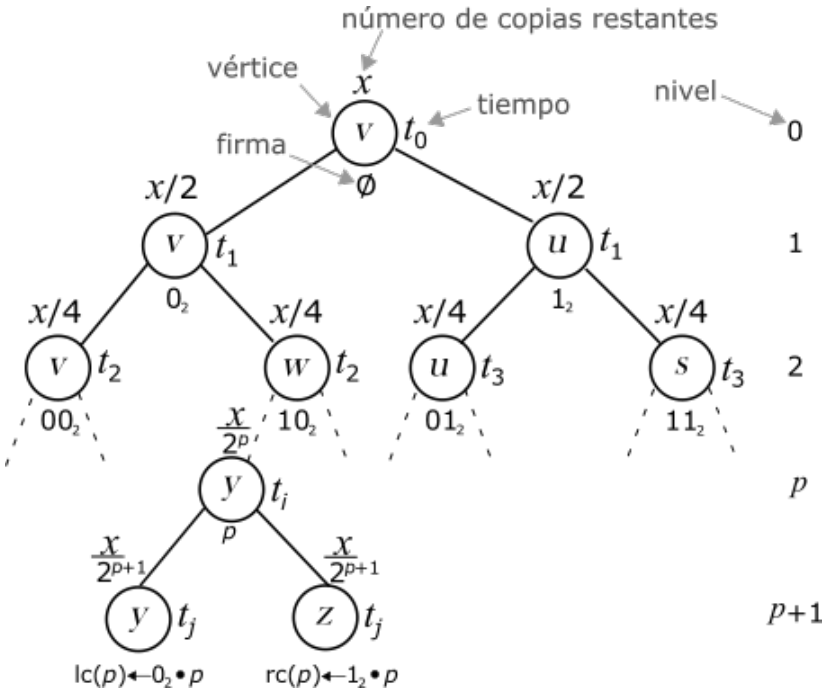


Figura 8. Ejemplo del árbol binario de transmisión utilizando el mensaje m_v.

La raíz del ABT representa la primer copia del mensaje original en el instante que se creó. Cada hijo izquierdo del ABT representa la copia del mensaje asociada al vér-

tice emisor (el padre) pero actualizando las propiedades del mensaje después de la transmisión. Alternativamente, cada hijo derecho representa una copia del mensaje en el vértice del receptor con las propiedades del mensaje actualizadas. Cada nodo ABT tiene un identificador único especificado por su firma. A continuación se explica las características y funcionamiento de la firma de cada mensaje.

Dado el alfabeto binario $\Sigma = \{0, 1\}$, Σ^n denota el conjunto de todas las cadenas binarias sobre Σ con longitud de n bits. Por ejemplo, $\Sigma^2 = \{00, 01, 10, 11\}$. Note, en la Figura 8, que Σ^i es el conjunto de firmas de todos los nodos ABT en el nivel i. El nivel máximo de un ABT de tamaño x es $\log x$. La ecuación 1 calcula el conjunto de firmas para los nodos ABT de un ABT de tamaño x usando la estrella de Kleene (Hopcroft y Ullman, 1979) sobre Σ.

$$\Sigma^* = \bigcup_{i=0}^{\log x} \Sigma^i \tag{1}$$

donde $\Sigma^i = \{0, 1\}^i$. Cada nodo ABT representa una copia del mensaje m_v. Note, en la Figura 8, que cada nodo ABT contiene varios atributos, los cuales se explican a continuación. La etiqueta dentro del nodo ABT representa el identificador del vértice que almacena la copia del mensaje m_v. El número encima del nodo ABT es el número de copias que esta copia puede generar. El número binario debajo de cada nodo ABT representa la firma de la copia física que se almacena. El nivel de un nodo es el número de transmisiones que se han realizado de esa copia específica. El número en la derecha de cada nodo ABT indica el instante de tiempo en el cual el protocolo genera/actualiza la firma de la copia. A continuación, se explica cómo se genera el ABT para un mensaje m_v.

Suponga que ninguno de los siguientes vértices que se contactan tienen el mensaje m_v y que tienen suficiente capacidad de almacenamiento disponible para guardar el mensaje m_v.

En general, sea ρ la firma de un nodo ABT interno. Sean las firmas de su hijo izquierdo e hijo derecho $lc(\rho)$ y $rc(\rho)$, respectivamente. Entonces, $lc(\rho) \leftarrow 0_2 \bullet \rho$ y $rc(\rho) \leftarrow 1_2 \bullet \rho$, donde el símbolo '$\bullet$' indica la concatenación de dos cadenas.

Ejemplo 1: Suponga que x es una potencia de dos y que en el instante de tiempo $t = t_0$, el vértice v crea la copia inicial del mensaje m_v, con $m_v.pnc = x$ y $m_v.sign = \emptyset$ (la raíz del ABT siempre tiene una firma $\emptyset$; ver la Figura 8).

Suponga que en la red, el vértice v contacta al vértice u, y el vértice v transmite el mensaje m_v al vértice u. Entonces, en el ABT, el nodo ABT con la firma $\emptyset$ genera dos nuevos nodos ABT, $lc(\emptyset)$ y $rc(\emptyset)$, con las firmas $lc(\emptyset) \leftarrow 0_2 \bullet \emptyset = 0_2$ y $rc(\emptyset) \leftarrow 1_2 \bullet \emptyset = 1_2$, ambas generadas en el instante de tiempo $t = t_1$. En la Figura 8, los dos nodos ABT en el nivel 1 representan estas dos copias almacenadas en los vértices v y u.

De manera similar, suponga que en la red, el vértice v contacta al vértice w, y el vértice v transmite el mensaje m_v al vértice w. Entonces, el nodo ABT con la firma 0_2 genera dos nuevos nodos ABT, $lc(0_2)$ y $rc(0_2)$, con las firmas $lc(0_2) \leftarrow 0_2 \bullet 0_2 = 00_2$ y $rc(0_2) \leftarrow 1_2 \bullet 0_2 = 10_2$, ambas generadas en el instante de tiempo $t = t_2$. El proceso de construcción del ABT finaliza cuando genera x hojas (i.e., después de $x-1$ transmisiones del mensaje m_v).

En general, cualquier vértice con firma ρ se le permite generar todas las copias del mensaje m_v cuyos identificadores, entre 0 y $x-1$, finalicen con el sufijo ρ. Específicamente, después de generar los nodos ABT $lc(\rho)$ y $rc(\rho)$, a cada uno de ellos se le permite generar todas las copias cuyos identificadores están entre 0 y $x-1$, y que finalicen con el sufijo $0_2 \bullet \rho$ y $1_2 \bullet \rho$, respectivamente. En la red, el emisor conserva la mitad de las copias disponibles, y el receptor conserva la otra mitad.

Inicialmente, el número de copias permitidas ($m_v.pnc$) es igual al número de copias disponibles ($m_v.anc$). El número de copias disponibles decrece a la mitad en cada transmisión del mensaje, i.e., $m_v.anc = m_v.pnc/2^{length(\rho)}$, donde $length(\rho)$ denota el número de bits (número de transmisiones del mensaje) en ρ. El número de copias permitidas ($m_v.pnc$) mantiene el mismo valor en cada transmisión del mensaje. Sólo cambia cuando el estimador converge (ver sección 4.4). En la siguiente sección se explica cómo se calcula el número de copias permitidas x utilizadas en el ABT y su estimador utilizado durante el proceso de intercambio de mensajes.

4.3. Estimador del número óptimo de copias del algoritmo EDEN

El algoritmo SnW utiliza un valor fijo L para todos los vértices para limitar el número de copias de cada mensaje. Por el contrario, el algoritmo EDEN utiliza, para cada vértice v, un parámetro $l_v(t)$ que funciona como un estimador local (dependiente del tiempo) del número óptimo de copias requerido, L'. La Figura 9 muestra un diagrama general que presenta la función del estimador en el algoritmo EDEN.

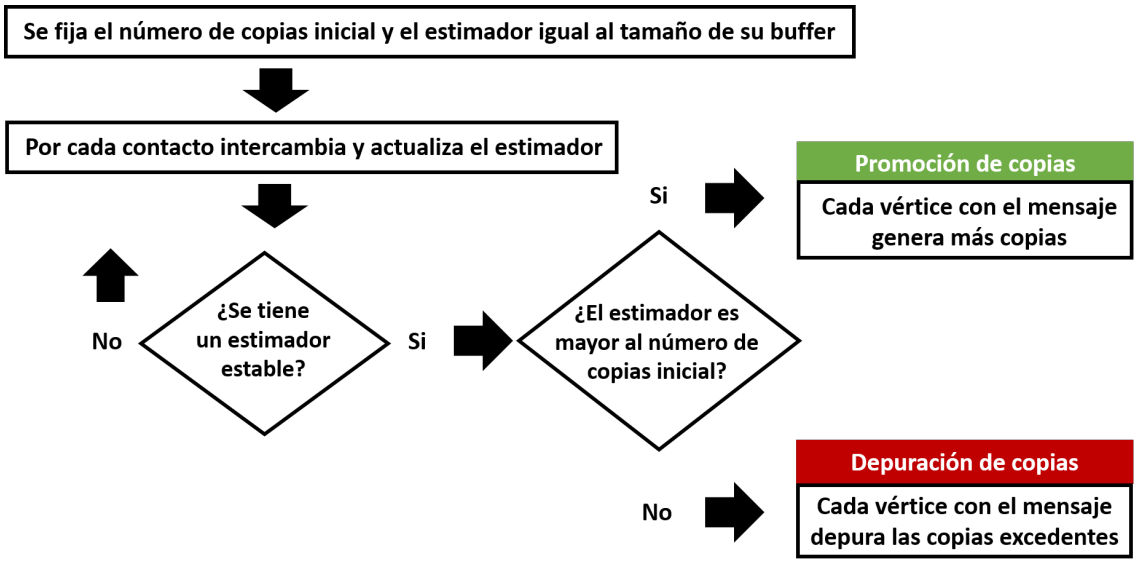


Figura 9. Diagrama de flujo del uso del estimador en el algoritmo EDEN.

Específicamente, en el momento $t = 0$, el vértice v fija su estimador local $l_v(0) = \kappa_v$. El algoritmo EDEN actualiza regularmente $l_v(t)$ durante la ejecución del algoritmo usando la información recopilada y que se almacena en los estimadores locales de los vértices contactados. Este algoritmo ajusta gradualmente cada $l_v(t)$ hasta que se acerca a L', es decir, conforme el tiempo $t \to \infty$, $l_v(\infty) \to L'$. Algunos trabajos anteriores usan un enfoque similar llamado promediado distribuido (Xiao y Boyd, 2004; Rajagopalan y Shah, 2011).

Cuando un contacto ocurre entre los vértices v y u en el tiempo $t = t_k$, estos vértices actualizan sus estimadores locales de la siguiente manera:
$l_v(t_k) = l_u(t_k) = \big(l_v(t_{k-\alpha}) + l_u(t_{k-\beta})\big)/2$, donde $l_v(t_{k-\alpha})$ y $l_u(t_{k-\beta})$ son los valores anteriores de los estimadores locales de v y u, respectivamente.

Cada vértice utiliza un arreglo llamado *historic* para almacenar los *historic.size*

valores más recientes de su estimador local. Se dice que el estimador local es *estable* si la diferencia entre el valor actual del estimador local y cada uno de los elementos en el arreglo *historic* es menor que un valor ϵ predefinido. La operación de este arreglo se explica en la sección 4.5.1.

Suponga que en el tiempo $t = t_\sigma$, el estimador $l_v(t_\sigma)$ es estable. Entonces, en el instante de tiempo $t = t_\sigma$, el vértice v elimina de su memoria cada mensaje cuyo número de copia es mayor que $l_v(t_\sigma)$. Simultáneamente, el vértice v incentiva la creación de nuevas copias de cada mensaje cuyo número de copia es menor que $l_v(t_\sigma)$. A continuación, se explican estos dos mecanismos.

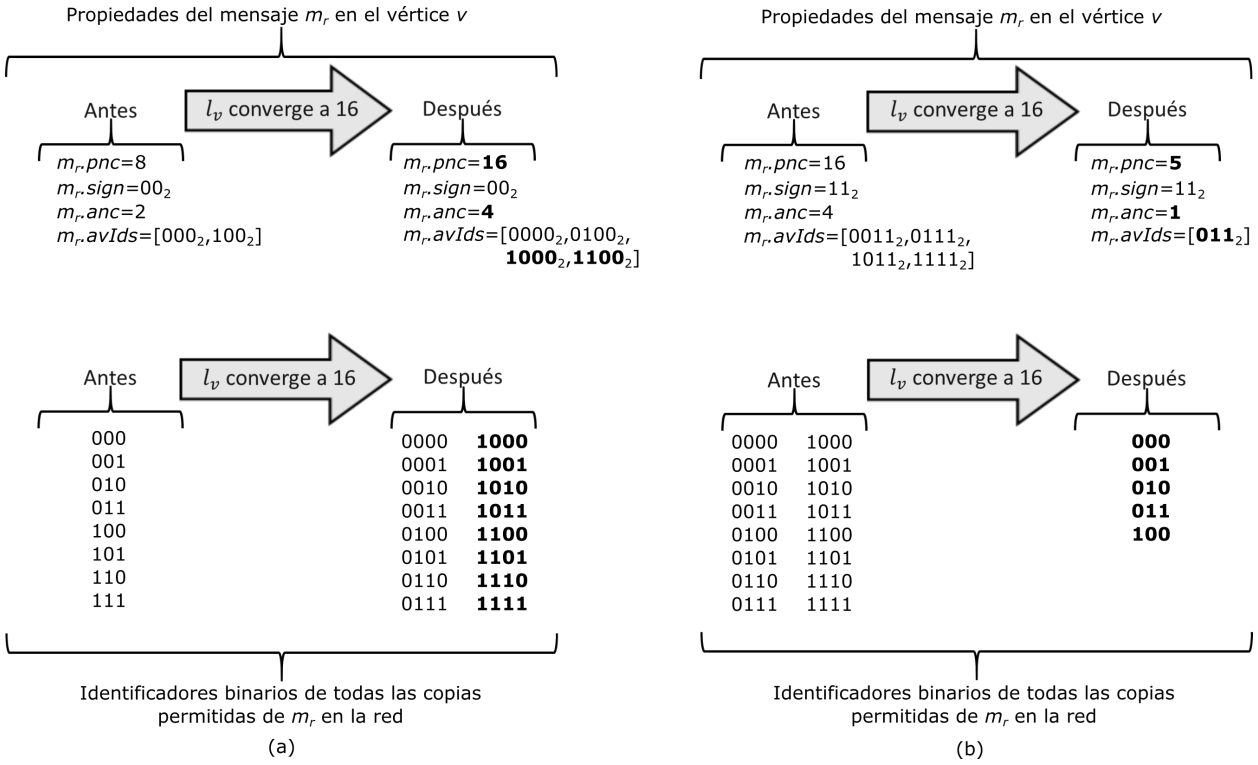


Figura 10. Ejemplos de las propiedades y los identificadores binarios del mensaje m_r. Los números en negritas son los valores actualizados. (a) Promoción de copias. (b) Depuración de copias.

4.4. Actualizando el número de copias permitido

Cuando un vértice v detecta que su estimador $l_v(t)$ es estable, v compara $l_v(t)$ contra $m_r.pnc$ para todos los mensajes m_r almacenados en v. Cuando difieren, el vértice v realiza los siguientes mecanismos por cada mensaje en su memoria. De ahora en adelante, por simplicidad, se denota a $l_v(t)$ como l_v.

4.4.1. Promoción de copias

Si $m_r.pnc < l_v$, entonces el mensaje m_r requiere más copias. Para lograr esto, v reemplaza $m_r.pnc$ con el valor de l_v. Para el algoritmo, este cambio es transparente (es decir, no necesita realizar ninguna acción especial para generar las copias adicionales para m_r).

Ejemplo 2: Suponga que un vértice v contiene un mensaje m_r con un número permitido de copias igual a 8 (es decir, $m_r.pnc = 8$) y $m_r.sign = 00_2$. Suponga también que v detecta que $l_v = 16$ es estable. Con estos valores, el vértice v calcula $m_r.anc = 2$ y $m_r.avIds = [000_2, 100_2]$; i.e., todos los identificadores binarios de 000_2 a 111_2 con sufijo 00_2 (ver la parte izquierda de la Figura 10a). Actualizar el valor de $m_r.pnc$ de 8 a 16 significa que ahora debe haber 16 copias del mensaje m_r con identificadores de 0000_2 a 1111_2. Note que de estos 16 identificadores, hay 4 con sufijo 00_2. Por lo tanto, resulta en más copias disponibles para m_r en el vértice v, i.e., $m_r.rnc = 4$ y $m_r.avIds = [0000_2, 0100_2, 1000_2, 1100_2]$ (ver la parte derecha de la Figura 10a).

4.4.2. Depuración de copias

Si $m_r.pnc > l_v$, entonces se permitió un número excesivo de copias del mensaje m_r. Para corregir este exceso, el algoritmo ejecuta uno de los dos casos siguientes para cada mensaje. De forma similar a la promoción de copias, el primer caso sólo actualiza el número permitido de copias, mientras que en el segundo caso detecta que la copia del mensaje es uno de los excedentes y luego la elimina.

- Caso *a*: Si $(l_v - 1)_2 \geq m_r.sign$, entonces el vértice v reemplaza $m_r.pnc$ con el valor de l_v y permite que el procedimiento de intercambio de mensajes continue de forma normal.
- Caso *b*: Si $(l_v - 1)_2 < m_r.sign$, entonces el vértice v elimina el mensaje m_r.

Ejemplo 3: Suponga que el vértice v contiene una copia del mensaje m_r con un número permitido de copias igual a 16 (es decir, $m_r.pnc = 16$) y $m_r.sign = 11_2$. Suponga también que v detecta que $l_v = 5$ es estable. Entonces, el vértice v ejecuta

el Caso *a* del mecanismo de depuración de copias. Antes de la ejecución del procedimiento de depuración de copias, $m_r.avIds = [0011_2, 0111_2, 1011_2, 1111_2]$; i.e., todos los identificadores binarios de 0000_2 a 1111_2 con sufijo 11_2 (ver la parte izquierda de la Figura 10b). Después de ejecutar el procedimiento de depuración de copias $m_r.avIds = [0011_2]$; i.e., todos los identificadores binarios de 0000_2 a 0011_2 con sufijo 11_2 (ver la parte derecha de la Figura 10b).

Ejemplo 4: Suponga que el vértice v contiene una copia del mensaje m_r con un número permitido de copias igual a 8 (es decir, $m_r.pnc = 8$) y $m_r.sign = 111_2$. Suponga también que v detecta que $l_v = 5$ es estable. Entonces, el vértice v ejecuta el Caso *b* del mecanismo de depuración de copias. Por lo tanto, el vértice v elimina el mensaje m_r de su memoria.

4.5. Pseudocódigo del algoritmo EDEN

El pseudocódigo 1 describe el procedimiento EDEN para el vértice v. Cada vértice v en la red ejecuta localmente este procedimiento. Dicho procedimiento recibe como entrada dos enteros, *historic.size* y ϵ, los cuales se utilizan para evaluar la convergencia del estimador. A continuación, se explica el pseudocódigo 1.

El paso 1 fija el estimador l_v igual al valor de la capacidad de almacenamiento del vértice v. El paso 2 fija la bandera Booleana *stable* a falso para indicar que l_v no es estable. El paso 3 fija el arreglo *historic* que contiene los *historic.size* valores más recientes de l_v. El paso 4 fija *counter* que indica el número de inserciones en *historic*. En el paso 5, el vértice v espera hasta que detecta la presencia de algún vértice u en su rango de comunicación. Cuando v detecta tal vértice u, v verifica que u no este ocupado transmitiendo a otro vértice. Si u está disponible, entonces el proceso de intercambio de datos inicia entre estos vértices (Pasos 6 al 14). En el paso 6, v actualiza su estimador local (ver el pseudocódigo 2). En el paso 7, v actualiza los mensajes en su memoria (ver el pseudocódigo 3). En el paso 8, v genera su 'resumen de mensajes disponibles' (ver el pseudocódigo 4). En los pasos 9 and 10, v transmite su resumen a u y recibe el resumen de u, respectivamente. En el paso 11, v calcula su 'resumen de mensajes requeridos' (ver el pseudocódigo 5). En los pasos 12 and 13, v transmite el resumen a u y recibe el resumen de u, respectivamente. En el paso 14, v transmite todos los mensajes requeridos por u y recibe todos los mensajes que

solicitó a u (ver el pseudocódigo 6). Después, v continua la ejecución en el paso 5 del pseudocódigo 1.

Note que el vértice con el identificador más pequeño toma inicialmente el control del canal de comunicación. En esta explicación, se supone que el identificador del vértice v es más pequeño que el del vértice u.

Pseudocódigo 1 EDEN

Entrada: *historic.size* $\in \mathbb{N}$ y una constante positiva ϵ para determinar la convergencia del estimador.

1: $l_v \leftarrow \kappa_v$
2: $stable \leftarrow false$
3: $historic[0] \leftarrow l_v$
4: $counter \leftarrow 1$
5: **while** $\exists\, u$ en el rango de comunicación con v **do**
6: Update-Estimator()
7: Update-Messages()
8: $s_v(2) \leftarrow$ Create-Available-Summary(2)
9: Transmit($s_v(2)$)
10: $s_u(2) \leftarrow$ Receive()
11: $r'_{v,u} \leftarrow$ Create-Requested-Summary($s_u(2)$)
12: Transmit($r'_{v,u}$)
13: $r'_{u,v} \leftarrow$ Receive()
14: Exchange-Messages($r'_{v,u}\ r'_{u,v}$)
15: **end while**

4.5.1. Procedimiento Update-Estimator

En el procedimiento Update-Estimator, el vértice v evalúa si l_v es estable (ya convergió) como se describe en la sección 4.3. Se describe este procedimiento en el pseudocódigo 2.

El algoritmo EDEN utiliza el arreglo *historic* como una cola circular para almacenar los *historic.size* valores más recientes de l_v. El arreglo calcula el valor (*counter* mod *historic.size*) para determinar la posición en el arreglo donde el siguiente valor de l_v se va a almacenar. A continuación, se explica el Pseudocódigo 2.

El paso 1 evalúa si l_v no es estable. Si l_v ya es estable, entonces este paso intercambia los estimadores y termina la ejecución; De lo contrario, continua en el paso 2. En los pasos 2 y 3, v transmite su estimador local l_v al vértice u y recibe el estimador l_u del vértice u, respectivamente. El paso 4 reemplaza el estimador l_v. El paso 5 calcula si l_v cumple el criterio de estabilidad. Si no es estable, entonces el procedimiento

agrega su valor en el arreglo *historic*, actualiza el valor de la variable *counter*, y termina la ejecución (pasos 8 y 9). Si l_v es estable, entonces el paso 6 cambia *stable* a verdadero.

Pseudocódigo 2 Update-Estimator

```
1: if stable = false then
2:     Transmit(l_v)
3:     l_u ← Receive()
4:     l_v ← (l_v + l_u)/2
5:     if counter ≥ historic.size and |l_v − historic[i]| ≤ ϵ, ∀i ∈ [0, historic.size − 1] then
6:         stable ← true
7:     else
8:         historic[counter mod historic.size] ← l_v
9:         counter ← counter + 1
10:    end if
11: end if
```

4.5.2. Procedimiento Update-Messages

En el procedimiento Update-Messages, el vértice v actualiza los mensajes en su memoria de acuerdo con el procedimiento descrito en la sección 4.4. El pseudocódigo 3 describe este procedimiento.

El paso 1 evalúa si l_v es estable. Si l_v ya es estable, continua con el paso 2. De lo contrario, sale del procedimiento. Los pasos del 2 al 9 evalúan y actualizan el número de copias disponibles de cada uno de los mensajes en la memoria de v, como se explica en la sección 4.4, y sale del procedimiento. El número de copias debe ser un valor entero; por lo tanto, en el caso de que l_v converja a un valor no entero, el paso 6 redondea este valor a través de un redondeo estocástico, e.g., si la parte fraccionaria de l_v es 0.6, selecciona aleatoriamente entre $l_v + 0.6$ y $l_v - 0.4$, con probabilidades de 0.6 y 0.4, respectivamente.

Pseudocódigo 3 Update-Messages

```
1: if stable = true then
2:     for all mensaje m_r en b_v do
3:         if m_r.pnc > l_v and m_r.sign > (l_v − 1)_2 then
4:             elimina el mensaje m_r de b_v
5:         else if m_r.updated = false then
6:             m_r.pnc ← stochastic_round(l_v)
7:             m_r.updated ← true
8:         end if
9:     end for
10: end if
```

4.5.3. Procedimiento Create-Available-Summary

El procedimiento Create-Available-Summary recibe como entrada un entero x y genera como salida un resumen de los mensajes en el vértice v, llamado $s_v(x)$. El pseudocódigo 4 describe este procedimiento.

Sea $b_v.size$ el número de mensajes almacenados en b_v. Sea $s_v(x)$ el conjunto de identificadores de los mensajes que tienen al menos x copias disponibles en el vértice v, e.g., $s_v(1)$ es el conjunto de identificadores de todos los mensajes en b_v. Note que $b_v.size = |s_v(1)|$. Se asigna el nombre de *resumen de mensajes disponibles* al conjunto $s_v(2)$, i.e., el conjunto de identificadores que tienen al menos dos copias disponibles (uno para guardar y otro para enviar). Note que $s_v(2) \subseteq s_v(1)$.

El paso 1 inicializa el resumen, y el paso 2 verifica si hay al menos un mensaje en la memoria de v; si es así, los pasos del 3 al 7 evalúan el número de las copias restantes permitidas ($m_r.anc$) de cada mensaje m_r en b_v. Si $m_r.anc$ es al menos x, entonces el procedimiento agrega m_r al resumen. Finalmente, el paso 9 regresa el resumen solicitado.

Pseudocódigo 4 Create-Available-Summary

Entrada: Un valor $x \in \mathbb{N}$, que representa el número mínimo de copias restantes de cada mensaje del resumen a generar.

```
1: s_v(x) ← ∅
2: if b_v.size > 0 then
3:     for all mensaje m_r en b_v do
4:         if m_r.anc ≥ x then
5:             s_v(x) ← s_v(x) ∪ m_r.id
6:         end if
7:     end for
8: end if
9: return s_v(x)
```

4.5.4. Procedimiento Create-Requested-Summary

En el procedimiento Create-Requested-Summary, el vértice v calcula $r_{v,u}$, el *resumen de mensajes requeridos* de v, es decir, el conjunto de identificadores de los mensajes que v solicita de u. Note que $r_{v,u} \subseteq s_u(2) \setminus s_v(1)$. Sea $r_{v,u}$ a lo más $|\kappa_v - b_v.size|$ identificadores seleccionados aleatoriamente del conjunto $\{s_u(2) \setminus s_v(1)\}$.

Este procedimiento recibe como entrada el conjunto $s_u(2)$. El paso 1 calcula el

conjunto $s_v(1)$ con el pseudocódigo 4. Este conjunto contiene los identificadores de todos los mensajes en la memoria de v. El paso 2 obtiene $r_{v,u}$, el conjunto de los identificadores de todos los mensajes disponibles en u que v no tiene. Los pasos del 3 al 6 seleccionan aleatoriamente del conjunto $\{s_u(2) \setminus s_v(1)\}$ a los identificadores de los mensajes que b_v puede almacenar. Finalmente, el paso 7 regresa el resumen de mensajes requeridos generado.

Pseudocódigo 5 Create-Requested-Summary

Entrada: El conjunto $s_u(2)$, que representa el resumen de mensajes disponibles del otro vértice.

1: $s_v(1) \leftarrow$ Create-Available-Summary(1)
2: $r_{v,u} \leftarrow s_u(2) \setminus s_v(1)$
3: **if** $|\kappa_v - b_v.size| < |r_{v,u}|$ **then**
4: $\quad \alpha \leftarrow (|r_{v,u}| - |\kappa_v - b_v.size|)$
5: $\quad$ Aleatoriamente elimina α elementos de $r_{v,u}$
6: **end if**
7: **return** $r_{v,u}$

4.5.5. Procedimiento Exchange-Messages

El procedimiento Exchange-Messages intercambia aquellos mensajes especificados por $r_{v,u}$ y $r_{u,v}$ entre v y u. El pseudocódigo 6 describe este procedimiento.

Cuando el vértice v envía (recibe) un mensaje m_r a (desde) u, actualiza la firma $m_r.sign$ de m_r. Para actualizar $m_r.sign$, el vértice v concatena un bit cero (bit uno) a un lado del bit más significativo de $m_r.sign$. La sección 4.2 describe este procedimiento de actualización.

Pseudocódigo 6 Exchange-Messages

Entrada: Un par de conjuntos $r_{v,u}, r_{u,v}$, representando los mensajes requeridos para cada vértice.

1: **if** $|r_{u,v}| > 0$ **then**
2: $\quad$ **for all** $m_r.id$ in $r_{u,v}$ **do**
3: $\quad\quad$ Transmit(m_r)
4: $\quad\quad$ $m_r.sign \leftarrow 0 \bullet m_r.sign$
5: $\quad$ **end for**
6: **end if**
7: **if** $|r_{v,u}| > 0$ **then**
8: $\quad$ **for all** $m_r.id$ in $r_{v,u}$ **do**
9: $\quad\quad$ $m_r \leftarrow$ Receive()
10: $\quad\quad$ $m_r.sign \leftarrow 1 \bullet m_r.sign$
11: $\quad\quad$ guarda el mensaje m_r en b_v
12: $\quad$ **end for**
13: **end if**

4.6. Algoritmo EDEN con calentamiento

El algoritmo *EDEN con calentamiento* (EDEN_c) es una versión simplificada de EDEN. Durante la primera etapa (*calentamiento*), cada vértice v solo intercambia su estimador y se le permite recibir mensajes. El vértice v termina esta etapa cuando el estimador converge. Durante la segunda etapa, cada vértice v empieza el procedimiento de intercambio de mensajes como en EDEN, pero ignora los procedimientos de actualización del estimador, promoción de copias, y depuración de copias. Note que EDEN ejecuta estas dos etapas simultáneamente.

La razón principal para introducir EDEN_c fue para analizar qué estrategia tiene mejor desempeño:

- Estrategia EDEN: empezar la transmisión de mensajes con un estimador que probablemente esté lejos de L' y requiriendo un mecanismo adicional para corregir el número de mensajes generados.
- Estrategia EDEN_c: empezar la transmisión de mensajes después de calcular un estimador muy cercano a L' y sin requerir ninguna corrección a posteriori en el número de mensajes generados.

El pseudocodigo para EDEN_c es el mismo que el de EDEN excepto por los siguientes cambios menores en los pseudocódigos 1 y 6. En el pseudocódigo 1, se inserta después del paso 6 la validación "**if** $stable = false$" y se inserta la línea "**end if**" justo después del paso 7. Adicionalmente, en el pseudocódigo 6, se inserta después del paso 1 la validación "**if** $stable = true$" y se inserta la línea "**end if**" justo después del paso 6.

4.7. Resumen

En este capítulo, se introduce el algoritmo EDEN que genera soluciones aproximadas al DMHM. Se describen las propiedades requeridas que tienen los mensajes para realizar el proceso de intercambio de mensajes. Para ilustrar este proceso, se define el árbol binario de transmisión. Se describe el estimador y su funcionalidad en el algoritmo EDEN.

Después, se presentan un conjunto de pseudocódigos que conforman el algoritmo EDEN. Finalmente, se describe una variante del algoritmo EDEN junto con la estrategia que utiliza.

Capítulo 5. Diseño experimental

En este capítulo, se presenta el diseño experimental utilizado para comparar el desempeño de los algoritmos utilizados para generar soluciones candidatas a el problema DMHM. En el Apéndice B se presentan múltiples experimentos preliminares que fueron de utilidad para definir el diseño experimental presentado en este capítulo. Este diseño experimental se aplica para todos los experimentos presentados en esta tesis.

5.1. Métricas para la evaluación del desempeño

En esta sección, se introduce un conjunto de métricas para medir la calidad en la distribución de mensajes en un escenario. Los protocolos de redes oportunistas se enfocan en mejorar distintas métricas de desempeño, tales como la tasa de transferencia efectiva (en inglés *througput*), tasa de entrega, uso de almacenamiento, tiempo de batería, dispersión de información (Clementi *et al.*, 2015), entre otros. Sin embargo, la mayoría de estas métricas pierden su significado o son inadecuadas en el contexto del DMHM. Por ejemplo, las métricas más relevantes son la tasa de entrega y el retardo de red, pero éstas no pueden calcularse sin un destino. Por esta razón, se introduce un conjunto de métricas que son adecuadas para medir la calidad de las soluciones generadas por los algoritmos que producen como salida soluciones aproximadas para el DMHM.

En la sección B.3.1.1 se muestran algunas de las métricas que inicialmente se consideraron para ayudar a evaluar diferentes soluciones para el problema DHMH. Finalmente, se seleccionaron las métricas cuyos objetivos se mencionan a continuación. Primero, evaluar la uniformidad en el número de copias de cada mensaje. Segundo, medir el uso de memoria de todos los vértices de la red. Finalmente, calcular el número de vértices que se deben seleccionar para obtener al menos una copia de cada mensaje. A continuación, se describen estas métricas.

5.1.1. Factor de uniformidad relativo

El *Factor de Uniformidad Relativo* (FUR) evalúa la uniformidad en el número de copias de los mensajes en la red. El FUR es igual al producto de los números de copias

de todos los mensajes divididos por el promedio de estos números, como se muestra en la ecuación 2.

$$\text{FUR} = \prod_{i=1}^{n} \frac{C_i}{\bar{C}}, \tag{2}$$

donde C_i es el número de copias del mensaje i en la red y $\bar{C}$ es dado por la ecuación 3.

$$\bar{C} = \sum_{i=1}^{n} \frac{C_i}{n} \tag{3}$$

Una propiedad de esta métrica es que cuanto mayor sea la uniformidad del número de copias, mayor será el valor del RUF. El RUF obtiene su valor máximo cuando los números de copias de todos los mensajes son iguales. En tal caso, el valor del RUF es igual a uno. Cuando el número de copias de un mensaje es cero, el valor del RUF es cero, el cuál es su valor mínimo posible.

5.1.2. Porcentaje de utilización del buffer

El *Porcentaje de Utilización del Buffer* (PUB) mide la ocupación de los mensajes en la memoria de todos los vértices de la red. El PUB es igual a la suma del número total de mensajes en la red dividida por la capacidad máxima de almacenamiento K en la red, como se muestra en la ecuación 4.

$$\text{PUB} = \sum_{i=1}^{n} \frac{C_i}{K} \tag{4}$$

El valor del PUB toma valores entre cero y uno. Esta métrica es proporcional al porcentaje de la capacidad de almacenamiento de la red que se ha utilizado para almacenar mensajes.

5.1.3. Tamaño de muestra necesario para recuperar la colección de mensajes

Esta métrica indica el número promedio λ de vértices que es necesario muestrear de la red para obtener Ω_n. Donde Ω_n es el conjunto de todos los mensajes en la red, es decir, la colección de todos los mensajes distintos en la red. Para el cálculo de esta métrica se supone una selección aleatoria sin reemplazo y con una distribución uniforme. También se supone que cada vértice seleccionado transmite todos los mensajes en su memoria a sus destinos.

En el escenario de aplicación, los vértices seleccionados representan aquellos dispositivos móviles que pudieron salir del área afectada y se conectaron a una red de comunicación global. El parámetro λ puede tomar cualquier valor entre $\{2, \ldots, n\}$ ya que se supone que $\kappa_v < n$ para todo v. Note que entre menor sea el valor de λ, mejor.

A partir de esta formulación, surgen preguntas interesantes como las siguientes: primero, ¿cuál es el número promedio de vértices que se deben seleccionar para obtener $\alpha \cdot |\Omega_n|$ mensajes diferentes para una constante $0 < \alpha < 1$? Segundo, ¿cuál es el valor de λ si se supone una distribución uniforme óptima de los mensajes en la red? Observe que este es el mejor valor que, en promedio, un algoritmo para el DMHM puede obtener. Dada la salida del DMHM M_ϕ e identificando en qué vértice está almacenada cada copia, se mide experimentalmente λ.

El contexto de esta métrica es similar a una variación del problema del colector de cupones con extracciones por grupo (Stadje, 1990; Ferrante y Saltalamacchia, 2014).

En las gráficas de las Figuras 18, 21, 23-26, y 28 se utiliza λ/n en lugar de λ, ya que es más fácil comparar los resultados de los experimentos para diferentes valores de n. Se denomina a la razón λ/n: *Promedio Normalizado del número de extracciones de Vértices* (PNV).

Vale la pena señalar que las métricas FUR y PUB son buenos indicadores de la uniformidad de copia de los mensajes y la utilización de la memoria, respectivamente; Sin embargo, por separado no se correlacionan con la calidad de las soluciones candidatas para el DMHM. El producto de la FUR y PUB (denotado por FUR×PUB) combina el comportamiento de estas dos métricas en una sola. Este producto y λ se correla-

cionan con la calidad de las soluciones candidatas para el DMHM. Además, para los escenarios prácticos mencionados en la sección 3.2, entre estas dos métricas, λ es probablemente la métrica más adecuada para el DMHM.

5.1.4. Tiempo de convergencia de la red

El *tiempo de saturación* es el instante de tiempo cuando la ocupación de almacenamiento en la red es máxima, i.e., $PUB = 1$. Este instante sólo depende del número de contactos y del tamaño de almacenamiento de cada dispositivo en la red.

Similarmente, el *tiempo de convergencia* de la red es el instante de tiempo cuando la ocupación es máxima y, para cada vértice $v \in V$, el contenido de b_v no cambia. El tiempo de convergencia depende del algoritmo de enrutamiento que se utilice. Este último es el que se utiliza al presentar los resultados experimentales.

El tiempo de convergencia es una de las métricas más importantes para el escenario de aplicación. Las víctimas atrapadas en una zona de desastre tiene una mayor oportunidad de supervivencia si son rescatadas en las primeras 72 horas, por lo que el tiempo es crucial en este tipo de escenarios. La respuesta de los cuerpos de rescate debe empezar durante este periodo crucial e intentar rescatar el mayor número de personas posibles utilizando las tecnologías de comunicación disponibles. Por lo tanto, es deseable que el tiempo de convergencia de la red sea lo más pequeño posible.

5.2. Simulador de redes

Se realizaron los experimentos utilizando el simulador ONE (en inglés *the Opportunistic Network Environment*) (Keränen *et al.*, 2009) version 1.5.1 RC2. Se eligió este simulador porque es un software de código abierto, comúnmente utilizado en investigación con DTNs (Martín-Campillo *et al.*, 2013; Sghaier *et al.*, 2014; Ababou *et al.*, 2015; Kaviani *et al.*, 2016; Amah *et al.*, 2017; Asuquo *et al.*, 2018), y tiene implementados algoritmos de enrutamiento para DTNs del estado del arte.

El propósito de los experimentos computacionales fue comparar diferentes protocolos de ruteo usando las métricas FUR, PUB, y PNV. El simulador ONE permite la representación del escenario asociado con el DMHM y la implementación de los algoritmos propuestos. Para los experimentos descritos en esta sección, se utilizaron los

Tabla 1. Parámetros de los experimentos para los escenarios A, B, C, D, y E.

Parámetro	**Escenarios**								
	A	**B**	**C**	**D**			**E**		
Número de vértices (n)	$16, 32, 64, 128$								
Radio de transmisión	10 m								
Velocidad de transmisión	250 kBps								
Velocidad de movimiento	0.5 - 1.5 m/s								
Semillas aleatorias	100								
Tamaño de mensaje	1 kB								
Creación de mensajes	Un mensaje por vértice en $t = 0$								
Tiempo de simulación	200,000 segundos								
Distribución de buffers	Ub	Sb	Nb						
Modelo de movilidad	Random waypoint			Shortest path map-based					
Esquema de energía	Sin restricciones				Completa	Aleatoria	Sin restricciones		
Tamaño del escenario	590 m × 532 m			4000 m × 3000 m					
Tiempo máximo de llegada	0						10k	20k	40k

parámetros de la Tabla 1. Se incluyeron las siguientes suposiciones en el escenario de simulación para simular con precisión el problema DMHM:

1. Cada vértice genera solo un mensaje al inicio de la simulación.

2. Los destinos finales de los mensajes no están en la red.

5.3. Selección de parámetros

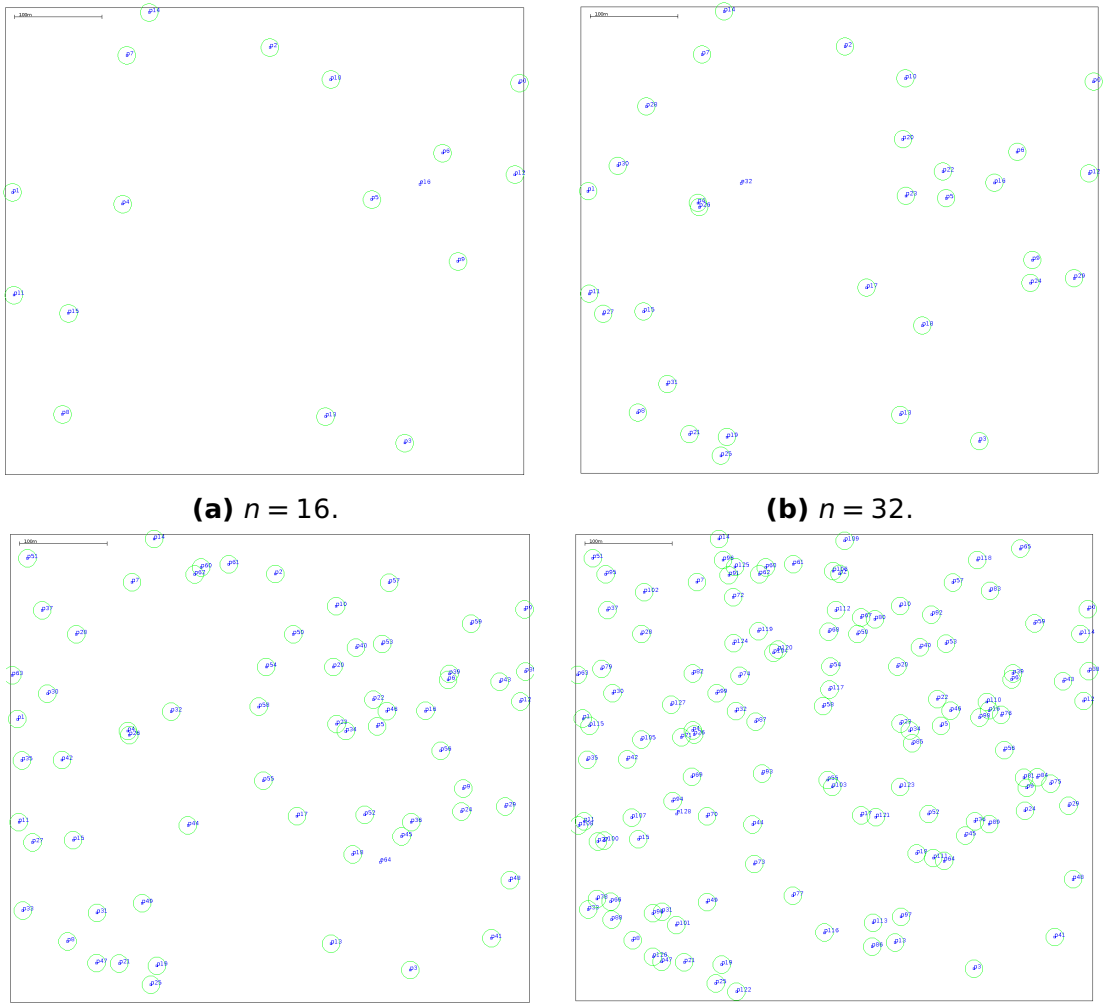
Se realizaron experimentos preliminares para determinar los valores apropiados de los parámetros del simulador para emular el DMHM (ver Apéndice B). La Tabla 1 enumera la configuración de los parámetros. A continuación, describimos cada uno de estos parámetros.

5.3.1. Número de vértices

El número total de vértices, n, en el área de simulación toma valores del conjunto $\{16, 32, 64, 128\}$. En la Figura 11 se observan las cuatro diferentes densidades de vértices en un área rectangular sin restricciones espaciales.

5.3.2. Área de simulación

Las simulaciones experimentales usan dos áreas de forma rectangular. La primera área no tiene restricciones espaciales (ver Figura 11), y su tamaño es de 590 m × 532 m. La segunda área considera restricciones espaciales. Esta área corresponde a un



(a) $n = 16$. **(b)** $n = 32$.

(c) $n = 64$. **(d)** $n = 128$.

Figura 11. Ejemplo del escenario cuadrado generado por el simulador ONE con las cuatro distintas densidades de vértices utilizadas en la experimentación. En cada escenario se representan los vértices como puntos azules con un identificador único y con su radio de transmisión representado como un circulo verde. a) Escenario con 16 vértices. b) Escenario con 32 vértices. c) Escenario con 64 vértices. d) Escenario con 128 vértices.

mapa de la ciudad de Ensenada, Baja California, México, y su tamaño es de 4000 m × 3000 m. Este mapa se generó por medio de la página OpenStreetMaps (Figura 12a) y se transformó con la herramienta osm2wkt para generar un grafo conectado. De tal forma que se pueda utilizar como un escenario con calles en el simulador ONE (Figura 12b).

(a) Mapa obtenido de https://www.openstreetmap.org

(b) Mapa utilizado en ONE simulator para el Escenario D.

Figura 12. Mapas de la ciudad de Ensenada, Baja California, México. (a) El mapa original en OpenStreetMap. (b) El mapa generado para la simulación.

5.3.3. Interface de comunicación

Bluetooth y Wi-Fi son las interfaces de comunicación más comunes utilizadas en redes oportunistas. En las simulaciones se utiliza Bluetooth porque se utiliza más comúnmente que Wi-Fi en comunicaciones oportunistas con dispositivos móviles heterogéneos. En la práctica, la mayoría de los dispositivos Bluetooth tienen fácilmente un alcance de transmisión de 10 m y una velocidad de transmisión de 250 kBps, independientemente de cualquier obstáculo o interferencia.

5.3.4. Modelos de movilidad

Los experimentos utilizan los modelos de movilidad ‘random waypoint’ y ‘shortest path map-based’ implementados en el simulador ONE. Para ambos modelos, se establece un intervalo de velocidad entre 0.5 y 1.5 m/s para simular el ritmo de una caminata humana promedio.

El modelo de movilidad *random waypoint* (Johnson y Maltz, 1996) modela un movimiento aleatorio de los usuarios de la siguiente forma: cada participante se coloca

en una posición aleatoria dentro del área de simulación. Posteriormente, a medida que avanza la simulación, cada participante se detiene o 'pausa' en su posición actual por un periodo de tiempo, después seleccionan aleatoriamente una nueva posición y velocidad de desplazamiento. Cada participante continúa con este comportamiento, alternadamente pausando y moviéndose a una nueva posición durante toda la simulación. Usando este modelo, los participantes parecen pasear por toda el área con un descanso entre un punto y otro. Este modelo movilidad es de los más utilizados al evaluar el desempeño de algoritmos de enrutamiento oportunista.

Por otra parte, el modelo de movilidad basado en mapas llamado *shortest path map based* funciona de forma similar a *random waypoint*. Primero, se elige un punto aleatorio en el mapa. Segundo, se calcula la ruta más corta que tiene que recorrer el vértice en el mapa para llegar al punto utilizando el algoritmo de Dijkstra (Dijkstra, 1959). Finalmente, el vértice se mueve desde su origen a su coordenada destino a través de la ruta trazada.

5.3.5. Parámetros de los mensaje

Se fijó el tamaño del mensaje a 1 kB. Este tamaño es útil para una amplia variedad de escenarios modelados por el problema DMHM. Se supone que cada vértice genera solamente un mensaje al tiempo que se une a la red.

5.3.6. Capacidad de almacenamiento de los vértices

Se utilizan tres tipos de distribuciones para la capacidad de almacenamiento de los vértices (buffers): 'Buffers uniformes' (Bu), 'Buffers semi-uniformes' (Bs), y 'Buffers no uniformes' (Bn).

- *Buffers uniformes*. En esta distribución, todos los vértices tienen el mismo tamaño de buffer, $\kappa_v = n/2$, es decir, cada vértice puede almacenar a lo más $n/2$ mensajes diferentes.
- *Buffers semi-uniformes*. Esta distribución divide equitativamente el conjunto de n vértices en cuatro grupos, y los grupos tienen un tamaño de buffer de $n/2$, $n/4$, $n/8$ y $n/16$, respectivamente.

Tabla 2. Valor óptimo del estimador l_v (L') para diferentes números de vértices y distribuciones.

Número de vértices (n)	Valor de L′		
	Bu	**Bs**	**Bn**
16	8	3.75	4.5
32	16	7.5	8.5
64	32	15	16.5
128	64	30	32.5

- *Buffers no uniformes*. Esta distribución divide equitativamente el conjunto de n vértices en $n/2$ grupos. El tamaño de buffer de cada grupo es i, donde $1 \leq i \leq n/2$.

No se consideran distribuciones donde $k_v > n$, para cualquier vértice. Porque puede resolverse de forma sencilla con cualquier algoritmo de enrutamiento que inunde la red. Una distribución más desafiante para el DMHM es cuando $k_v \leq n$, para cualquier vértice.

La Tabla 2 muestra el valor óptimo de l_v para cada distribución y número de vértices. Note que en algunos casos este valor no es entero, por lo que son casos de mayor dificultad y como consecuencia es posible que no exista una distribución óptima de todos los mensajes.

5.3.7. Modelo de energía y parámetros

En los experimentos se utilizó el modelo de energía y los valores de los parámetros propuestos por Rodriguez-Silva *et al.* (2012). Los autores de este modelo midieron el promedio de energía consumida de un teléfono móvil Nokia E52 usando una interface Bluetooth. Ellos obtuvieron que la energía consumida por el escaneo, envío, y recepción de archivos es de 0.06 mW/s, 0.08 mW/s, y 0.08 mW/s, respectivamente. Ellos reportaron que la capacidad de la batería de este teléfono es de 4800 mW/s.

Este modelo de energía es adecuado para nuestro escenario de aplicación. El simulador ONE ya incluye una implementación de este modelo, el cual se ha utilizado recientemente en algunos trabajos de investigación (Ababou *et al.*, 2015; Rolim *et al.*, 2016; Sobin *et al.*, 2017) con los mismos valores de los parámetros propuestos por Rodriguez-Silva *et al.* (2012).

En este trabajo se consideran tres esquemas diferentes de energía (sin restricciones, carga completa, y carga aleatoria) para la capacidad de la batería. El primer

esquema supone que cada vértice tiene energía sin restricciones. El segundo esquema supone que cada vértice tiene una carga completa de energía al principio de la simulación. El tercer esquema supone que la energía inicial de cada batería es un valor aleatorio con una distribución uniforme entre 50 % y 100 % de la capacidad máxima. Ningún esquema considera la posibilidad de recargar la batería

5.3.8. Tiempo máximo de llegada de los vértices

La simulación considera cuatro esquemas diferentes para los tiempos de llegada de los vértices. Estos esquemas suponen que los vértices se unen a la red aleatoriamente con cuatro distribuciones uniformes en los intervalos [0, 0], [0, 10k], [0, 20k], y [0, 40k] segundos. En particular, el intervalo [0, 0] significa que todos los vértices se unen a la red al comienzo de la simulación

5.4. Escenarios de simulación

Esta subsección presenta cinco escenarios de simulación. Cada escenario tiene diferentes suposiciones que se ilustran en los parámetros de la Tabla 1.

- *Escenario A*. Distribución de buffer Bu y una área sin restricciones espaciales.
- *Escenario B*. Distribución de buffer Bs y una área sin restricciones espaciales.
- *Escenario C*. Distribución de buffer Bn y una área sin restricciones espaciales.
- *Escenario D*. Distribución de buffer Bn, una área con restricciones espaciales, y con/sin restricciones de energía.
- *Escenario E*. Distribución de buffer Bn, una área con restricciones espaciales, sin restricciones de energía, y diferentes tiempos de llegadas de los vértices.

El objetivo de los escenarios *A*, *B*, y *C* es analizar los efectos de diferentes distribuciones de buffers. Mientras que el escenario *D* solo considera la distribución de buffer Bn, con restricciones de energía (ver Sección 5.3.7), y utilizando una área con un mapa (ver Figura 12) para representar un escenario más realista. Finalmente, el objetivo del escenario *E* es medir el impacto de diferentes tiempos de llegada de los vértices a la red (ver Sección 5.3.8) cuando se utiliza el mismo escenario realista que en el escenario *D* sin restricciones de energía.

5.5. Análisis estadístico

Para todos los experimentos se realizaron 100 ejecuciones por cada modelo de movilidad y algoritmo de ruteo, cambiando la semilla aleatoria inicial en cada ejecución. Al finalizar cada ejecución se realizó un análisis estadístico de los resultados de las métricas. Se utilizó la siguiente lógica por cada par de muestras. Primero, se utilizó la prueba de Lilliefors (1967) para determinar si los datos tienen o no una distribución normal. Si el resultado de esta prueba indica que los datos no provienen de una distribución normal, se aplica la prueba de suma de rangos de Wilcoxon (Conover, 1999); De lo contrario, se aplica la prueba t-student para determinar si existe una diferencia estadísticamente significativa entre los resultados. En el Apéndice C se provee una descripción más detallada de este análisis.

5.6. Entorno computacional

Los experimentos se ejecutaron en una computadora Dell PowerEdge R900 con 12 procesadores Intel Xeon a 2.13 GHz, 32 GB de RAM y con el sistema operativo Linux CentOS 6. Para la implementación de los algoritmos se utilizó Java 1.8 y se compiló con JDK 8.

5.7. Parámetros de los algoritmos

Se comparó el algoritmo EDEN contra dos algoritmos de enrutamiento oportunista bien conocidos, inundación epidémica (*epidemic flooding*) y Spray and Wait (*SnW*). No se modificó ningún parámetro del primero, dejando exactamente como está implementado en el simulador ONE. Para el algoritmo SnW, el número inicial de copias, L, se asignó igual al 15 % de n como sus autores lo recomiendan.

Además, para comparar los resultados de todos los algoritmos anteriores contra un algoritmo ideal, se generó el número óptimo de copias ($\sum \frac{k_v}{n} \ \ \forall v$) de cada mensaje m_v y se distribuyeron aleatoriamente en los vértices. Se llamó a este “algoritmo” *Ideal*, ya que refleja una distribución de mensajes ideal si se conoce *a priori* el número óptimo de copias para cada mensaje y un ente global las distribuye aleatoriamente en la memoria de los vértices. El algoritmo Ideal se muestra en el pseudocódigo 7.

Pseudocódigo 7 Distribución ideal de mensajes

Entrada: La colección $\Omega_n = \{m_1, m_2, \ldots, m_{n-1}, m_n\}$, i.e., un conjunto ordenado con el mensaje inicial para cada vértice.

1: $K \leftarrow 0$ ▷ Variable para la memoria total de la red.
2: $L \leftarrow 0$ ▷ Variable para el número de copias óptimo.
3: $CopiasPorMsj \leftarrow \emptyset$ ▷ Lista para las copias restantes por mensaje.
4: **for each** vértice $v_i \in V$ **do**
5: guarda el mensaje m_i en b_i ▷ Agrega a cada vértice su mensaje inicial.
6: $K \leftarrow K + \kappa_i$ ▷ Calcula la memoria total en la red.
7: **end for**
8: $L \leftarrow K/n$ ▷ Obtiene el número de copias óptimo L.
9: **for each** mensaje $m_i \in \Omega_n$ **do** ▷ Genera el número de copias por mensaje.
10: **if** $L == \lceil K/n \rceil$ **and** $L == \lfloor K/n \rfloor$ **then**
11: $CopiasPorMsj[i] \leftarrow L - 1$
12: **else** ▷ Si L tiene parte decimal aplica redondeo estocástico.
13: $CopiasPorMsj[i] \leftarrow stochastic\_round(K/n) - 1$
14: **end if**
15: **end for**
16: **while** $CopiasPorMsj.size > 0$ **do**
17: $\omega_n \leftarrow \emptyset$
18: **for each** element i in $CopiasPorMsj$ **do** ▷ Crea una nueva colección de mensajes.
19: $\omega_n \leftarrow \omega_n \cup m_i$
20: $CopiasPorMsj[i] \leftarrow CopiasPorMsj[i] - 1$
21: **if** $CopiasPorMsj[i] == 0$ **then** ▷ Se utilizan solamente los mensajes con copias.
22: elimina i de $CopiasPorMsj$
23: **end if**
24: **end for**
25: permuta aleatoriamente ω_n
26: **for each** mensaje $m_i \in \omega_n$ **do**
27: sea v_x un vértice seleccionado aleatoriamente y sin reemplazo de V
28: **if** $|\kappa_x - b_x.size| \leq m_i.size$ **and** $m_i \notin b_x$ **then** ▷ Si v_x tiene espacio y no tiene a m_i.
29: guarda el mensaje m_i en b_x
30: **else**
31: regresa al paso 27 ▷ Si no se guarda el mensaje selecciona otro vértice.
32: **end if**
33: **end for**
34: **end while**

Finalmente, para las dos versiones del algoritmo EDEN, se usó $historic.size = 50$ y $\epsilon = 0.5$ como parámetros iniciales. La Tabla 3 ilustra los parámetros iniciales utilizados por los algoritmos.

Tabla 3. Parámetros de los algoritmos.

Algoritmos	**Parámetros iniciales**		
Epidemic flooding (F)	-		
Spray and wait (SnW)	$L = (0.15 \times n)$	modo binario	
EDEN (E)	$m = 50$	$\epsilon = 0.5$	*calentamiento* inactivo
EDEN$_C$ (E$_C$)	$m = 50$	$\epsilon = 0.5$	*calentamiento* activo

5.8. Resumen

En este capítulo, se describen todas las variables, parámetros y algoritmos utilizados para poder realizar los experimentos propuestos en esta tesis. Sin embargo, es importante definir el papel de cada uno de estos elementos en el diseño experimental.

Las variables dependientes o lo que se pretende medir son todas las métricas descritas en la sección 5.1. De éstas, las variables PUB, FUR y el producto entre ellos utilizan una medida de intervalo. Alternativamente, las variables del número promedio de extracciones y el tiempo de convergencia utilizan una medida de razón. En las figuras del Capítulo 6, éstas variables se muestran en el eje horizontal de las gráficas. Las variables independientes en el experimento son los algoritmos mencionados en la sección 5.7. En las figuras del Capítulo 6, éstas variables se muestran en el eje vertical de las gráficas.

Para cada experimento se simula un escenario con el software descrito en la sección 5.2 y fijando un conjunto de los parámetros descritos en la sección 5.3, como el modelo de movilidad, el número de vértices y la capacidad de almacenamiento. Estas covariables (variables controladas independientes que junto a una o más explican un fenómeno) forman un escenario como los planteados en la sección 5.4. Este escenario representa un sujeto y un algoritmo representa un tratamiento. De tal forma que se le da el mismo escenario a cada algoritmo y se miden los resultados obtenidos

por medio de las variables dependientes. Es decir, se utiliza un diseño intra-sujetos (*within-subject design*). Finalmente, se utiliza el análisis estadístico mencionado en la sección 5.5 para definir, por pares de muestras, qué algoritmo (tratamiento) es el mejor para cada escenario (sujeto).

Capítulo 6. Resultados experimentales

En este capítulo se presentan los resultados de la simulación para todos los escenarios. Primero, se presenta el tiempo promedio de convergencia del estimador en el algoritmo EDEN para el escenario *C*, con diferentes densidades de vértices, y el modelo de movilidad *random waypoint*. Después, se presentan los resultados de las métricas FUR, PUB, PNV, y el tiempo de convergencia de la red de todos los algoritmos utilizando los escenarios *A*, *B*, y *C* (definidos en la sección 5.4), con diferentes densidades de vértices y el modelo de movilidad *random waypoint*. Posteriormente, se muestran los resultados de PNV y el tiempo de consumo de energía de los vértices para todos los algoritmos en el escenario *D*. Finalmente, se muestran los resultados de PNV y el tiempo de convergencia de la red para el escenario *E*.

6.1. Convergencia del estimador

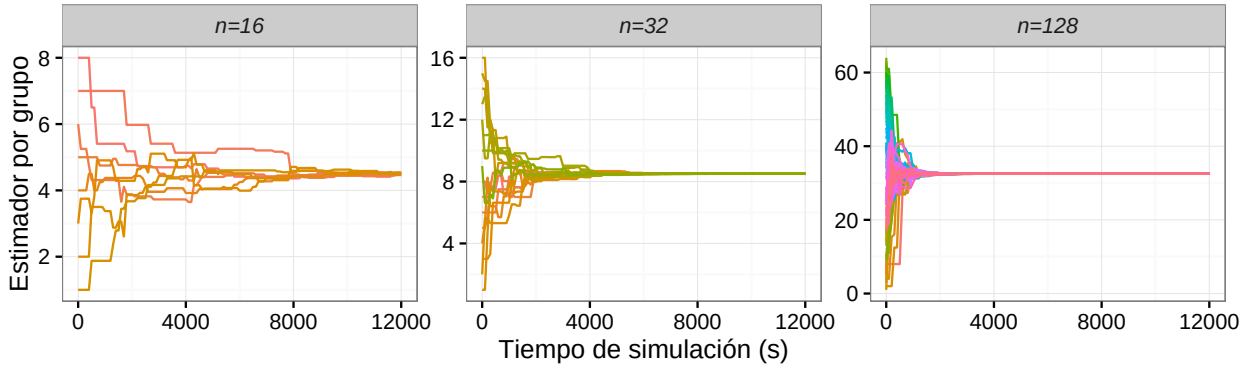


Figura 13. Tiempo de convergencia del estimador al usar el modelo de movilidad *random waypoint* para el escenario *C* para diferentes densidades de vértices.

La Figura 13 muestra el tiempo promedio de convergencia del estimador agrupando los vértices que tienen la misma capacidad de almacenamiento (ver Sección 5.3.6) y para diferentes valores de *n* en el escenario *C*. Estos valores son siempre inferiores a 8,000 segundos (aproximadamente 2 horas). Cuanto mayor sea la densidad, menor es el tiempo de convergencia del estimador. Las gráficas de la Figura 13 ilustran el comportamiento general de la convergencia del estimador, ya que el comportamiento no cambia significativamente usando los escenarios *A* y *B*.

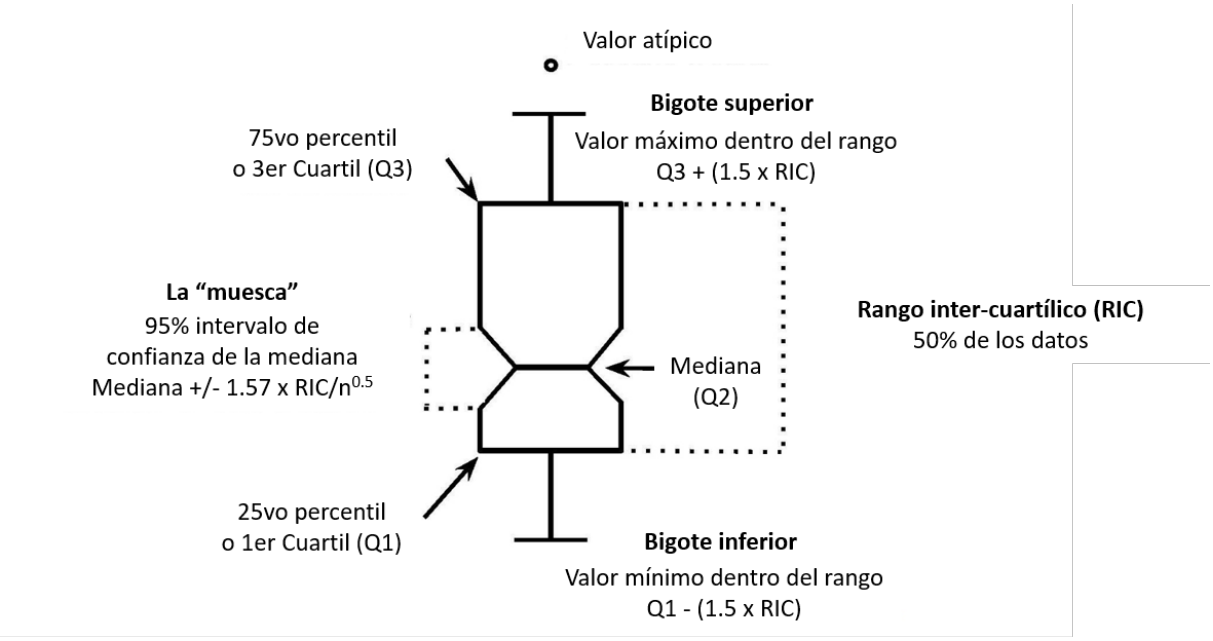


Figura 14. Descripción de un diagrama de caja con muesca.

6.2. Simulación para el escenario *A*

Esta sección presenta los resultados experimentales para las métricas FUR, PUB, FUR×PUB, PNV, y el tiempo de convergencia utilizando el escenario *A*. Los primeros tres escenarios presentados en este capítulo se enfocan en evaluar el impacto en el desempeño de los algoritmos al considerar tres diferentes distribuciones de tamaño de buffers, utilizando una distribución diferente para cada uno de los escenarios (ver sección 5.4). Estos tres escenarios consideran que todos los vértices en la red tienen las misma interface de comunicación para transmitir mensajes (ver sección 5.3.3) y no tienen restricciones de energía. Además, todos los participantes en la red se mueven aleatoriamente (ver sección 5.3.4) dentro de una área sin restricciones espaciales (ver sección 5.3.2). Finalmente, en los primeros cuatro escenarios todos los vértices participan desde el inicio de la simulación (ver sección 5.3.8).

En este primer escenario se analiza el efecto de utilizar buffers uniformes para todos los vértices (ver sección 5.3.6), i.e., todos en la red utilizan la misma capacidad de almacenamiento para los mensajes.

En las siguientes figuras, cada diagrama de caja con muescas representa la dispersión de los datos obtenidos como resultado de 100 ejecuciones. La Figura 14 muestra

una descripción de un diagrama de caja con muesca. La parte inferior y superior de la caja son el primer y el tercer cuartil, respectivamente. La línea dentro de la caja es el segundo cuartil (la mediana). Los extremos de los bigotes representan el punto más bajo (más alto) dentro de 1.5 RIC del tercer (primer) cuartil. Los datos no incluidos entre los bigotes se consideran valores atípicos y se ilustran con puntos negros.

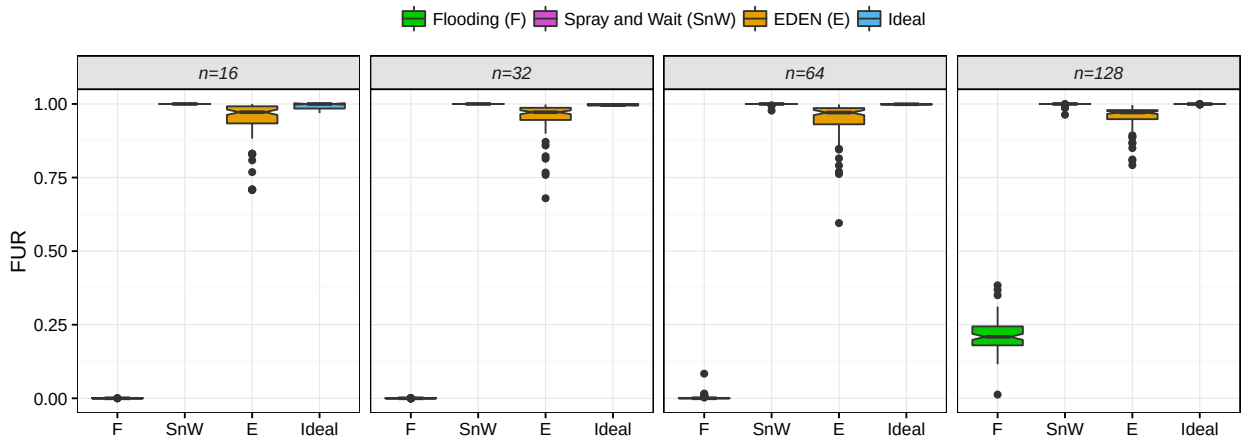


Figura 15. Factor de Uniformidad Relativo (FUR) para el escenario *A*, para todos los algoritmos y valores de *n*.

La Figura 15 muestra el FUR para diferentes algoritmos y valores de *n*. El FUR para *epidemic flooding* es muy bajo para todas las densidades en comparación con los otros algoritmos, en general, menor a 0.25. A pesar de que el FUR de EDEN y SnW son cercanos a 1, SnW muestra el mejor rendimiento con diferencia significativa para las densidades de 32, 64 y 128 de acuerdo a la Tabla 11.

En la Figura 16 se observa el PUB para diferentes algoritmos y valores de *n*. SnW tiene el peor rendimiento para esta métrica, menor a 0.4, ya que este algoritmo da prioridad a la uniformidad de los mensajes en lugar de la utilización de la memoria. En general, los valores PUB para los algoritmos de *epidemic flooding* y EDEN son superiores a 0.94. Particularmente, *epidemic flooding* siempre obtiene aproximadamente un valor de uno, i.e., prácticamente llena las memorias de todos los vértices en cada ejecución para todos los valores de *n*, obteniendo el mejor rendimiento para esta métrica.

En la Figura 17 se presenta la métrica FUR×PUB para diferentes algoritmos y valores de *n*. Los valores de este producto son similares a los valores del FUR ya que

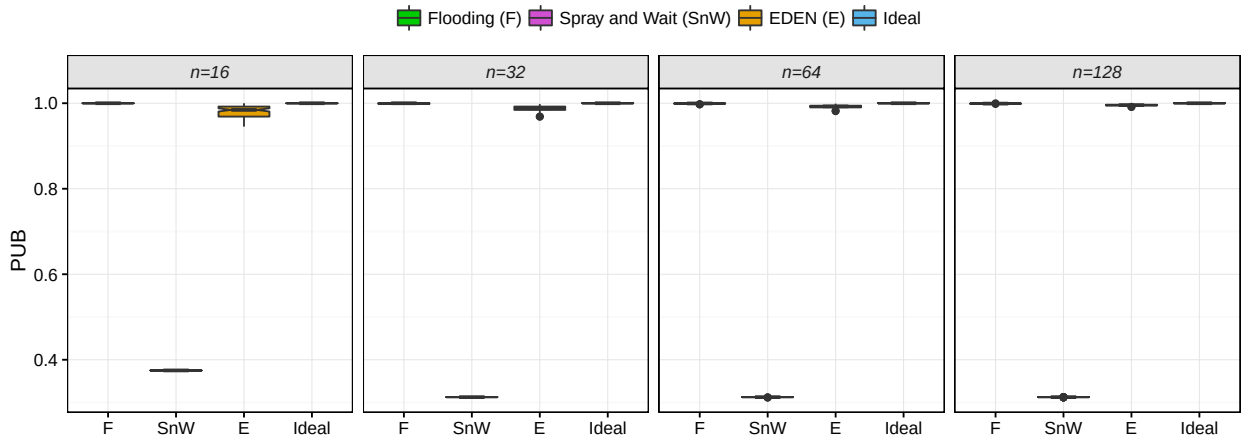


Figura 16. Porcentaje de Utilización del Buffer (PUB) para el escenario *A*, para todos los algoritmos y valores de *n*.

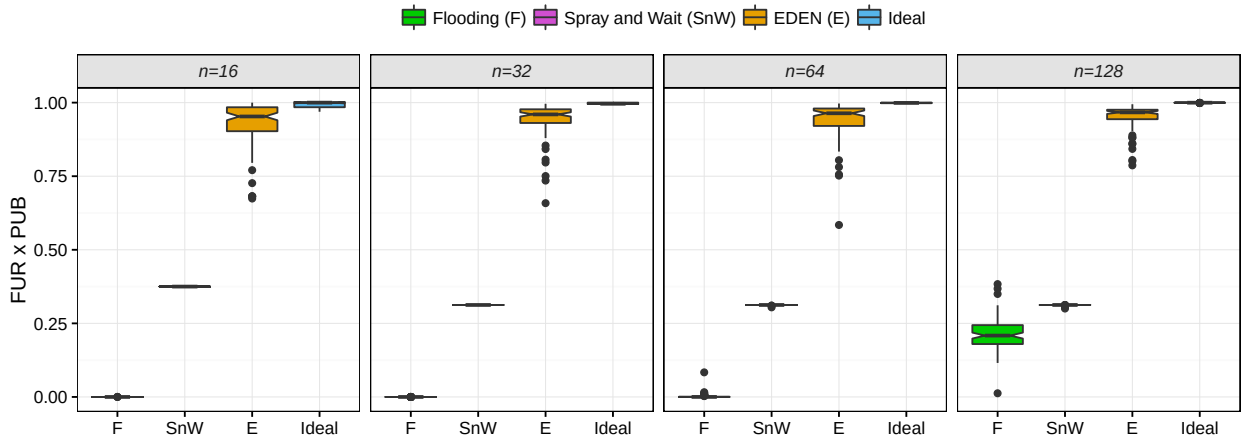


Figura 17. Producto del FUR y PUB (FUR×PUB) para el escenario *A*, para todos los algoritmos y valores de *n*.

la mayoría de los valores en PUB son muy cercanos a uno para todos los algoritmos, excepto para SnW. Para este último algoritmo, su producto siempre es menor a 0.38. Por simplicidad, en el resto de este capítulo sólo se muestra la figura con FUR×PUB, omitiendo las figuras por separado de FUR y PUB.

La Figura 18 muestra el PNV para el modelo de movilidad *random waypoint* utilizando diferentes algoritmos y valores de *n*. Para esta métrica se busca minimizar el número promedio de extracciones, ya que entre más pequeño sea, mejor. En *epidemic flooding*, cuando un vértice llena su memoria sigue recibiendo mensajes reemplazando los mensajes más antiguos, es por ello que es posible que algunos mensajes no

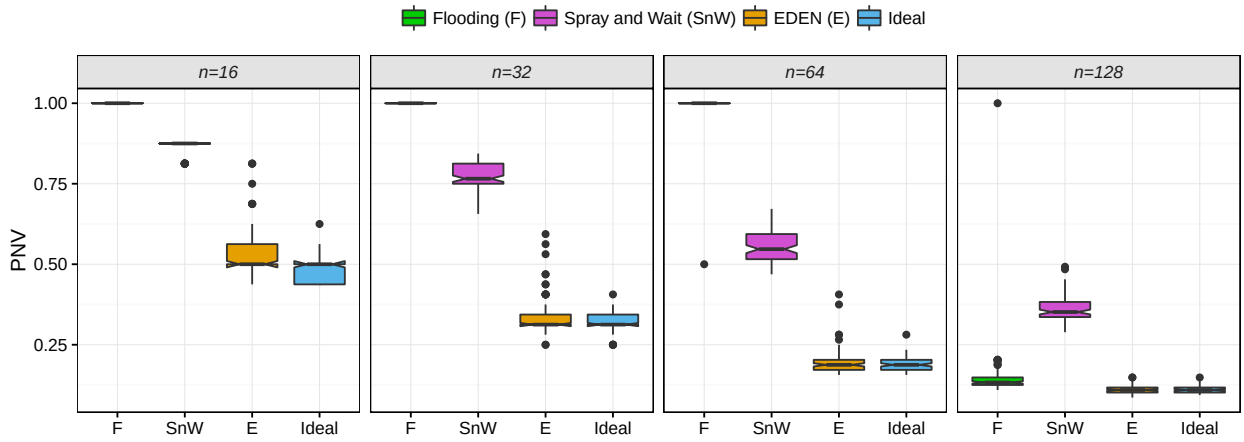


Figura 18. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *A*, para todos los algoritmos y valores de *n*.

existan en la red al terminar la simulación. Por lo que es posible que incluso después de extraer todos los vértices, este algoritmo no pueda obtener al menos una copia de cada mensaje.

En la Figura 18 se observa la consecuencia de este comportamiento con *epidemic flooding* cuando n es igual a 16, 32, y 64, donde, incluso después de extraer todos los vértices de la red no se puede obtener Ω_n (es decir, $\lambda/N = 1$). El segundo peor algoritmo es SnW para todo n. El algoritmo EDEN obtiene el mejor rendimiento de todos los algoritmos analizados. Ésto se ve reflejado en la Tabla 11, donde se observa que no hay diferencia significativa entre los valores de EDEN para $n = 128$ y el desempeño del algoritmo ideal.

La Figura 19 muestra el tiempo de convergencia de la red para varios algoritmos y valores de n. *Epidemic flooding* produce los peores resultados. Inclusive para este algoritmo, la red no converge en el tiempo de simulación final para $n > 16$. Por el contrario, ya que SnW transmite muy pocos mensajes en comparación con los otros algoritmos, su tiempo de convergencia es el más pequeño, en promedio, menor a $7,500$ segundos. Para EDEN, su tiempo de convergencia es en promedio $23,000$ segundos para toda n.

El tiempo de convergencia de la red en los escenarios *B*, *C*, y *D* sin restricción de energía, son muy similares a los del escenario *A*, ya que el orden de jerarquía de los

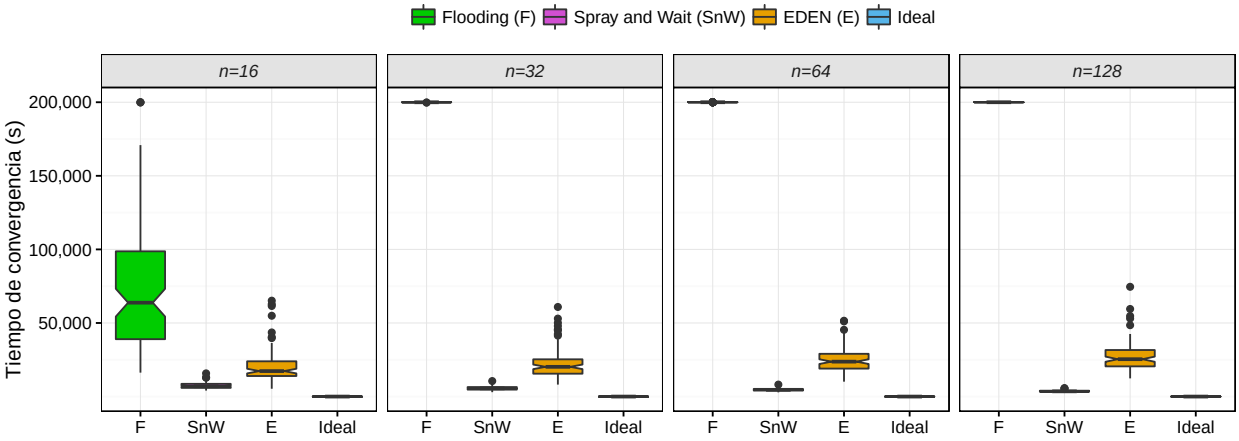


Figura 19. Tiempo de convergencia de la red para el escenario *A*, para todos los algoritmos y valores de *n*.

algoritmos se mantiene sin importar el escenario. Adicionalmente, cuando EDEN con calentamiento se incluye en la gráfica de esta métrica, su rendimiento es ligeramente peor que el de EDEN. Por estas razones, se omiten las gráficas de esta métrica en las siguientes secciones; sin embargo, estas gráficas se incluyen dentro del Apéndice D.

6.3. Simulación para el escenario *B*

En esta sección se presentan los resultados de la simulación para el escenario *B*. En este segundo escenario se utilizan los mismos parámetros iniciales que en el escenario *A*, pero ahora utilizando una distribución de buffers semi-uniformes (ver sección 5.3.6).

En la Figura 20 se muestra la métrica FUR×PUB para varios algoritmos y valores de *n*. El producto para *epidemic flooding* es prácticamente cero y es el más bajo de todos los algoritmos y para todos los valores de *n*. En general, SnW genera el segundo peor desempeño. Su producto en promedio siempre es menor que 0.8, para todos los valores de *n*. Finalmente, EDEN supera a EDEN_C para todo *n*, excepto para 16, donde tienen un desempeño similar. Para $n = 16$, este comportamiento especial se debe probablemente al efecto del redondeo de l_v, ya que su valor óptimo es 3.75 (ver Tabla 2) y el algoritmo propuesto requiere un número entero de copias para obtener un mejor desempeño.

Note que en particular para EDEN_C, no todos los vértices inician la transmisión de sus mensajes al mismo tiempo. Cada vértice comienza sus transmisiones de mensajes

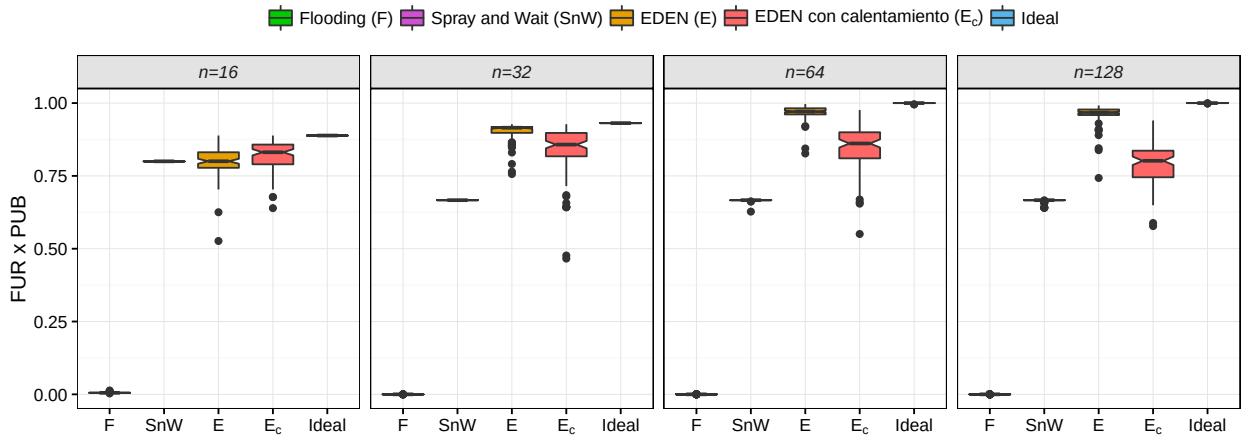


Figura 20. Producto del FUR y PUB (FUR×PUB) para el escenario *B*, para todos los algoritmos y valores de *n*.

inmediatamente después de que su estimador converge. Por lo tanto, los primeros vértices que transmiten no tienen problema al propagar las copias de sus mensajes; por el contrario, los últimos vértices que inician sus transmisiones rara vez pueden asignar todas las copias de sus mensajes. Este comportamiento disminuye la métrica de FUR×PUB.

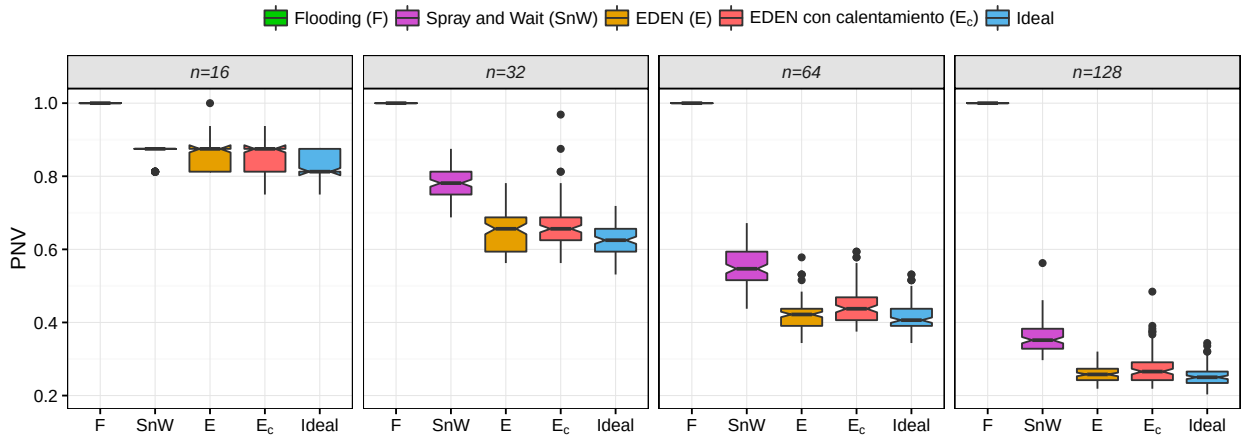


Figura 21. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *B*, para todos los algoritmos y valores de *n*.

En la Figura 21 se muestra el PNV para el modelo de movilidad *random waypoint* para todos los algoritmos y valores de *n*. *Epidemic flooding* obtiene un valor de uno para esta métrica, siendo el peor de todos los algoritmos y para todos los valores de *n*. En general, SnW obtienen el segundo peor desempeño. Ya que sus valores son en

promedio 10 % peores que los de EDEN y $EDEN_c$, para todo n, excepto para $n = 16$, donde son muy similares. El rendimiento de EDEN es en promedio 2 % mejor que el de $EDEN_c$ para todo n, excepto para $n = 16$, en la que son muy similares. Note que no hay diferencia significativa en los valores entre el Ideal y las variantes de EDEN, como se puede ver en la Tabla 12.

6.4. Simulación para el escenario *C*

En esta sección se presentan las simulaciones para el escenario *C*. En este tercer escenario se utilizan los mismos parámetros iniciales que en los escenarios *A* y *B*, pero ahora utilizando una distribución de buffers no uniformes (ver sección 5.3.6).

En la Figura 22 se presenta FUR×PUB para todos los algoritmos y todos los valores de n. El producto para *epidemic flooding* es prácticamente cero, siendo el más bajo para todo n. SnW genera el segundo más bajo para todo n. Su producto es menor que 0.66, en promedio, para todo n. Por último, EDEN es al menos 5 % mejor que $EDEN_c$ para todo n e incluso, entre más grande n, mayor es la diferencia entre estos dos algoritmos.

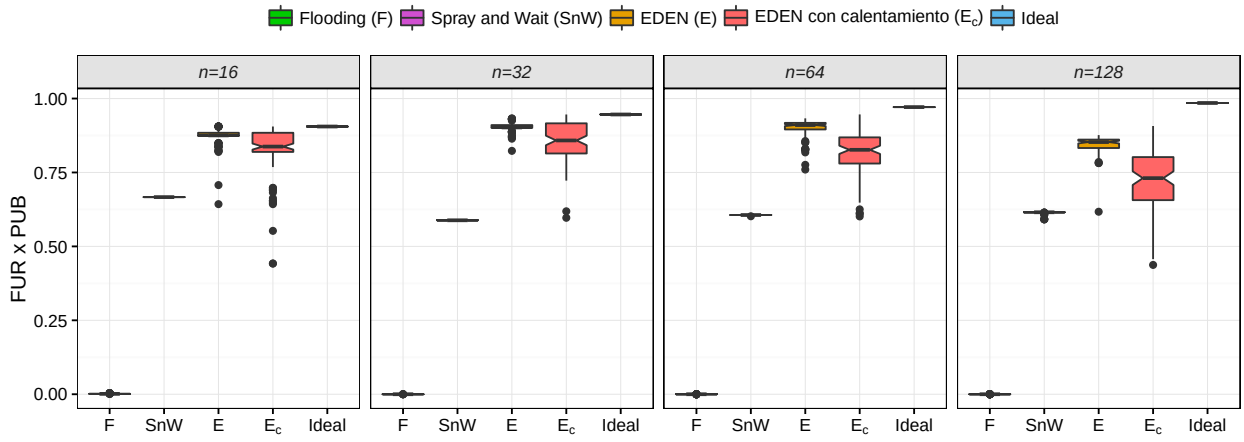


Figura 22. Producto del FUR y PUB (FUR×PUB) para el escenario *C*, para todos los algoritmos y valores de n.

La Figura 23 muestra el PNV para el modelo de movilidad *random waypoint* para todos los algoritmos y valores de n. EDEN es al menos 2 % mejor que $EDEN_c$, en promedio, para todo n. Además, $EDEN_c$ es al menos 10 % mejor que SnW, en promedio, para todo n. Finalmente, *epidemic flooding*, con un valor de uno para esta métrica, es

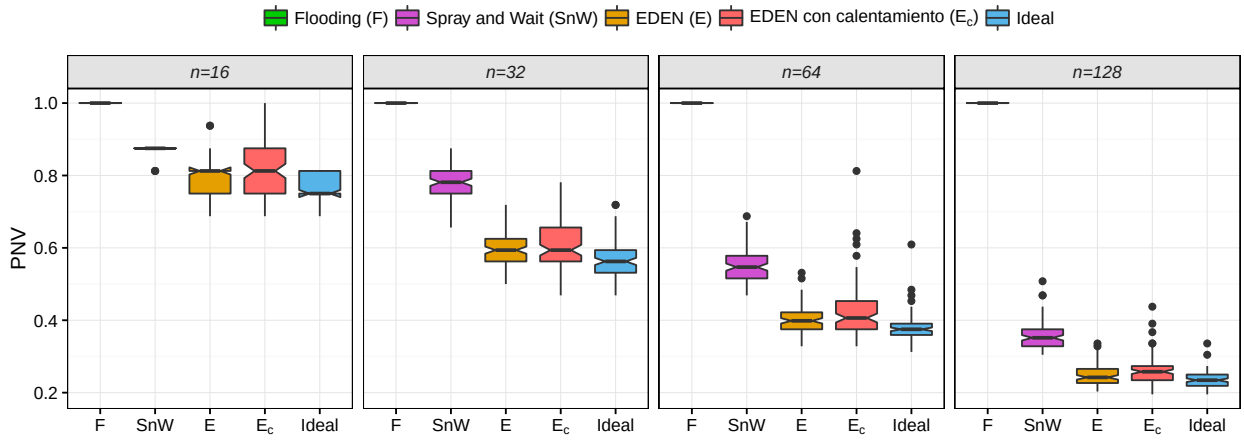


Figura 23. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *C*, para todos los algoritmos y valores de *n*.

el peor de todos los algoritmos.

6.5. Simulación para el escenario *D*

En esta sección se muestran los resultados para la métrica PNV en el escenario *D*, el cual considera restricciones de energía. En este cuarto escenario se utilizan los mismos parámetros iniciales que en el escenario *C*, pero ahora considerando tres diferentes esquemas de energía para los dispositivos en la red (ver sección 5.3.7). De aquí en adelante todos los escenarios utilizan una distribución de buffers no uniformes y la movilidad de todos los vértices es aleatoria (ver sección 5.3.4) dentro de un mapa con restricciones espaciales (ver sección 5.3.2). Estas suposiciones ayudan a evaluar el desempeño de los algoritmos simulando un escenario más realista.

En la Figura 24 se presenta el PNV para todos los algoritmos y todos los valores de *n* utilizando el esquema de energía sin restricciones, i.e., todos los vértices tienen carga infinita de energía. La Figura 24 mantiene prácticamente las mismas tendencias que la Figura 23 (escenario *C*). Por lo tanto, la descripción de las Figuras 23 y 24 es la misma. En el resto de esta sección, todas las comparaciones de gráficas de las figuras 25 y 26 son con respecto a las de la Figura 24. Note que SnW y *epidemic flooding* mantienen prácticamente los mismos valores para todos los esquemas de energía.

La Figura 25 presenta el PNV para todos los algoritmos y todos los valores de *n*, utilizando el esquema con carga de energía completa, i.e., todos los vértices empiezan

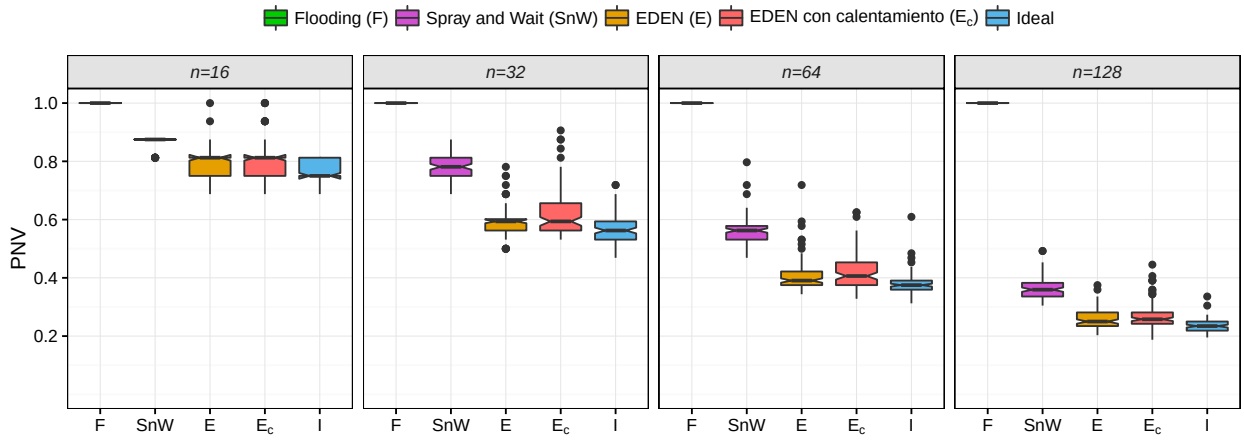


Figura 24. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *D*, sin restricción de energía, para todos los algoritmos y valores de *n*.

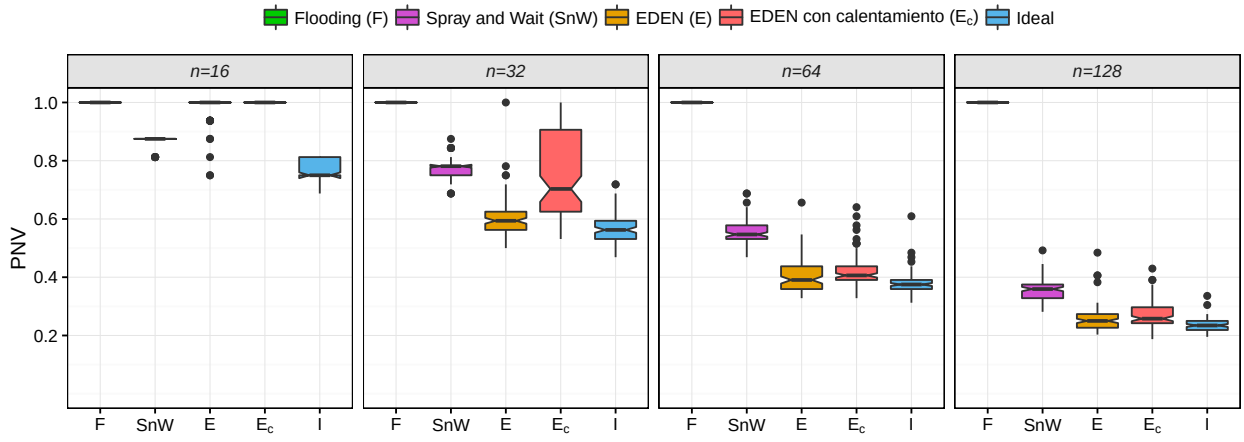


Figura 25. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *D*, con carga de energía completa, para todos los algoritmos y valores de *n*.

la simulación con carga completa y transmiten mensajes hasta que agotan su energía. Con respecto a cuando se utiliza un esquema de energía sin restricciones, todas las gráficas de la Figura 25 mantienen sus valores para todo *n*, excepto para los siguientes casos. Primero, el desempeño de EDEN_c decrece 13 %, en promedio, para $n = 32$. Segundo, los desempeños de EDEN y EDEN_c decrecen 19 %, en promedio, para $n = 16$. Más aún, note que los desempeños de estos dos algoritmos son incluso peores que SnW para $n = 16$.

Finalmente, la Figura 26 muestra el PNV para todos los algoritmos y valores de *n* utilizando el esquema de carga de energía aleatoria, i.e., todos los vértices comienzan

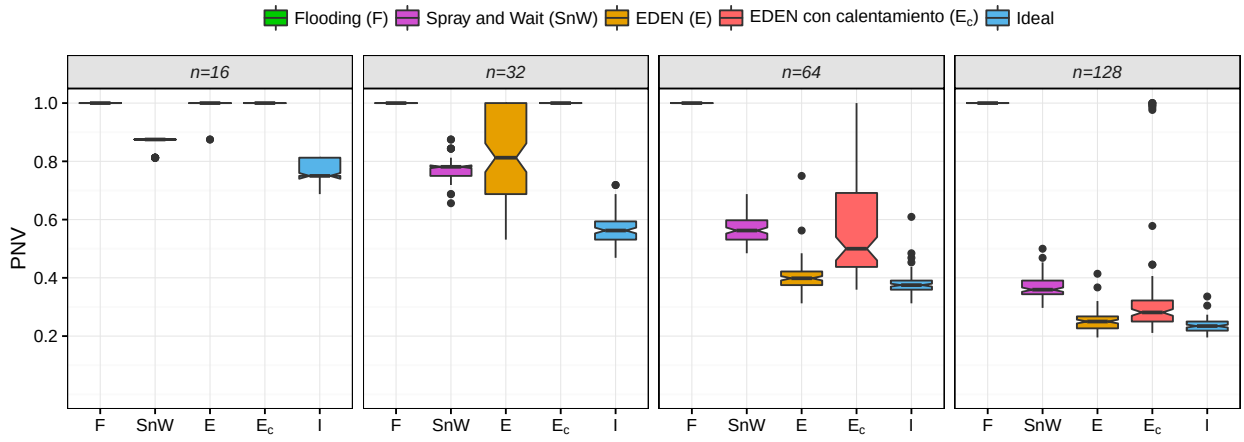


Figura 26. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *D*, con carga de energía aleatoria, para todos los algoritmos y valores de *n*.

la simulación con carga aleatoria y transmiten mensajes hasta que agotan su energía. EDEN mantiene los mismos valores para $n = 128$ y $n = 64$; mientras que, empeora su desempeño 23 % y 20 %, en promedio, para $n = 32$ y $n = 16$, respectivamente. EDEN_C mantiene los mismos valores sólo para $n = 128$; mientras que empeora su desempeño 19 %, 37 % y 16 %, en promedio, para $n = 64$, $n = 32$, y $n = 16$, respectivamente.

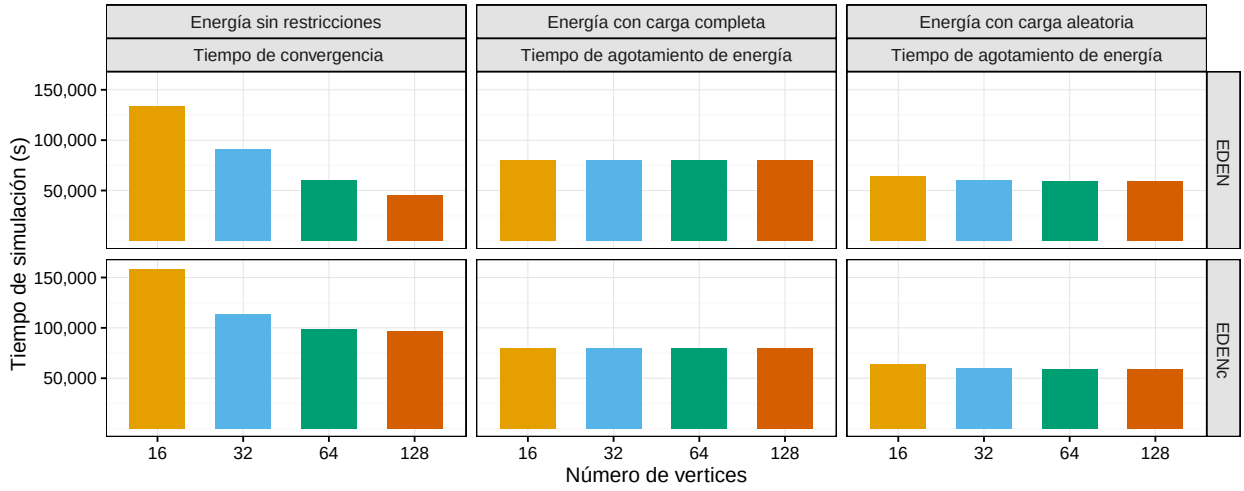


Figura 27. Tiempos de convergencia de la red para el esquema de energía sin restricciones (columna izquierda). Tiempos de agotamiento de energía para los esquemas de energía con carga completa y carga aleatoria (columna central y columna derecha, respectivamente). Utilizando el escenario *D* para los algoritmos EDEN y EDEN_C, y para todos los valores de *n*.

Los casos anteriores muestran un notorio decremento para EDEN y EDEN_C. La principal razón de este decremento es porque el promedio del tiempo de convergencia de la red para el esquema de energía sin restricciones (columna izquierda de la Fi-

gura 27) es más grande que los tiempos de agotamiento de energía del esquema con carga completa y con carga aleatoria (columna central y columna derecha de la Figura 27, respectivamente). Note que entre menor sea el número de vértices, más extenso es el tiempo de convergencia. Por esta razón, el decremento en el desempeño de EDEN y EDEN_C es más evidente para valores pequeños de n que para valores grandes de n. En particular, EDEN_C es el algoritmo más afectado por el retardo en el inicio de intercambio de mensajes debido a su etapa de calentamiento.

6.6. Simulación para el escenario *E*

Esta sección presenta los resultados para las métricas PNV y el tiempo de convergencia utilizando el escenario *E*. En este quinto escenario se utilizan los mismos parámetros iniciales que en el escenario *D* con el enfoque de energía sin restricciones. Particularmente, en este escenario se analiza el efecto de agregar vértices a la red en diferentes instantes. Para esto se consideran cuatro esquemas diferentes para los tiempos de llegada de los vértices a la red (ver sección 5.3.8).

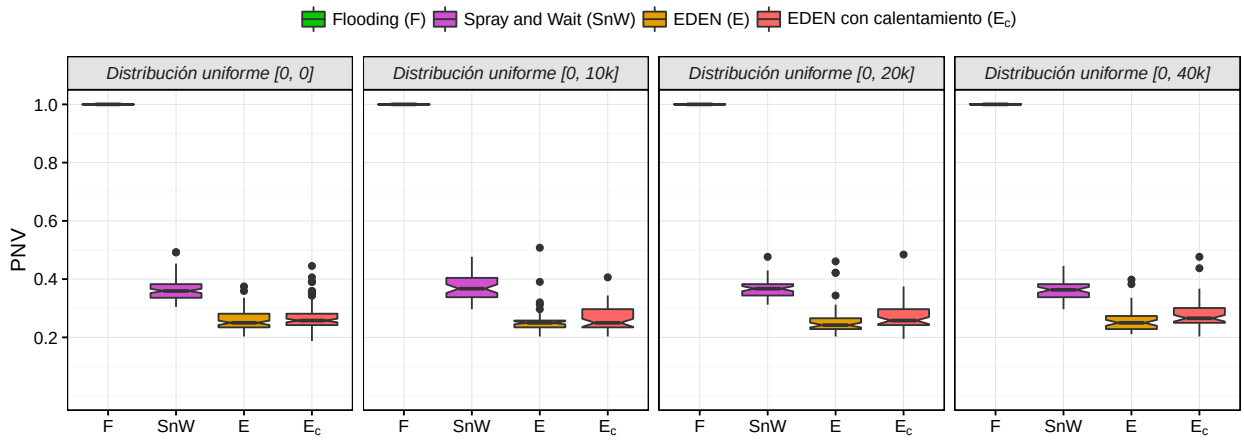


Figura 28. Promedio Normalizado de Vértices (PNV) para obtener todos los mensajes diferentes para el escenario *E*, con diferentes tiempos de llegada de los vértices, para varios algoritmos y para $n = 128$.

La Figura 28 muestra el PNV para varios algoritmos, cuatro distribuciones uniformes de tiempos de llegada, y $n = 128$. Para propósitos de comparación, la primera columna (de izquierda a derecha) de la figura incluye los resultados cuando todos los vértices se unen a la red al principio de la simulación. Los resultados experimentales indican que los diferentes tiempos de llegada prácticamente no afecta el PNV de ningún algoritmo. Es por ello, que en esta sección se omiten las gráficas para otros valores de n.

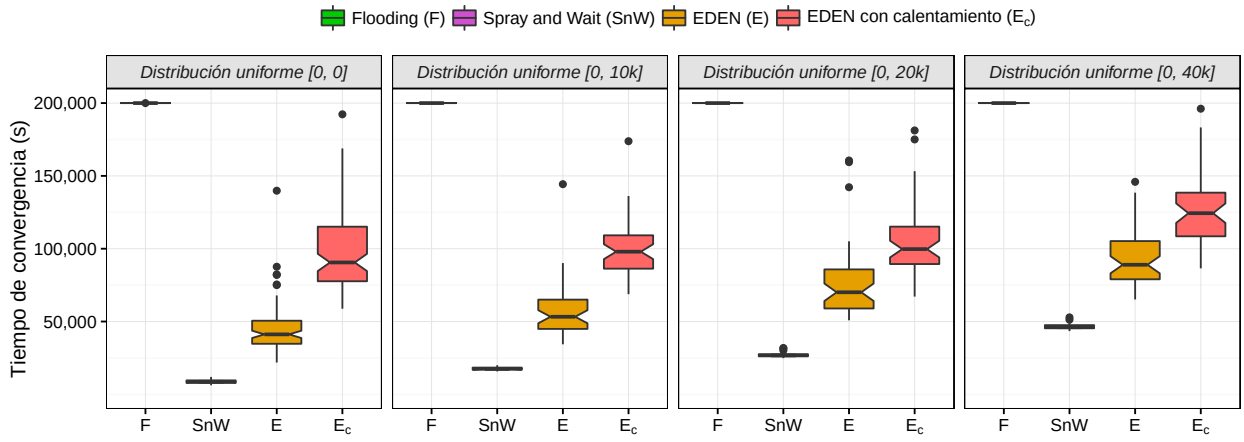


Figura 29. Tiempo de convergencia de la red para el escenario *E*, con diferentes tiempos de llegada de los vértices, para varios algoritmos y para $n = 128$.

La Figura 29 muestra el tiempo de convergencia de la red para varios algoritmos, cuatro distribuciones uniformes de tiempos de llegada, y $n = 128$. En general, SnW, EDEN, y EDEN_C requieren más tiempo para converger. Entre más amplio sea el intervalo del tiempo de llegada de los vértices, más grande es el tiempo de convergencia de la red. Particularmente, EDEN_C es el más afectado. Se conjetura que si se considera este retardo en los tiempos de llegada de los vértices a la red en escenarios con restricciones de energía, el impacto sería mucho más negativo.

6.7. Resumen

En este capítulo se presentaron los resultados de las simulaciones al comparar los algoritmos propuestos, EDEN y EDEN_C, contra otros algoritmos de enrutamiento del estado del arte en cinco escenarios con diferentes características. En estos escenarios se evalúa la capacidad de los algoritmos para resolver el DMHM. Cada escenario estudia una característica primordial en una red de comunicación. De tal forma que conforme se avanza en el desarrollo del capítulo se abordan escenarios cada vez más realistas.

Al inicio del capítulo se muestra que el tiempo de convergencia del estimador (utilizado en los algoritmos propuestos) es relativamente rápido, ya que toma máximo dos horas en converger. Este tiempo es esencial ya que ambos algoritmos necesitan un buen estimador para funcionar adecuadamente e incluso uno de ellos tiene que

esperar a que su estimador converja para enviar mensajes.

Los resultados del escenario *A* ayudan a analizar un estado de la red donde todos los vértices tienen el mismo tamaño para almacenar mensajes. En este escenario EDEN es mejor que los otros algoritmos en la métrica PNV (la métrica más importante para el DMHM). Posteriormente, en los resultados de los escenarios *B* y *C* la capacidad de almacenamiento de los vértices es paulatinamente más desigual entre ellos. Lo cual afecta en mayor medida a EDEN y EDEN_C. Sin embargo, en general se sigue superando a SnW, el algoritmo más cercano, en la métrica PNV.

Después, con los resultados del escenario *D* se estudia el efecto de agregar diferentes enfoques de energía para los vértices y un modelo de movilidad con restricciones en un mapa. Se consideran tres enfoques de disponibilidad de energía, donde conforme se avanza, cada enfoque considera una disponibilidad más austera de energía. Estos resultados muestran que EDEN y EDEN_C se ven más afectados conforme la energía es más limitada, mientras que el resto de los algoritmos mantienen el mismo desempeño. A pesar de esto, en general EDEN es el mejor en la métrica PNV, excepto para los casos donde *n* es más pequeño.

Por último, con el escenario *E* se analiza el efecto al considerar diferentes tiempos de llegada de los vértices a la red. Para todos los algoritmos se observó que el impacto en la métrica de PNV es muy reducido. Por el contrario, sí afecta la métrica del tiempo de convergencia de la red en la mayoría de los algoritmos.

Capítulo 7. Conclusiones y trabajo futuro

En esta tesis se propone un nuevo problema llamado distribución máxima y homogénea de mensajes (DMHM). Este problema surge como un intento de mitigar la falta de comunicación en situaciones en donde el funcionamiento de las redes de comunicación se interrumpe. Específicamente, cuando el destino del mensaje no está disponible temporalmente en la zona donde no hay conexión. También se propone el algoritmo EDEN y una variante de este, el algoritmo EDEN_c, que implementan un mecanismo de replicación y diseminación de mensajes para generar soluciones aproximadas al DMHM. Estas dos propuestas se comparan contra dos algoritmos clásicos de enrutamiento oportunista (*epidemic flooding* y SnW). Adicionalmente, se comparan contra el rendimiento de un algoritmo ideal. Se realizaron simulaciones numéricas para evaluar estos algoritmos en diferentes escenarios.

Note que los algoritmos *epidemic flooding* y SnW no se diseñaron específicamente para resolver el DMHM; sin embargo, son algoritmos comúnmente utilizados en enrutamiento oportunista. SnW da prioridad a la uniformidad de los mensajes en lugar de la utilización de la memoria; por el contrario, *epidemic flooding* hace lo opuesto. Mientras tanto, el objetivo de EDEN es el de maximizar los dos parámetros; por lo tanto, obtiene el mejor rendimiento para las métricas FUR×PUB y PNV para todos los escenarios.

Al comparar los algoritmos propuestos, es claro que EDEN_c es más simple que EDEN, ya que no requiere los procedimientos de promoción de copias ni de depuración de copias. Por esta razón, también requiere menos transmisiones de mensajes y un tiempo más extenso para lograr la convergencia en la red que EDEN. No obstante, en general, el rendimiento de EDEN_c es a lo más 2 % peor que EDEN para el PNV en la mayoría de los escenarios. Además, EDEN y EDEN_c obtienen valores a lo más 2 % y 4 % peores que el algoritmo ideal, respectivamente, para PNV en los escenarios *A*, *B*, y *C*. En estos tres escenarios se analiza principalmente, el impacto en el desempeño de los algoritmos al utilizar diferente capacidad de almacenamiento para los mensajes. Para esto se modifica gradualmente la uniformidad de los buffers de un escenario a otro. Esto afecta en mayor medida a EDEN y EDEN_c. Sin embargo, en general se sigue superando a SnW, el algoritmo más cercano, en la métrica PNV.

En general, para las dos variantes de EDEN, nuestros resultados experimentales muestran que los tiempos de convergencia tanto del estimador como de la red dependen del número de contactos entre los vértices de la red: cuanto más grande sea el número de vértices, menor será el tiempo de convergencia.

En el escenario *D*, se analiza el efecto de utilizar restricciones de energía en los dispositivos de la red. En este escenario, en general, los desempeños de EDEN y EDEN_c se deterioran a medida que disminuye la capacidad de la batería. Cuanto menor es el número de vértices, mayor es la degradación del rendimiento de estos dos algoritmos. A pesar de este comportamiento, EDEN es el mejor, según el criterio PNV, para la mayoría de los resultados de simulación de este escenario. SnW y la *epidemic flooding* mantienen prácticamente el mismo rendimiento para todos los esquemas de energía.

Note que el escenario *D* con un esquema de energía con carga aleatoria puede considerarse similar a una situación en la que todos los vértices salen gradualmente de la red. De tal forma que el agotamiento de una batería da como resultado la extracción de un vértice de la red. Los resultados en este escenario implican que cuanto antes se eliminen los vértices de la red, peor es el rendimiento de los algoritmos propuestos.

En el escenario *E* se consideran diferentes tiempos de llegada de los vértices a la red. El desempeño de los algoritmos, bajo el criterio PNV, es prácticamente el mismo que cuando se considera que todos los vértices llegan al comienzo de la simulación; sin embargo, los diferentes tiempos de llegada afectan considerablemente los tiempos de convergencia de la red.

Por lo tanto, retomando el objetivo general de esta investigación expresado con la siguiente pregunta de investigación: ¿Cuál es el mecanismo más adecuado para distribuir mensajes en un modelo de red sin cobertura global, de tal manera que el número de copias de cada mensaje sea máximo y uniforme, considerando un almacenamiento limitado de cada participante de la red y distintos escenarios de movilidad, energía, y número de participantes? Podemos afirmar que el algoritmo más adecuado para la gran mayoría de los casos es el algoritmo EDEN.

Para el trabajo futuro, sería interesante estudiar mecanismos que permitan ahorrar energía al algoritmo EDEN, debido a que el suministro de energía es un recurso elemental en escenarios de desastres (Araneda *et al.*, 2010). Otra dirección de inves-

tigación interesante para trabajo futuro es la evaluación de EDEN con modelos de movilidad que representen de forma realista los movimientos dentro de una área de desastre, por ejemplo, como el que es definido por Aschenbruck *et al.* (2009).

Por otra parte, un reto mayor sería modelar el DMHM en un entorno donde los vértices no necesariamente sean cooperativos e incluso algunos puedan realizar acciones maliciosas (e.g., envían estimadores incorrectos). Para lidiar con este tipo de comportamiento sería necesario utilizar un método basado en incentivos modelando el problema en un marco de teoría de juegos. Este tipo de estrategias aplicadas en escenarios de enrutamiento se considera un área fértil para investigación futura (Sobin *et al.*, 2016).

Literatura citada

Ababou, M., Kouch, R. E., Bellafkih, M., Ababou, N. 2015. Energy efficient and effect of mobility on acdtn routing protocol based on ant colony. En: *Proceedings of the International Conference on Electrical and Information Technologies, ICEIT*, Marzo. Marrakech, Morocco, pp. 335–340.

Aloi, G., Briante, O., Di Felice, M., Ruggeri, G., Savazzi, S. 2017. The sense-me platform: Infrastructure-less smartphone connectivity and decentralized sensing for emergency management. *Pervasive and Mobile Computing*, 42: 187 – 208.

Amah, T. E., Kamat, M., Abu Bakar, K., Abali, A. M., Moreira, W., Oliveira-Jr, A. 2017. Addressing the issue of routing unfairness in opportunistic backhaul networks for collecting sensed data. *Journal of Sensor and Actuator Networks*, 6(4).

Araneda, J., Rudnick, H., Mocarquer, S., Miquel, P. 2010. Lessons from the 2010 chilean earthquake and its impact on electricity supply. En: *Power System Technology (POWERCON), 2010 International Conference on*. pp. 1–7.

Arnholt, A. T. 2012. *BSDA: Basic Statistics and Data Analysis*. R package version 1.01.

Aschenbruck, N., Gerhards-Padilla, E., Martini, P. 2009. Modeling mobility in disaster area scenarios. *Performance Evaluation*, 66(12): 773 – 790. Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks.

Asuquo, P., Cruickshank, H., Ogah, C. P. A., Lei, A., Sun, Z. 2018. A distributed trust management scheme for data forwarding in satellite dtn emergency communications. *IEEE Journal on Selected Areas in Communications*, 36(2): 246–256.

Box, J. F. 1987. Guinness, gosset, fisher, and small samples. *Statistical Science*, 2(1): 45–52.

Burgess, J., Gallagher, B., Jensen, D., Levine, B. N. 2006. Maxprop: Routing for vehicle-based disruption-tolerant networks. En: *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, April. pp. 1–11.

Caragiannis, I., Galdi, C., Kaklamanis, C. 2005. *Basic Computations in Wireless Networks*, pp. 533–542. Springer Berlin Heidelberg, Berlin, Heidelberg.

CIC. 2013. Riesis, sistema de atención a contingencia sísmica. Consultado el 27 de Junio de 2013, de `http://riesis.cic.ipn.mx/RieSis`.

CICESE. 2013. Resnom. Consultado el 27 de Junio de 2013, de `http://resnom.cicese.mx/`.

Clementi, A., Silvestri, R., Trevisan, L. 2015. Information spreading in dynamic graphs. *Distributed Computing*, 28(1): 55–73.

Conover, W. 1999. *Practical nonparametric statistics*. Wiley series in probability and statistics: Applied probability and statistics. Wiley.

Cooperativa.cl 2011. Cantidad de llamadas tras el terremoto triplicaron las conexiones de año nuevo. Consultado el 6 de Junio de 2018, de `http://archive.is/G6od`.

Cormen, T. H., Stein, C., Rivest, R. L., Leiserson, C. E. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education, segunda edición.

Dainotti, A., Squarcella, C., Aben, E., Claffy, K. C., Chiesa, M., Russo, M., Pescapé, A. 2011. Analysis of country-wide internet outages caused by censorship. En: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, New York, NY, USA. ACM, IMC '11, pp. 1–18.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–271.

Ferrante, M. Saltalamacchia, M. 2014. The Coupon Collector ' s Problem. Reporte técnico, Universitat Autònoma de Barcelona, Barcelona.

Google. 2005. Google crisis response. Consultado el 19 de Mayo de 2017, de `http://www.google.com/crisisresponse/`.

Gross, J. Ligges, U. 2015. *nortest: Tests for Normality*. R package version 1.0-4.

Heinemann, A. 2007. *Collaboration in opportunistic networks*. Tesis de doctorado, TU Darmstadt, Fachbereich Informatik.

Hopcroft, J. E. Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.

Hossmann, T., Legendre, F., Carta, P., Gunningberg, P., Rohner, C. 2011. Twitter in disaster mode: Opportunistic communication and distribution of sensor data in emergencies. En: *Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition*, New York, NY, USA. ACM, ExtremeCom '11, pp. 1:1–1:6.

Huang, C.-M., chan Lan, K., Tsai, C.-Z. 2008. A survey of opportunistic networks. En: *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, march. pp. 1672 –1677.

ITU. 2016. Measuring the information society report 2016. Consultado el 19 de Mayo de 2017, de `http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2016/Mobile_cellular_2000-2015.xls`.

Johnson, D. B. Maltz, D. A. 1996. *Dynamic Source Routing in Ad Hoc Wireless Networks*, pp. 153–181. Springer US, Boston, MA.

Kaviani, M., Kusy, B., Jurdak, R., Bergmann, N., Liu, V. 2016. Energy-aware forwarding strategies for delay tolerant network routing protocols. *Journal of Sensor and Actuator Networks*, 5(4).

Keränen, A., Ott, J., Kärkkäinen, T. 2009. The one simulator for dtn protocol evaluation. En: *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Simutools '09, pp. 55:1–55:10.

Knuth, D. E. 1969. *The Art of Computer Programming: Sorting and Searching*, Vol. 3. Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.

Kuhn, F. Oshman, R. 2011. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1): 82–96.

Kuosmanen, P. 1999. Classification of ad hoc routing protocols. Reporte técnico, Finnish Defence Forces Naval Academy.

Kwan, M.-P. Lee, J. 2005. Emergency response after 9/11: the potential of real-time 3d gis for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems*, 29(2): 93–113.

LeBrun, J., Chuah, C.-N., Ghosal, D., Zhang, M. 2005. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. En: *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*. Vol. 4, pp. 2289–2293 Vol. 4.

Lilien, L., Kamal, Z. H., Bhuse, V., Gupta, A., (wireless Sensornet Lab, W. 2006. Opportunistic networks: The concept and research. En: *Challenges in Privacy and Security," Proc. NSF Intl. Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN 2006*.

Lilien, L., Kamal, Z. H., Bhuse, V., Gupta, A. 2007. *The Concept of Opportunistic Networks and their Research Challenges in Privacy and Security*, pp. 85–117. Springer US, Boston, MA.

Lilliefors, H. W. 1967. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318): 399–402.

Lindgren, A., Doria, A., Schelén, O. 2004. Probabilistic routing in intermittently connected networks. En: P. Dini, P. Lorenz, y J. de Souza (eds.), *Service Assurance with Partial and Intermittent Resources*, Vol. 3126 de *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 239–254.

Martín-Campillo, A., Crowcroft, J., Yoneki, E., Martí, R. 2013. Evaluating opportunistic networks in disaster scenarios. *Journal of Network and Computer Applications*, 36(2): 870 – 880.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F. 2015. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.6-7.

Mirai Solutions GmbH 2016a. *XLConnect: Excel Connector for R*. R package version 0.2-12.

Mirai Solutions GmbH 2016b. *XLConnectJars: JAR Dependencies for the XLConnect Package*. R package version 0.2-12.

Miranda, K., Molinaro, A., Razafindralambo, T. 2016. A survey on rapidly deployable solutions for post-disaster networks. *IEEE Communications Magazine*, 54(4): 117–123.

OpenGarden. 2014. Firechat. Consultado el 06 de Junio de 2018, de `http://www.opengarden.com/firechat.html`.

Pelusi, L., Passarella, A., Conti, M. 2006. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11): 134 –141.

Poonguzharselvi, B. Vetriselvi, V. 2013. Survey on routing algorithms in opportunistic networks. En: *2013 International Conference on Computer Communication and Informatics*, Jan. pp. 1–5.

R Core Team 2016. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rajagopalan, S. Shah, D. 2011. Distributed averaging in dynamic networks. *IEEE Journal of Selected Topics in Signal Processing*, 5(4): 845–854.

Ramesh, M. V., Jacob, A., Aryadevi, R. 2012. Participatory sensing platform to revive communication network in post-disaster scenario. En: *Wireless and Optical Communications Conference (WOCC), 2012 21st Annual*, april. pp. 118 –122.

Reina, D. G., Askalani, M., Toral, S. L., Barrero, F., Asimakopoulou, E., Bessis, N. 2015. A survey on multihop ad hoc networks for disaster response scenarios. *Int. J. Distrib. Sen. Netw.*, 2015: 3:3–3:3.

Rodriguez-Silva, D., Costa, A., Macedo, J. 2012. Energy impact analysis on dtn routing protocols. En: *Proc. ExtremeCom*, Marzo. Zurich, Switzerland, pp. 1–6.

Rolim, C. O., Rossetto, A. G., Leithardt, V. R. Q., Borges, G., Geyer, C., dos Santos, T. F. M., Souza, A. M. 2016. Situation awareness and computational intelligence in opportunistic networks to support the data transmission of urban sensing applications. *Computer Networks*, 111: 55–70. doi = 10.1016/j.comnet.2016.07.014.

Rosas, E., Hidalgo, N., Gil-Costa, V., Bonacic, C., Marin, M., Senger, H., Arantes, L., Marcondes, C., Marin, O. 2016. Survey on simulation for mobile ad-hoc communication for disaster scenarios. *Journal of Computer Science and Technology*, 31(2): 326–349.

Saha, S., Sushovan, Sheldekar, A., Joseph C., R., Mukherjee, A., Nandi, S. 2011. Post disaster management using delay tolerant network. *Communications in Computer and Information Science*, 162 CCIS: 170–184.

Sepúlveda, P., Ramiro, V., Barros, T., Piquer, J. 2010. Soundness of chilean internet routes. En: *Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the*. pp. 298–306.

Sghaier, N., Augustin, B., Mellouk, A. 2014. On enhancing network-lifetime in opportunistic wireless sensor networks. En: *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, April. pp. 2617–2622.

Snedecor, G. Cochran, W. 1989. *Statistical methods*. Número v. 276 en: Statistical Methods. Iowa State University Press. p. 503.

Sobin, C. C., Raychoudhury, V., Marfia, G., Singla, A. 2016. A survey of routing and data dissemination in delay tolerant networks. *Journal of Network and Computer Applications*, 67: 128 – 146.

Sobin, C. C., Raychoudhury, V., Saha, S. 2017. An energy-efficient and buffer-aware routing protocol for opportunistic smart traffic management. En: *Proc. ICDCN*. Hyderabad, India, pp. 25:1–25:8.

Spyropoulos, T., Psounis, K., Raghavendra, C. S. 2005. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. En: *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, New York, NY, USA. ACM, WDTN '05, pp. 252–259.

Spyropoulos, T., Psounis, K., Raghavendra, C. 2007. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. En: *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, March. pp. 79–85.

Spyropoulos, T., Psounis, K., Raghavendra, C. 2008. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *Networking, IEEE/ACM Transactions on*, 16(1): 77–90.

Stadje, W. 1990. The collector's problem with group drawings. *Advances in Applied Probability*, 22(4): 866–882.

Tanenbaum, A. Wetherall, D. 2011. *Routing algorithms*, pp. 362–363. Pearson Prentice Hall, New Jersey, USA.

Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., Sheu, J.-P. 2002. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.*, 8(2/3): 153–167.

Urbanek, S. 2016. *rJava: Low-Level R to Java Interface*. R package version 0.9-8.

Vahdat, A. Becker, D. 2000. Epidemic routing for partially-connected ad hoc networks. Reporte técnico, Duke University.

Vargas, E. 2013. Presentó cicese aplicación sísmica android. Consultado el 27 de Junio de 2013, de `http://www.ensenada.net/noticias/nota.php?id=28635`.

Venables, W. N. Ripley, B. D. 2002. *Modern Applied Statistics with S*. Springer, cuarta edición. New York. ISBN 0-387-95457-0.

Wang, K., Shao, Y., Shu, L., Sun, Y., He, L. 2015. An improved spray and wait algorithm based on rvns in delay tolerant mobile sensor networks. En: *2015 IEEE International Conference on Communications (ICC)*, June. pp. 3552–3556.

Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83.

Xiao, L. Boyd, S. 2004. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1): 65 – 78.

Yan, C., Li, J., Zhao, W. J., Qu, X., Zhao, X. 2010. Design and development of disaster rapid mapping system in case of earthquake. En: *2010 International Conference on Multimedia Technology*. ICMT 2010, pp. 1 –4.

Anexo A. Algoritmo de distribución de mensajes

En este anexo se describe la versión preliminar del algoritmo EDEN descrito en el Capítulo 4. A este algoritmo preliminar se le denominó 'algoritmo de distribución homogénea de mensajes' (Alg-DHM). Ya que originalmente el objetivo de este algoritmo era resolver el problema DMHM. Inicialmente se describe el funcionamiento general del algoritmo Alg-DHM. Después, se definen un par de contadores utilizados en Alg-DHM para llevar un control del número de copias de los mensajes. Por último, se presenta un análisis de la propagación de un mensaje utilizando un árbol de dispersión de un mensaje.

A.1. Descripción general del algoritmo

Para explicar el algoritmo Alg-DHM se establecen las siguientes suposiciones: 1) Los vértices a y b entran en contacto. 2) Inicialmente el vértices a actúa como maestro y el vértice b como esclavo. 3) El vértices a calcula previamente su resumen de mensajes R_a. El proceso del intercambio de los mensajes del algoritmo Alg-DHM se presenta en el Pseudocódigo 8. A continuación se describe este pseudocódigo.

En el paso 1 inicia la ejecución del procedimiento recibiendo como parámetro el resumen de los mensajes, R_a, generado por el propio vértice a. En el paso 2, el vértice a solicita el resumen de sus mensajes, R_b, al vértice b. En el paso 3 el vértice a calcula los mensajes candidatos que desea recibir, C_a, a partir de los mensajes en R_b no presentes en R_a. En el paso 4 se evalúa si C_a no está vacío; si ésto se cumple, entonces continua en el paso 5; de lo contrario, termina la ejecución de este procedimiento. En el paso 5 se determinan los mensajes a solicitar, S_a, aplicando una política de intercambio donde se selecciona un subconjunto S_a de C_a tal que no sobrepase el tamaño del buffer κ_a. En el paso 6 se evalúa si S_a no está vacío; si esto se cumple, entonces continua en el paso 7; de lo contrario, termina la ejecución de este procedimiento. En el paso 7, el vértice a solicita al vértice b el envío de los mensajes en S_a. Al finalizar este procedimiento se vuelve a ejecutar pero ahora con los roles intercambiados, es decir, el vértice b ahora es el maestro y el vértice a el esclavo.

Pseudocódigo 8 Alg-DHM message exchange

```
1: procedure MessageExchange(R_a)
2:     requests R_b to vertex b
3:     C_a ← R_b \ R_a
4:     if C_a ≠ ∅ then
5:         S_a ← {x ∈ C_a : |S_a ∪ R_a| ≤ κ_a}          ▷ simple exchange policy
6:         if S_a ≠ ∅ then
7:             request the messages in S_a to vertex b
8:         end if
9:     end if
10: end procedure
```

Además del intercambio de mensajes, el algoritmo Alg-DHM utiliza dos contadores para poder controlar el número de copias que se tienen por cada mensaje. Estos contadores se explican a continuación.

A.1.1. Contador origen y contador copia

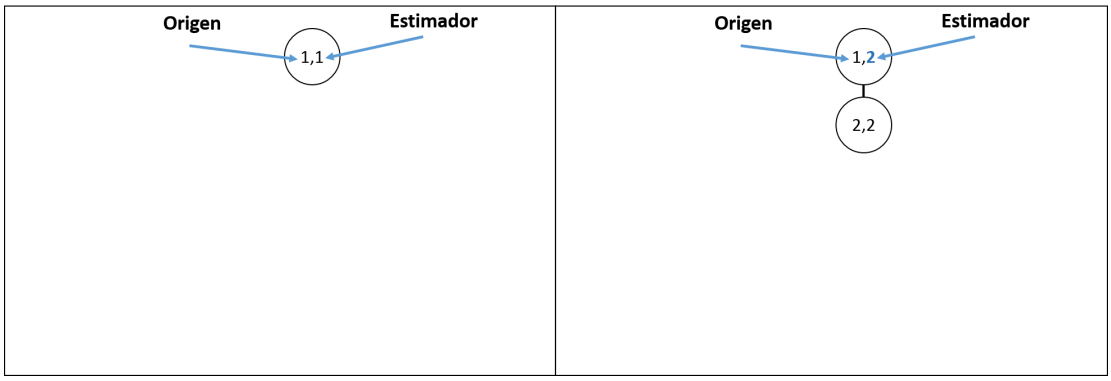
El algoritmo Alg-DHM utiliza un par de contadores incluidos en cada mensaje. Estos contadores se utilizan para la toma de decisiones y se actualizan al momento de crear una nueva copia de un mensaje. El primer contador, llamado ‘contador origen’, se le asigna un valor igual al número de copia en el momento que se origina la copia del mensaje. El segundo contador, llamado ‘contador copia’, es un estimador local del número de copias del mensaje, es decir, toma un valor igual al número de copias más uno. Este valor representa el número de copias que el vértice sabe con certeza que se crearon antes, más su copia recién creada.

Inicialmente, cuando se crea un mensaje, se utilizan las siguiente reglas. A la primer copia de un mensaje se le asigna un valor de uno a su contador origen y su contador copia. El contador origen no cambia una vez que se le asigne un valor. Por el contrario, el contador copia se actualiza en el vértice emisor cada que se envía el mensaje.

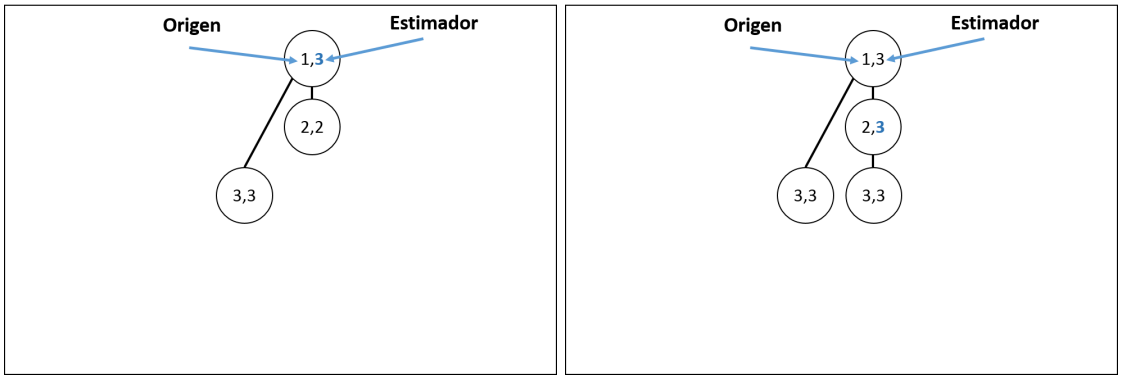
Específicamente, al enviar un mensaje, el vértice emisor actualiza el contador copia incrementando su valor en uno por cada envío. Mientras que el vértice receptor, fija ambos contadores igual al estimador del vértice emisor más uno. Para clarificar este proceso se utiliza una representación visual en forma de árbol, la cual se explica en la siguiente sección.

A.1.2. Árbol de dispersión de un mensaje

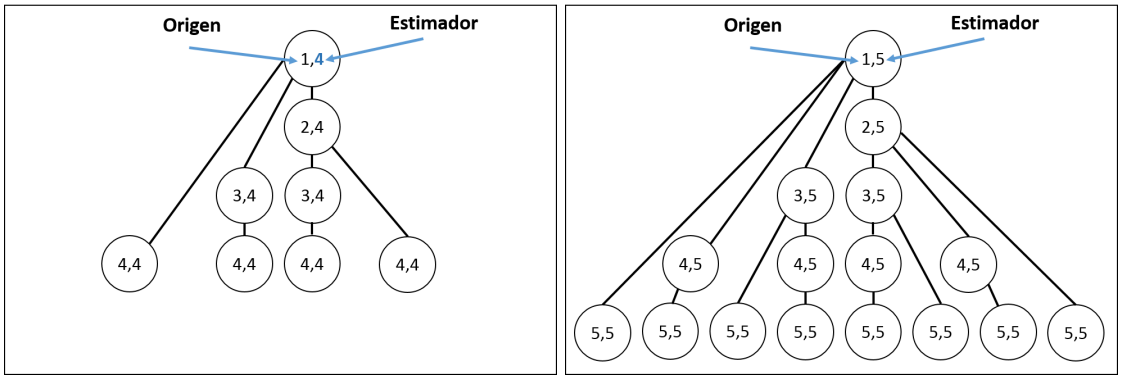
En esta sección se define el árbol de dispersión de un mensaje. Este árbol permite visualizar cómo se propagan las copias de un mensaje, al mostrar los cambios en el contador origen y el contador de copias (estimador de copias). Este árbol fue el precursor del árbol binario de transmisión introducido en la sección 4.2 ya que ambos se utilizan para comprender la distribución de copias de un mensaje.



(a) Nodo inicial del árbol. **(b)** Segundo nodo del árbol.



(c) Tercer nodo del árbol. **(d)** Cuarto nodo del árbol.



(e) Todos los nodos generados con x=4. **(f)** Todos los nodos generados con x=5.

Figura 30. Crecimiento del árbol de dispersión de un mensaje con un máximo número de origen x=5. El par ordenado numérico dentro de cada nodo representa el número de origen y el número estimado de copias del mensaje. En negritas se denota el aumento en el número estimador al crear un nodo.

Tabla 4. Total de número de nodos (copias) en el árbol dado el número máximo de la copia origen.

Máximo valor de copia origen (x)	1	2	3	4	5	6
Número de nodos en el árbol (α)	1	2	4	8	16	32

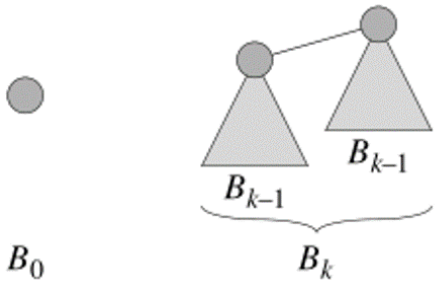
Cuando se examina un mensaje en particular con las copias que se generan de este mensaje y los cambios que ocurren en sus contadores origen y copia, se obtiene el árbol de dispersión de un mensaje, el cual se ilustra en la Figura 30.

Es interesante el crecimiento que se observa en el árbol de la Figura 30, ya que indica cómo al limitar el nivel se puede limitar el número de copias de un mensaje. Además, que puede calcularse el número de nodos en cierto nivel del árbol, i.e., el número de copias con el mismo contador de origen.

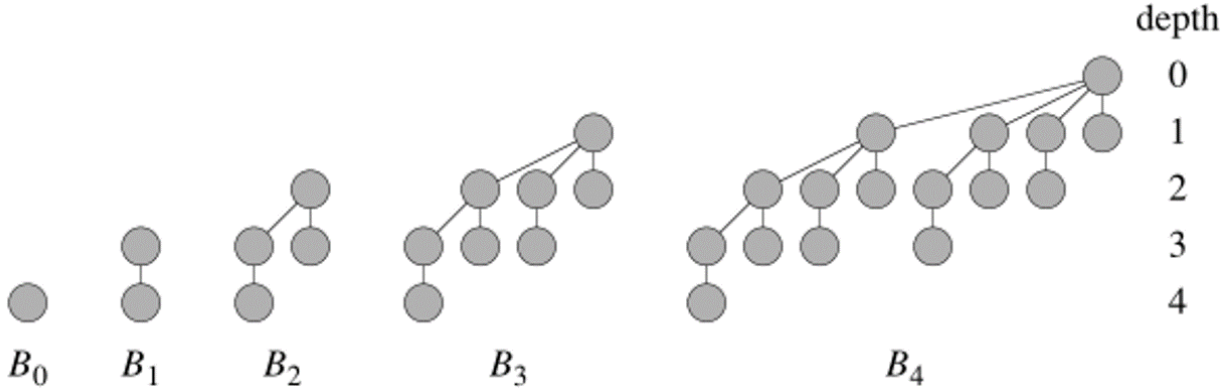
Sea x el máximo valor de la copia origen. Sea α el número de nodos en el árbol, i.e., el número de copias potenciales en la red. Por el crecimiento del árbol es claro que $\alpha = 2^{x-1}$, por lo tanto $x = \log_2(\alpha) + 1$. Utilizando esta fórmula es posible limitar el número de copias de un mensaje. Un ejemplo de esto se muestra en la Tabla 4, donde se observa cuantas copias (α) a lo más se generarían al limitar a un valor fijo el número de copia origen. El problema con este tipo de enfoque es que sólo se puede limitar el número de copias máximo a valores que son pares, lo cual es una limitación considerable. Se le denomina ‘árbol desbalanceado’ a un árbol de dispersión de un mensaje si sus nodos no tienen el mismo número de copia. Dentro de los arboles de la Figura 30, la Figura 30c es la única que muestra un árbol desbalanceado.

Suponga que se quieren analizar las relaciones entre las copias que se hacen de un mensaje. Este árbol puede parecer familiar a otra estructura de datos, por lo que surge la siguiente pregunta: ¿Existe en la literatura un árbol igual o similar a éste?. La respuesta es sí, existen árboles muy similares. A continuación se describe uno de ellos.

Un árbol binomial es un tipo de árbol que se puede definir recursivamente. El árbol binomial B_0 tiene 1 nodo. El árbol binomial B_k consiste en dos árboles binomiales B_{k-1} unidos por medio de una arista entre sus raíces como se muestra en la Figura 31a (Cormen *et al.*, 2001). Es decir, B_k es un árbol con $k-1$ subárboles (Knuth, 1969), representados por $B_0, B_1, \ldots, B_{n-1}$ con $k > 0$ (ver Figura 31b).



(a) Árbol binomial B_0 y árbol binomial B_k ilustrando su recursividad.



(b) Árboles binomiales B_0, B_1, B_2, B_3, y B_4.

Figura 31. Ejemplos del árbol binomial. Imágenes sustraídas de Cormen *et al.* (2001).

Se puede representar visualmente el árbol en la Figura 30f, el cual se generó utilizando los contadores origen y copia de un mensaje con $x = 5$, para que se vea igual a un árbol binomial B_4 (ver Figura 32). La ventaja de utilizar este tipo de árboles es que es una estructura bien estudiada y cuenta con propiedades interesantes que pueden ser de utilidad para un potencial algoritmo de distribución de mensajes homogéneo que utilice estos contadores. Algunas de las propiedades del árbol binomial son las siguientes:

1. Tiene 2^k vértices.
2. La altura del árbol es k.
3. Tiene exactamente $\binom{k}{i}$ nodos en la profundidad i, con $i = 0, 1, \ldots, k$.
4. La raíz tiene grado k, el cual es el mayor de cualquier nodo.
5. El grado máximo de cualquier vértice de n nodos es $\log n$.

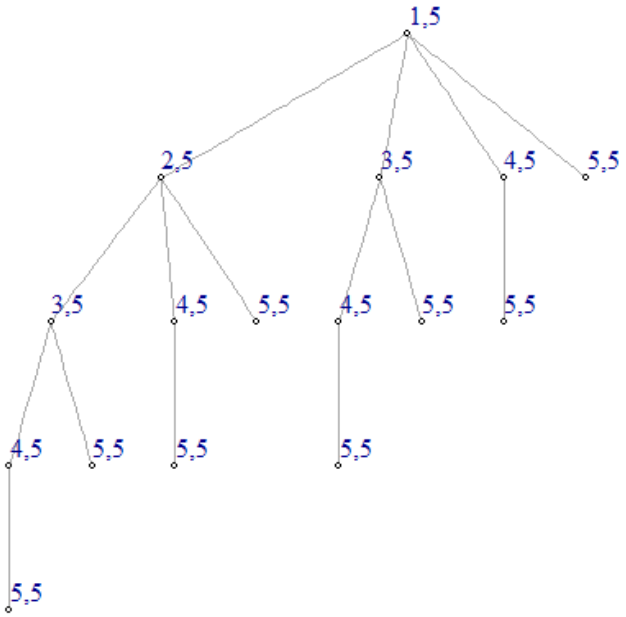


Figura 32. Árbol de dispersión de un mensaje con $x = 5$ representado visualmente como un árbol binomial B_4.

A continuación se muestra un mapeo uno a uno de las propiedades del árbol binomial con la información que se puede obtener y ser de utilidad en el escenario del problema DMHM:

1. Máximo número de copias de un mensaje.
2. Máximo número de saltos de una misma copia.
3. El número de copias que tienen i saltos.
4. El vértice que genera el mensaje es el que puede realizar el mayor número de copias del mensaje.
5. Si se quiere tener n copias de un mensaje, el número de copias máximo que puede generar un vértice es $\log n$.

También se podría guardar el número de salto que ha dado la copia de un mensaje para obtener la profundidad de un árbol binomial. Este número de salto puede servir como criterio de desempate para definir qué nodo del árbol binomial eliminar, es decir, qué copia en la red eliminar. Si se encuentra una forma de etiquetado único para cada nodo del árbol binomial se podría controlar y depurar el número de copias más

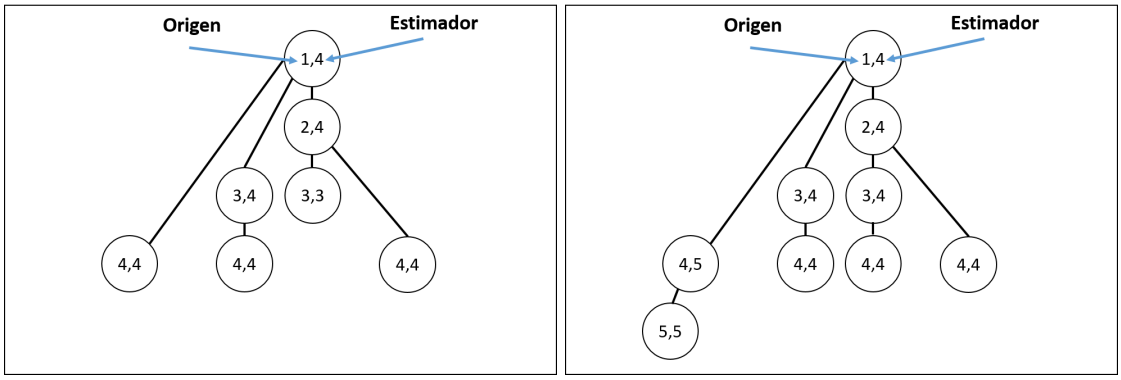
fácilmente. Una forma de generar un etiquetado único es guardar los padres o nodos por los que ha pasado una copia.

A.1.3. Utilidad del contador origen y el contador copia

En esta sección se describe la función que cumplen los contadores en la toma de decisiones dentro del algoritmo Alg-DHM. En el paso 5 del Pseudocódigo 8 se menciona que se determinan los mensajes a solicitar aplicando una política de intercambio. Inicialmente se utiliza una política de intercambio voraz, es decir, se solicitan todos los mensajes que no tenga el vértice que solicita y además, que éste los pueda almacenar en su memoria. En caso de tener múltiples mensajes solicitados por un vértice, la política de intercambio ordena dando prioridad a aquellos mensajes que tengan contador de origen más pequeño y como segundo criterio de desempate prioriza los que tengan el contador de copia más pequeño. Con la finalidad de fomentar que sobrevivan los mensajes con menos copias en la red y se eliminen los que tienen muchas copias, i.e., homogeneizar el número de copias en la red.

Sin embargo, se identificaron algunos casos particulares donde no es posible homogeneizar los mensajes a pesar de tener valores grandes en sus contadores. Por ejemplo, suponga que todos los vértices tienen sus buffers saturados. Además, suponga que el número de copias óptimo en la red es cuatro; por lo tanto, idealmente el árbol de dispersión de un mensaje para todos los mensajes en la red, debería verse como el de la Figura 30e. Finalmente, suponga que todos los mensajes tienen un contador de copia igual a cuatro excepto dos copias, una del mensaje m_i y otra de m_j. La copia del mensaje m_i (m_j) tiene los contadores origen y copia de $3, 3$ ($5, 5$) como se ilustra en la Figura 33a (Figura 33b). Por lo cual m_i necesita una copia más y m_j necesita eliminar una copia. El problema surge si los dos mensajes están almacenados en el mismo vértice, ya que el algoritmo Alg-DHM no podría reemplazar el mensaje excedente de m_j con el mensaje faltante de m_i. En casos extremos si no se controla este tipo de situaciones se puede tener árboles con ramas muy cortas y otros árboles con ramas muy largas.

Para evitar este tipo de situaciones se propone una segunda etapa de intercambio, en esta etapa no hay borrado de mensajes sino solo intercambio de mensajes. El objetivo de esta etapa es mezclar ciertos mensajes para que el reemplazo por contadores



(a) Árbol con una copia faltante. **(b)** Árbol con una copia excedente.

Figura 33. Ejemplos de árboles de dispersión de un mensaje desbalanceados.

eventualmente elimine aquellos mensajes con contadores potencialmente muy grandes. Para ejecutar esta segunda etapa de intercambio se tienen que cumplir ciertas condiciones. Las cuales se explican a continuación.

Suponga que dos vértices establecen contacto y que sus buffers están saturados. Se tiene que encontrar un par de mensajes, uno en cada vértice, que satisfagan lo siguiente: uno de los mensajes tiene un valor de contador origen muy grande (máximo y único entre todos los mensajes en los dos vértices) y el otro mensaje debe tener un valor de contador origen muy pequeño (mínimo y único entre todos los mensajes en los dos vértices). Si se cumplen estas condiciones, simplemente se intercambian los mensajes y se habilitan para borrar mensajes por reemplazo utilizando el contador de origen para determinar que mensaje remover.

Anexo B. Experimentos preliminares

En este anexo se muestra inicialmente dos rondas de experimentos preliminares que se realizaron para evaluar las limitaciones y capacidades del simulador ONE al modificar los parámetros iniciales. En todos los experimentos de estas primeras dos rondas se utiliza un algoritmo de enrutamiento de inundación epidémico, con la finalidad de establecer la potencialmente máxima entrega de mensajes, ya que este algoritmo maximiza la entrega de mensajes a expensas de los recursos.

Posteriormente, se presentan otras dos rondas de experimentos pero ahora con la finalidad de comparar el algoritmo Alg-DHM (descrito en el Anexo A) con el algoritmo de enrutamiento de inundación epidémico en diferentes escenarios.

Para las dos primeras rondas, los experimentos presentados establecen las siguientes suposiciones:

- El número de vértices, n, en la red es constante.
- Se genera solamente un mensaje por vértice en el tiempo $t = 0$.
- El vértice destino de los mensajes no se encuentra en la red.
- Se utiliza una política de intercambio de mensajes cooperativa, i.e., se intercambian todos los mensajes posibles que permita la interfaz de comunicación y la capacidad de almacenamiento de los vértices.
- Se utiliza un TTL (*Time To Live*) para los mensajes con valores muy grandes para evitar borrar mensajes por antigüedad.
- El tamaño del buffer de cada vértice permite almacenar todos los mensajes distintos en la red, i.e., $\kappa_i = n, \forall i$.
- Solamente se puede tener una copia por mensaje en la memoria de cada vértice.
- El estado de contacto se establece con un vértice a la vez.

Tabla 5. Parámetros para los experimentos preliminares.

Parámetro	Valor
Número total de vértices (n)	6
Número de grupos de vértices	2
Número de vértices en el grupo 1 (G1)	4
Número de vértices en el grupo 2 (G2)	2
Modelo de movilidad de G1	Random waypoint
Modelo de movilidad de G2	Random walk
Radio de transmisión	10 m
Velocidad de transmisión	250 kBps
Velocidad de movimiento	0.5 - 1.5 m/s
Semillas aleatorias	10
Tamaño de mensaje	1 kB
Tamaño del buffer (κ_i)	6 kB
Creación de mensajes	Un mensaje por vértice en $t = 0$
Tiempo de simulación	100,000 segundos
Esquema de energía	Sin restricciones
Tamaño del escenario	600 m × 400 m

B.1. Primera ronda de experimentos preliminares

En esta sección se muestran gráficas de la primera ronda de experimentos preliminares. Estos experimentos son principalmente exploratorios, en el sentido que se evalúa si se puede obtener información relevante acerca del funcionamiento del algoritmo epidémico implementado en el simulador y de los límites propios del simulador ONE.

Primero se presentan los tiempos de saturación de memoria de cada uno de los vértices utilizando el algoritmo epidémico. En teoría, estos tiempos son los más pequeños que se pueden obtener para este escenario. Después se presentan algunas gráficas del número y tiempo de contacto entre los vértices. La Tabla 5 muestra los parámetros utilizados para la primera ronda de experimentos preliminares.

B.1.1. Análisis del tiempo de saturación de memoria de los vértices

En esta sección se presentan los experimentos preliminares que permiten analizar qué tan rápido un algoritmo de ruteo epidémico basado en inundación ya implementado en ONE puede llenar la memoria de todos los dispositivos en la red.

Como se describe en la sección 5.1.3, Ω_n es la colección de todos los mensajes distintos en la red. En el renglón 13 de la Tabla 5 se establece que cada vértice genera un mensaje. Por lo tanto, $|\Omega_n| = n$.

Uno de los parámetros más relevantes en el análisis de la saturación de la memoria es el tamaño del buffer. Se estudian dos casos al modificar este parámetro. El primero, cuando se tiene un número de mensajes menor o igual al espacio de almacenamiento de cada vértice, i.e., $|\Omega_n| \leq \kappa_i$, este caso es trivial, ya que cualquier algoritmo de enrutamiento de inundación podría resolver fácilmente el problema de DMHM con la misma rapidez que otros. El otro caso es más interesante, si se considera que existen más mensajes en la red de los que puede almacenar un vértice en su memoria, i.e., $|\Omega_n| > \kappa_i$. Para esta caso es necesario utilizar mecanismos y políticas de borrado para asegurarse que se obtiene una cantidad homogénea de mensajes en la red. En estos experimentos se utiliza un número de mensajes igual al espacio de almacenamiento de memoria. El fijar este parámetro permite establecer cuál es el menor tiempo posible que un algoritmo de enrutamiento necesita para llenar los buffers de todos los vértices en la red.

Otro parámetro que se pretende analizar en estos experimentos es el del modelo de movilidad. Los modelos aleatorios de movilidad que se consideran son *random walk* y *random waypoint*. Al ser los dos aleatorios, en teoría, si se utiliza solamente uno de ellos para la movilidad de los vértices, se deben obtener resultados similares. Otra forma de modificar la movilidad es utilizando mapas en el escenario en lugar de un escenario sin caminos predefinidos. Esto agrega realismo y lo enfoca a una ciudad en particular, pero quita libertad de movimiento y generalidad a la simulación. Este último aspecto de utilizar mapas se llevó a cabo en los experimentos finales presentados en el Capítulo 6.

La Tabla 6 muestra los tiempos de saturación de cada vértice. Denotando los vértices que utilizan el modelo de movilidad *random waypoint* (v_1,v_2,v_3, y v_4) y los que utilizan el modelo de movilidad *random walk* (v_5 y v_6). Además, las tres últimas columnas de la derecha muestran el tiempo de saturación promedio, máximo y mínimo de toda la red.

En la Tabla 6 se puede observar que las diferencias entre los tiempos de saturación de *random waypoint* y *random walk* son muy pequeñas entre sí. Por lo cual es necesario evaluar estos modelos de movilidad con diferentes valores de *n* para obtener mayores conclusiones.

Tabla 6. Resultados de los tiempos de saturación.

Núm. de ejecución	Random waypoint				Random walk		Tiempos de saturación		
	v_1	v_2	v_3	v_4	v_5	v_6	Promedio	Máximo	Mínimo
1	12,362	15,308	8,896	13,354	10,705	8,901	11,587.67	15,308	8,896
2	7,768	7,523	5,657	5,658	8,581	6,618	6,967.50	8,581	5,657
3	20,771	18,177	12,036	12,041	17,373	12,071	15,411.50	20,771	12,036
4	7,345	8,331	7,326	7,331	8,037	7,763	7,688.83	8,331	7,326
5	14,636	16,270	16,092	12,661	12,660	18,337	15,109.33	18,337	12,660
6	6,814	7,771	9,985	10,970	8,497	6,812	8,474.83	10,970	6,812
7	9,893	8,018	8,419	6,771	6,009	6,011	7,520.17	9,893	6,009
8	17,645	17,300	16,527	14,193	11,493	11,495	14,755.50	17,645	11,493
9	1,625	8,624	6,877	1,628	4,281	1,809	4,140.50	8,624	1,624
10	7,329	7,328	11,176	8,924	9,349	7,431	8,589.50	11,176	7,328
Promedio	10,619	11,465	10,299	9,353	9,699	8,725	10,027.00	12,964	7,984
Máximo	20,771	18,177	16,527	14,193	17,373	18,337	15,412.00	20,771	12,660
Mínimo	1,624	7,328	5,657	1,628	4,281	1,809	4,141.00	8,331	1,624
Desv. Est.	5,764	4,631	3,709	4,006	3,658	4,437	3,944.69	4,634	3,390

La Figura 34 se obtiene al graficar la ejecución 1 de la Tabla 6. Esta figura ilustra el tiempo de simulación en el cual cada vértice en la red llenó su buffer en esta ejecución. Para esta ejecución, en promedio los vértices saturan sus buffers aproximadamente en $11,500$ segundos.

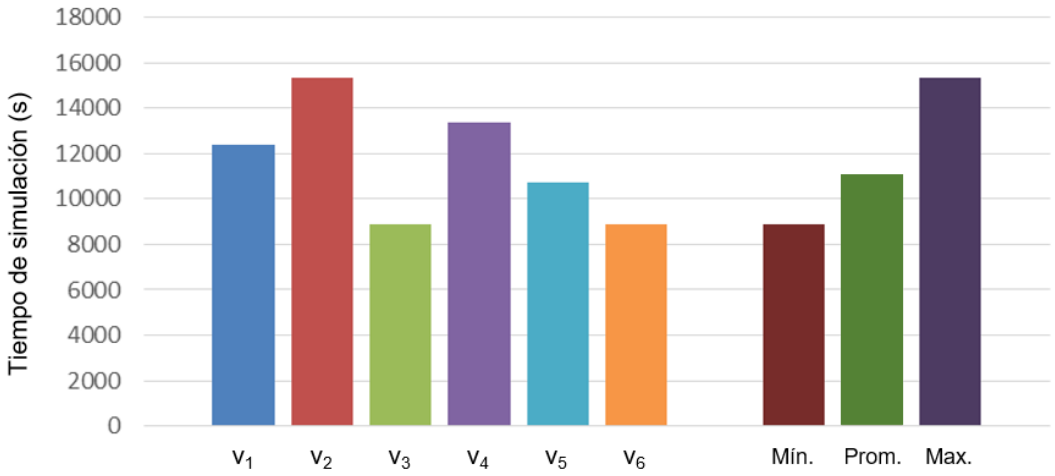


Figura 34. Tiempo de simulación que cada vértice tarda en alcanzar el límite de almacenamiento en su memoria para la primera ejecución (ver Tabla 6) de los experimentos preliminares. Las primeras seis barras de la izquierda son los tiempos de cada vértice. Las últimas tres barras muestran el promedio, el máximo y el mínimo de todos los vértices.

Con estos experimentos es posible concluir que el tiempo de saturación máximo de la red es de utilidad porque permite evaluar qué tan rápido un algoritmo de enrutamiento es capaz de llenar el buffer de todos los vértices en la red. Note que el tiempo de saturación de la red puede ser menor o igual al tiempo de convergencia (ver sección 5.1.4), dependiendo, entre otros factores, del algoritmo de enrutamiento. Esto es debido a qué algunos algoritmos de enrutamiento continúan intercambiando mensajes, incluso después de saturar sus buffers.

B.1.2. Análisis del número y duración de los contactos entre vértices

En esta sección se presentan los experimentos preliminares que permiten analizar el número y duración de los contactos entre todos los dispositivos en la red.

Un contacto ocurre cuando un par de vértices están dentro del área de cobertura de cada uno, i.e., la distancia entre ellos es lo suficientemente pequeña para que sus interfaces puedan detectarse mutuamente y potencialmente puedan intercambiar mensajes. Un vértice puede estar en contacto con muchos otros a la vez. La Figura 35 ilustra la frecuencia acumulada del número de contactos ordenados cronológicamente entre cualquier par de vértices. En la figura se puede observar que se generan 2,500 contactos entre todos los vértices durante los 100,000 segundos, es decir, en promedio ocurre un contacto cada 400 segundos. De lo anterior se puede deducir que el escenario es muy disperso, por lo que sería conveniente utilizar un valor de n más grande.

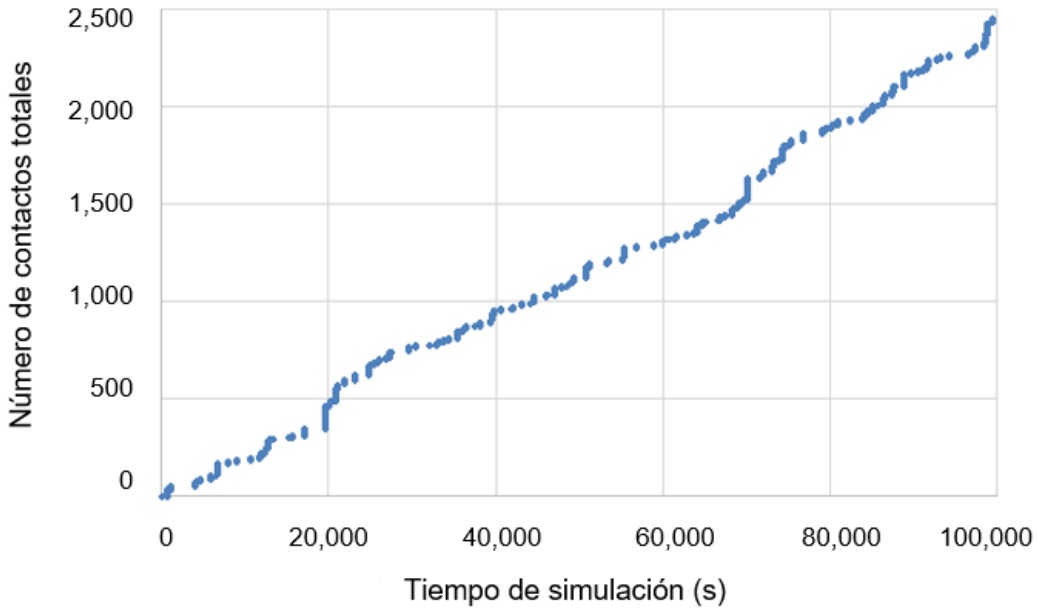


Figura 35. Frecuencia acumulada del número de contactos para una ejecución en el escenario preliminar.

La duración de un contacto es el intervalo de tiempo durante el cual dos vértices estuvieron en contacto. En la Figura 36 se muestra la frecuencia del tiempo de contacto para todos los vértices. En esta figura se puede observar que la mayoría de los contactos duran entre 8 y 15 segundos. Es decir, en este escenario se podrían intercambiar entre 8 y 15 mensajes por cada contacto.

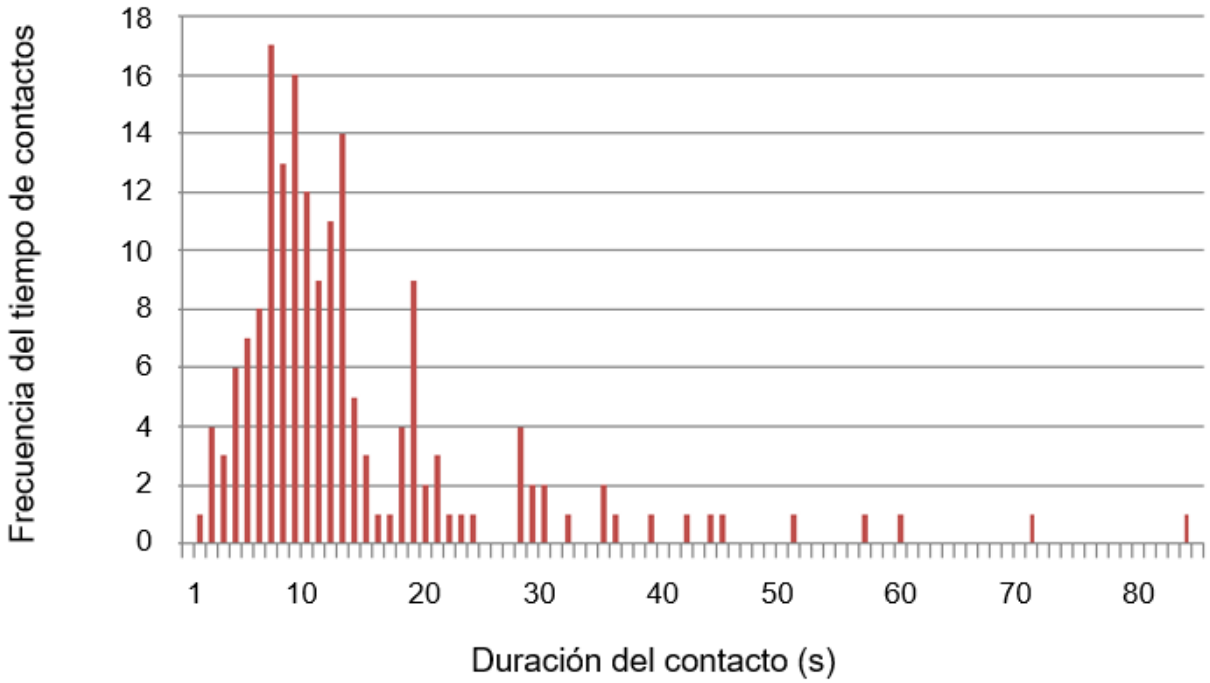


Figura 36. Frecuencia del tiempo de contacto entre todos los vértices.

Lo anterior se debe a que el simulador ONE utiliza ciclos de simulación de un segundo para el contacto entre los vértices; a pesar de que las suposiciones de la interfaz de comunicación, y el tamaño de cada mensaje (ver Tabla 5) permitirían, en teoría, enviar hasta 250 mensajes por segundo. Las limitaciones del simulador permiten que se mande a lo más un mensaje por segundo de contacto.

En la Figura 37 se presentan los instantes en que inician los diferentes contactos que el vértice v_6 tuvo ordenados cronológicamente desde $t = 0$ hasta t =80,000 s. En esta figura se puede apreciar que los intervalos de tiempo entre diferentes contactos de un mismo vértice son muy extensos; incluso en este escenario, un vértice nunca tiene más de un contacto a la vez.

Por otra parte, una interacción es cuando dos vértices están en contacto y además intercambian mensajes. Un vértice puede estar interactuando con un solo vértice a la vez (las simulaciones en ONE permiten solamente interacciones de tipo half-duplex).

La Figura 38 presenta el número de interacciones totales y el número de interacciones únicas (sin repetir) que tuvo cada vértice en la red. Como se puede observar todos tuvieron 5 interacciones únicas, es decir, todos los vértices interactuaron con todos durante la simulación. Por otra parte, en las interacciones totales se puede ob-

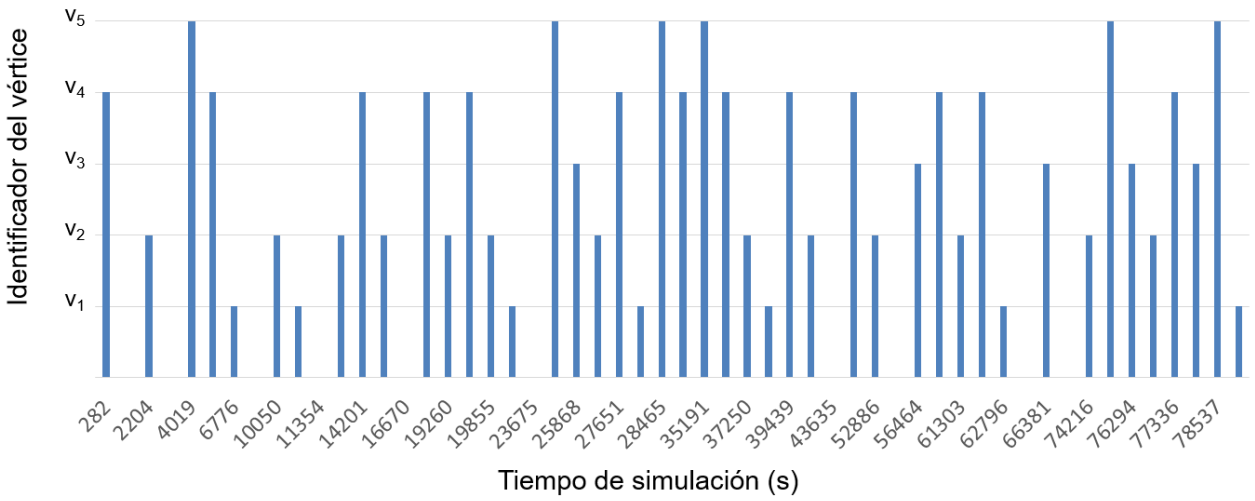


Figura 37. Instantes del inicio de los contactos para el vértice 6 (v_6) ordenados cronológicamente desde el inicio de la simulación hasta 80,000 s.

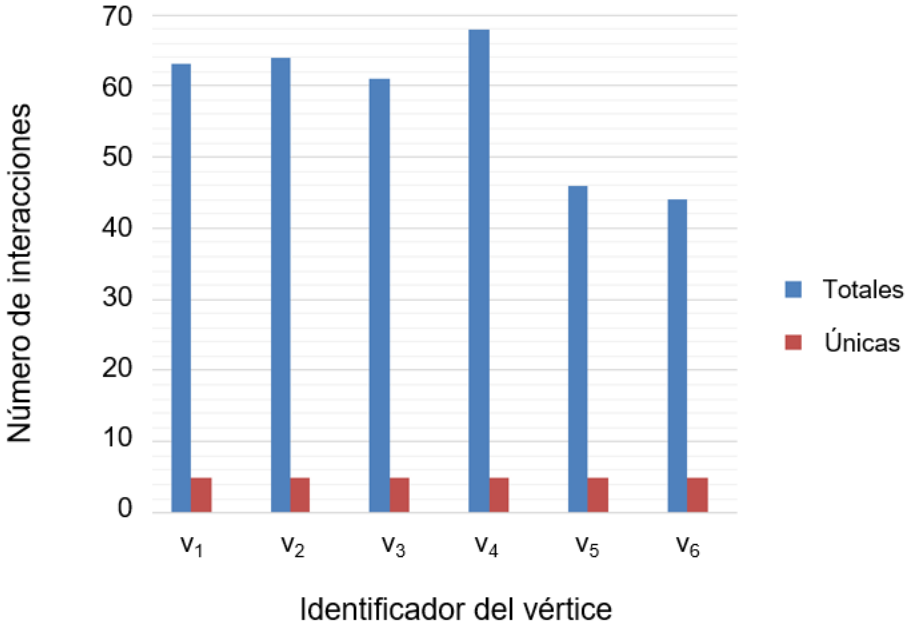


Figura 38. Número de interacciones totales e interacciones únicas por vértice en el escenario preliminar.

servar que los vértices v_5 y v_6 lograron menos interacciones que los demás vértices. Esta es una consecuencia directa del modelo de movilidad *random walk* utilizado por este par de vértices.

Analizar el número y duración de los diferentes contactos de los vértices permite entender algunas de las implicaciones de utilizar ciertos modelos de movilidad. Mientras que al analizar la interacción de los vértices ayuda a comprender el efecto del algoritmo de enrutamiento utilizado.

Tabla 7. Parámetros para la segunda ronda de experimentos preliminares.

Parámetro	Valor
Número total de vértices (n)	$10, 20, 40, 80, 160, 320, 640, 1280$
Modelo de movilidad	Random waypoint, Random walk
Radio de transmisión	10 m
Velocidad de transmisión	250 kBps
Velocidad de movimiento	0.5 - 1.5 m/s
Semillas aleatorias	100
Tamaño de mensaje	1 kB
Tamaño del buffer (κ_i)	n×1 kB
Creación de mensajes	Un mensaje por vértice en $t = 0$
Tiempo de simulación	50,000 segundos
Esquema de energía	Sin restricciones
Tamaño del escenario	590 m × 532 m

B.2. Segunda ronda de experimentos preliminares

En esta sección se presentan la segunda ronda de los experimentos preliminares. A diferencia de la primera, en esta ronda se aumentó el número de ejecuciones a 100 y se redujo el tiempo de simulación. Además, se utilizan diez valores distintos de n para evaluar el efecto ante distintas densidades de vértices. Finalmente, se utiliza un solo modelo de movilidad aleatorio, *random walk* o *random waypoint*, para todos los vértices durante cada ejecución. Con el objetivo de analizar el impacto de los modelos en los resultados de la simulación. La Tabla 7 muestra los parámetros de estos experimentos.

Una de las variables a analizar en esta segunda ronda de experimentos preliminares es el efecto de variar el número de vértices dentro del escenario. Por lo que se utiliza una gran variabilidad en el valor de n. La Figura 39 muestra la mayoría de los escenarios con diferentes valores de n. En esta figura se observa cómo la densidad de los vértices aumenta gradualmente. Otro aspecto que afecta al variar n es el número de vecinos que tiene cada vértice. En la sección B.2.3 se analiza con mayor profundidad el impacto al utilizar diferentes valores de n.

B.2.1. Comparación entre *random walk* y *random waypoint*

Uno de las parámetros a analizar en esta ronda de experimentos es el de los modelos de movilidad aleatorios. Para analizarlos, se diseñó un escenario particular (ver Figura 40): se colocaron vértices estáticos (puntos azules) que sólo sean receptores en forma de rejilla, de tal forma que no traslapen entre sus áreas de cobertura (círculo

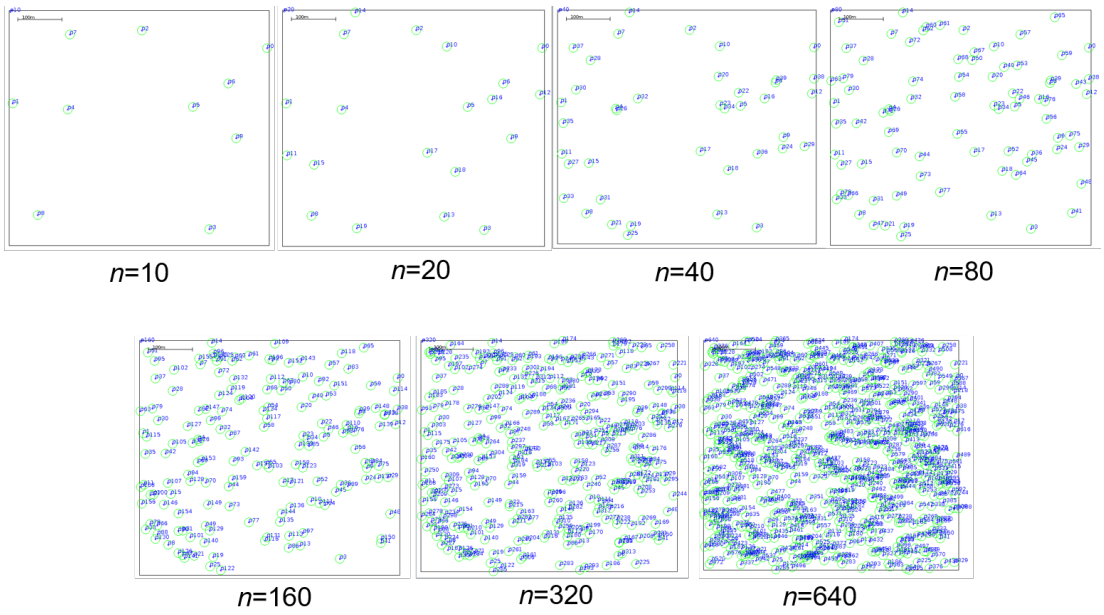


Figura 39. Representación visual de diferente número de vértices en el escenario, utilizando valores de $n = 10$, $n = 20$, $n = 40$, $n = 80$, $n = 160$, $n = 320$, y $n = 640$.

con contorno verde) y que cubran todo el escenario. Además, se permite que sólo un vértice utilice uno de los modelos de movilidad, *random walk* o *random waypoint*, para que vaya transmitiendo un mensaje.

Al finalizar la simulación se contabilizan el número de copias del mensaje en la red. En la Figura 40 se puede observar el escenario diseñado para los dos modelos de movilidad. En esta figura, cada vértice que recibió el mensaje tiene un cuadro verde en la esquina superior izquierda de su área de cobertura. Se puede observar que el modelo de movilidad *random walk* genera una movilidad más acotada que *random waypoint*.

Al analizar la implementación de estos modelos de movilidad en el simulador ONE, se identificó que *random walk* tiene acotada la máxima distancia que un vértice se puede desplazar, mientras que *random waypoint* no tiene un límite, entre más grande es el escenario mayor diferencia se da entre estos dos modelos de movilidad. Es por esto que para los experimentos finales se elige utilizar *random waypoint*.

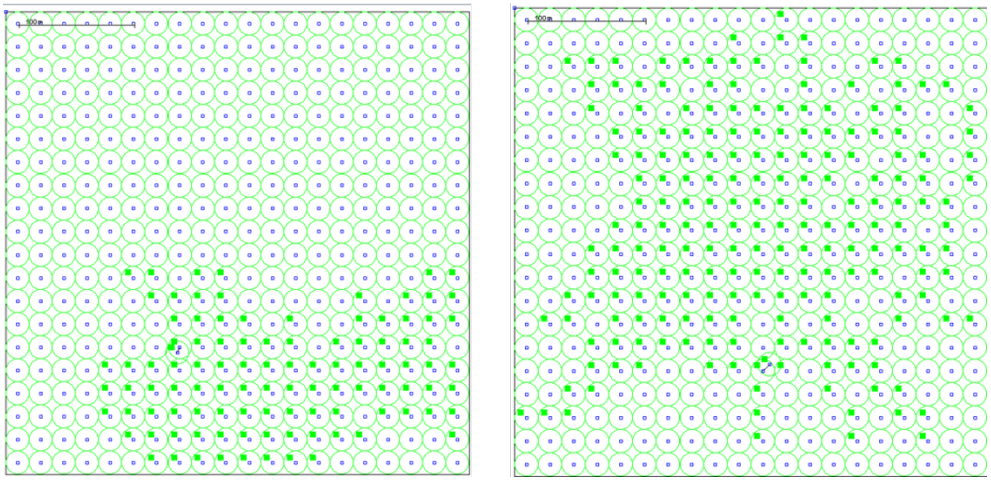


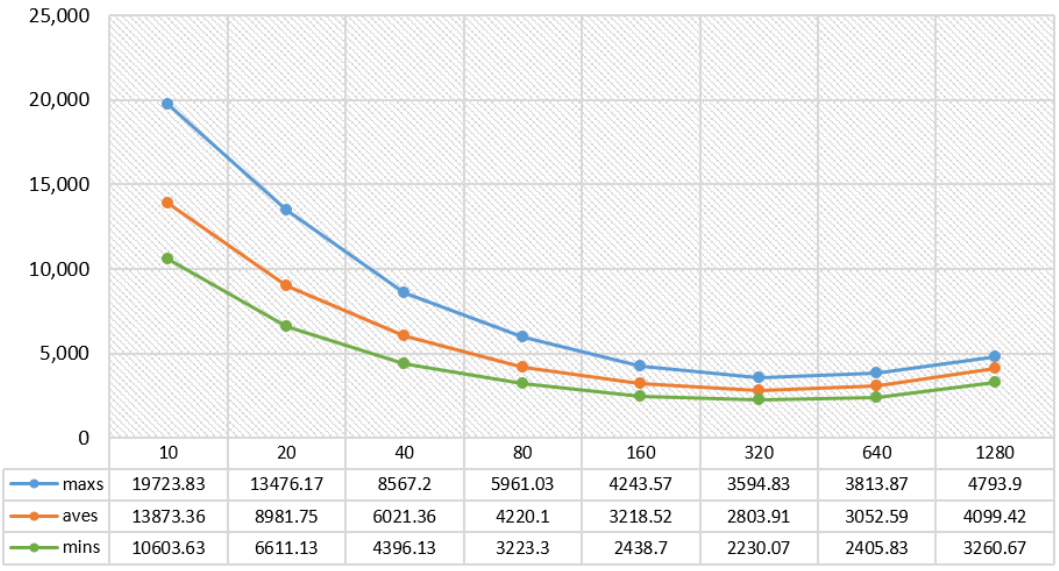
Figura 40. Representación visual de la movilidad utilizando *random walk* (parte izquierda) y *random waypoint* (parte derecha).

B.2.2. Análisis de los tiempos de saturación

En esta sección se muestran las gráficas con los tiempos de saturación de los buffers para todos los valores de n y los dos modelos de movilidad aleatoria considerados en la segunda ronda de experimentos preliminares.

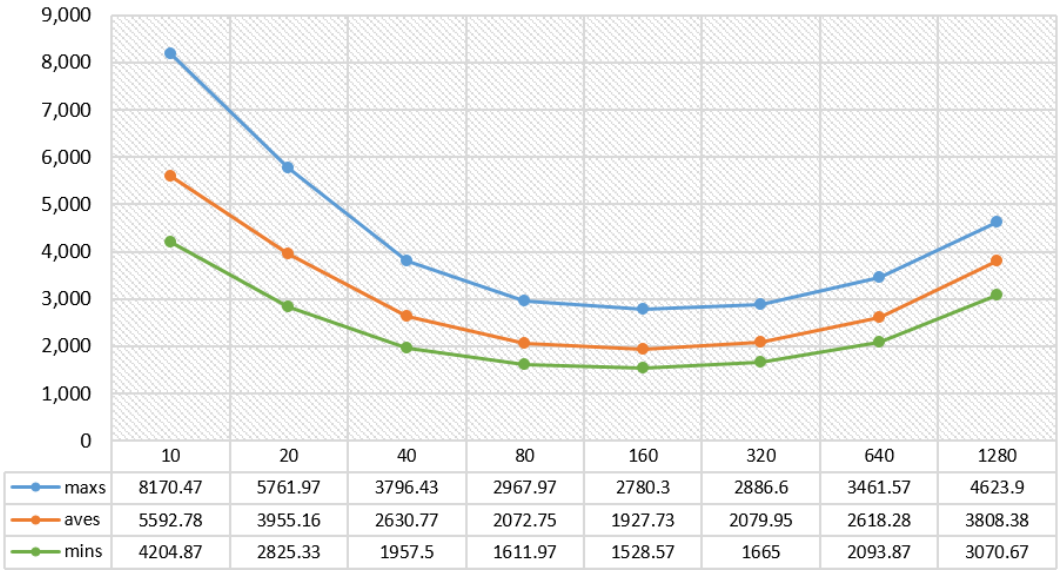
En las figuras 41 y 42 se muestra el tiempo mínimo de saturación (tiempo en el cual el primer vértice satura su memoria), el tiempo promedio de saturación, y el tiempo máximo de saturación (tiempo en el cual el último vértice satura su memoria) para *random walk* y *random waypoint*, respectivamente. En ambas figuras se puede observar que las tendencias en las curvas de ambos modelos son muy similares; sin embargo, al observar la escala del eje vertical (tiempo de simulación) es notorio que al utilizar *random waypoint* se requiere menos tiempo para saturar los buffers en la red sin importar el valor de n.

La Figura 43 muestra el tiempo máximo de saturación para *random walk* y *random waypoint*. En la figura se aprecia que los tiempos máximos de saturación al utilizar *random waypoint* son más pequeños para todo n; sin embargo, la diferencia entre los tiempos se reduce conforme el valor de n aumenta. Note que cuando $n = 160$ y $n = 320$ es cuando se obtienen los tiempos de saturación más pequeños para ambos modelos de movilidad. Además, para los casos con $n > 320$ los tiempos tienden a aumentar. En la sección B.2.3 se discute porqué se dan estas tendencias.



	10	20	40	80	160	320	640	1280
maxs	19723.83	13476.17	8567.2	5961.03	4243.57	3594.83	3813.87	4793.9
aves	13873.36	8981.75	6021.36	4220.1	3218.52	2803.91	3052.59	4099.42
mins	10603.63	6611.13	4396.13	3223.3	2438.7	2230.07	2405.83	3260.67

Figura 41. Tiempo mínimo, promedio y máximo de saturación de los buffers para todos los valores de *n* utilizando el modelo de movilidad *random walk*.



	10	20	40	80	160	320	640	1280
maxs	8170.47	5761.97	3796.43	2967.97	2780.3	2886.6	3461.57	4623.9
aves	5592.78	3955.16	2630.77	2072.75	1927.73	2079.95	2618.28	3808.38
mins	4204.87	2825.33	1957.5	1611.97	1528.57	1665	2093.87	3070.67

Figura 42. Tiempo mínimo, promedio y máximo de saturación de los buffers para todos los valores de *n* utilizando el modelo de movilidad *random waypoint*.

B.2.3. Análisis del efecto de variar el número de vértices en la red

En esta sección se muestra un análisis del impacto al utilizar diferentes valores de *n* en los resultados de la segunda ronda de experimentos preliminares.

Con base a los resultados observados, en los tiempos de saturación reportados en la sección B.2.2, se establece la siguiente hipótesis: durante una ejecución, el vecindario

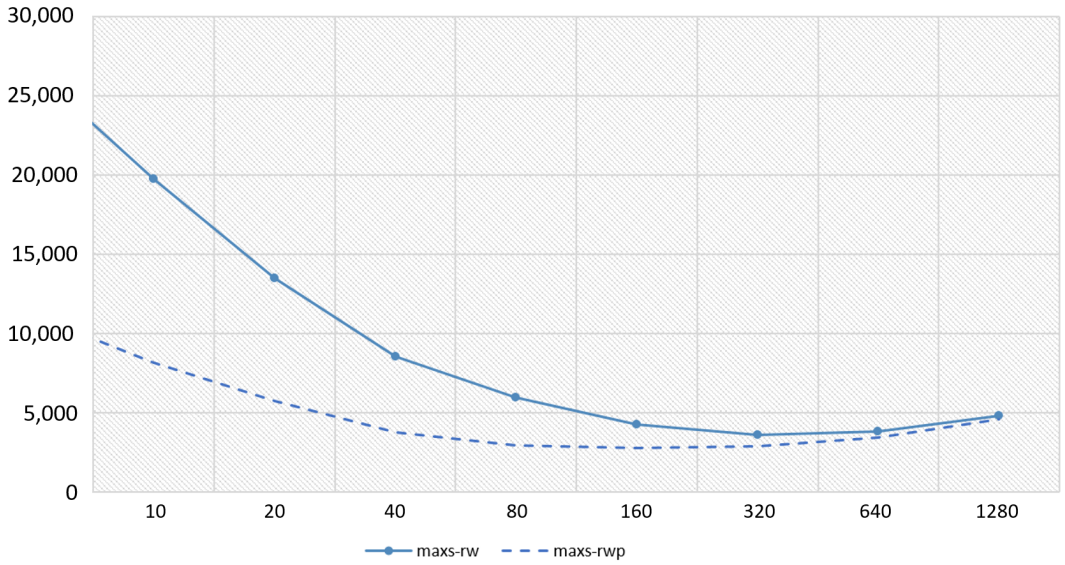


Figura 43. Tiempo máximo de saturación de los buffers para todos los valores de *n* utilizando los modelos de movilidad *random walk* y *random waypoint*.

puede ser tan grande que el vértice no alcanza a intercambiar mensajes con todos sus vecinos. Para confirmar lo anterior se contabilizó el promedio del número de contactos y el número de interacciones de todos los vértices para los escenarios con $n = 80$, $n = 160$, $n = 320$, y $n = 640$. El análisis experimental se enfoca en estos valores de n porque son los casos en los cuales se obtienen los tiempos de saturación más pequeños (ver sección B.2.2).

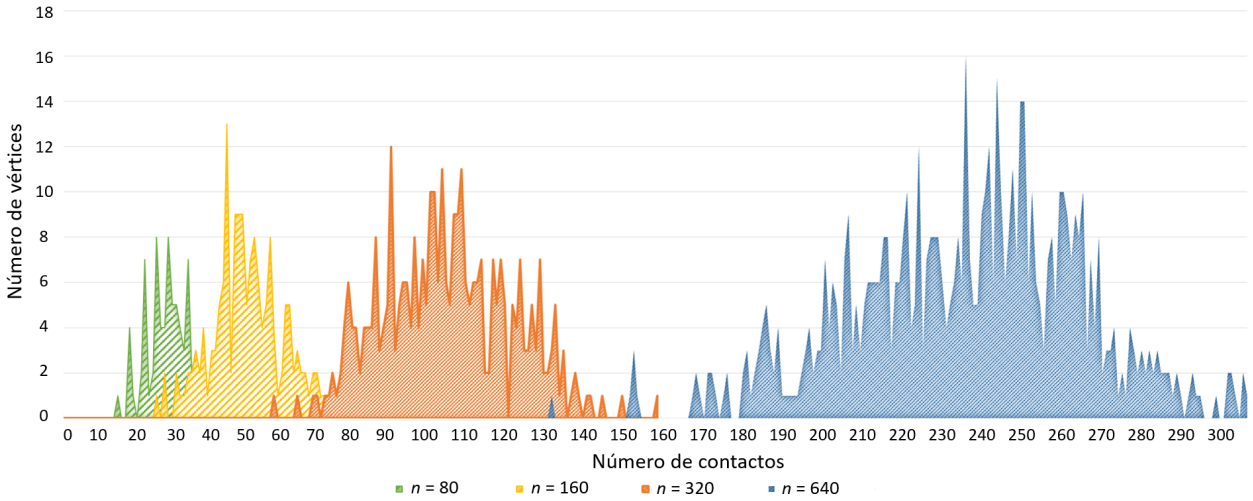


Figura 44. Frecuencia del número de contactos de todos los vértices utilizando $n = 80$, $n = 160$, $n = 320$, y $n = 640$, para la segunda ronda de experimentos preliminares.

En la Figura 44 se ilustra la diferencia en la frecuencia de contactos al utilizar estos cuatro diferentes valores de *n*. En esta figura se puede observar como la distribución

en el número de contactos es muy similar, aunque al aumentar el valor de n aumenta la dispersión y la media de los contactos.

Por otra parte, la Tabla 8 presenta el concentrado del número de contactos y el número de interacciones de todos los vértices para estos cuatro valores de n.

En la última columna de la Tabla 8 se puede observar la razón de las interacciones entre los contactos, es decir, el número de veces que un contacto se convierte en una interacción. En general, se puede observar que en la mitad de las ocasiones el vértice no alcanza a comunicarse con todos los vértices que encuentra. Es notorio que los valores de $n = 160$ y $n = 320$ son los que obtienen una tasa de interacción por contacto más alta. Además, afecta de manera similar tener un número reducido de vértices a un número muy alto de vértices en un escenario de tamaño fijo. Finalmente, se puede decir que de nada sirve tener un gran número de vecinos si el vértice solamente puede transmitir con uno a la vez. En general, el utilizar un valor de n muy pequeño o muy grande puede impactar de forma similar el desempeño de los algoritmos de enrutamiento.

Tabla 8. Número de contactos, número de interacciones, y la razón de los contactos entre las interacciones de todos los vértices utilizando $n = 80$, $n = 160$, $n = 320$, y $n = 640$, para la segunda ronda de experimentos preliminares.

Número de vértices	Número de interacciones	Número de contactos	Número de inter./contac.
80	1,096	2,290	0.47
160	4,356	7,758	0.56
320	17,087	32,186	0.53
640	70,336	148,530	0.47

B.3. Experimentos de evaluación para el algoritmo Alg-DHM

En esta sección se presentan dos rondas de experimentos donde se evalúa el desempeño del algoritmo Alg-DHM utilizando dos enfoques diferentes y comparándolo contra un algoritmo de inundación epidémico. A la primera versión del algoritmo Alg-DHM se le denomina DHM-1. En esta versión se utilizan todas las características descritas en el apéndice A. Por otra parte, a la segunda versión del algoritmo Alg-DHM se le denomina DHM-2. En esta versión se utilizan todas las características del algoritmo Alg-DHM excepto la segunda etapa de intercambio.

B.3.1. Primera ronda de experimentos

Los experimentos presentados en esta sección utilizan los parámetros de la Tabla 9. Vale la pena destacar que a partir de estos experimentos se utiliza una capacidad de almacenamiento de mensajes, κ_i, a lo más de $n/2$. Es decir, de los n mensajes distintos en la red, cada vértice puede almacenar como máximo la mitad.

Tabla 9. Parámetros para la primera ronda de los experimentos preliminares para evaluar el algoritmo Alg-DHM.

Parámetro	Valor
Número total de vértices (n)	$8, 16, 32$
Modelo de movilidad	Random waypoint, Random walk
Radio de transmisión	10 m
Velocidad de transmisión	250 kBps
Velocidad de movimiento	0.5 - 1.5 m/s
Semillas aleatorias	100
Tamaño de mensaje	1 kB
Tamaño del buffer (κ_i)	$n/2$ kB
Creación de mensajes	Un mensaje por vértice en $t = 0$
Tiempo de simulación	200,000 segundos
Esquema de energía	Sin restricciones
Tamaño del escenario	590 m × 532 m

B.3.1.1. Métricas de desempeño

Para estos experimentos se presentan un conjunto de métricas para poder comparar el desempeño de los algoritmos. La finalidad de estas métricas son principalmente evaluar qué tan homogéneos están distribuidos los mensajes y qué tan llenos están los buffers de los vértices en la red.

La métrica denominada ‘Factor de uniformidad’ (FU) se define con la ecuación 5.

$$FU = \prod_{i=1}^{n} \frac{c_i}{\kappa_i} \tag{5}$$

donde c_x es la frecuencia absoluta del número de copias del mensaje m_x en la red, $1 \leq c_x \leq n$. Esta métrica sirve para medir qué tan bien distribuidos están los mensajes en la red, a la cual se le denomina factor de uniformidad (FU). La métrica FU se evalúa de forma global y suponiendo que se conocen el número de nodos, el tamaño y contenido de los buffers.

La métrica ‘Porcentaje de utilización del buffer’ (PUB) se define con la ecuación 6.

$$PUB = \sum_{i=1}^{n} \frac{c_i}{\kappa_i} \quad (6)$$

El PUB refleja el progreso de llenado del buffer. Una vez que los buffers se saturan, el PUB no sirve para discriminar la uniformidad. Por el contrario, FU cambia su valor de forma abrupta y sin relación a la uniformidad del escenario conforme se llenan los buffers.

Para los casos donde $\kappa \geq n$. FU resulta de mayor utilidad que PUB. Por lo que para obtener una mejor interpretación de los resultados se deben utilizar FU y PUB o una combinación de ambos.

La métrica 'Factor de uniformidad relativo' (FUR) se define con la ecuación 7.

$$FUR = \prod_{i=1}^{n} \frac{c_i}{max(c_i)} \quad (7)$$

El FUR supone que la capacidad relativa es el máximo número de mensajes dentro de un vértice. Además, permite evaluar eficazmente la uniformidad del escenario, ya que cuando existe el mismo número de mensajes en la red es cuando alcanza su máximo valor.

Se requiere medir la uniformidad y la utilización del buffer, por lo cual se propone combinar las dos métricas por medio de una suma ponderada. A esta métrica se le denomina 'Promedio del factor de uniformidad relativo y del porcentaje de utilización del buffer' (Ave(FUR,PUB)), y se define con la ecuación 8.

$$Ave(FUR, PUB) = \frac{FUR + PUB}{2} \quad (8)$$

B.3.1.2. Resultados de la primera ronda de experimentos

En las figuras 45, 46, 47, 48, 49 y 50 se muestran los valores que van tomando las métricas FU, FUR, PUB y Ave(FUR, PUB) para los diferentes algoritmos durante una ejecución.

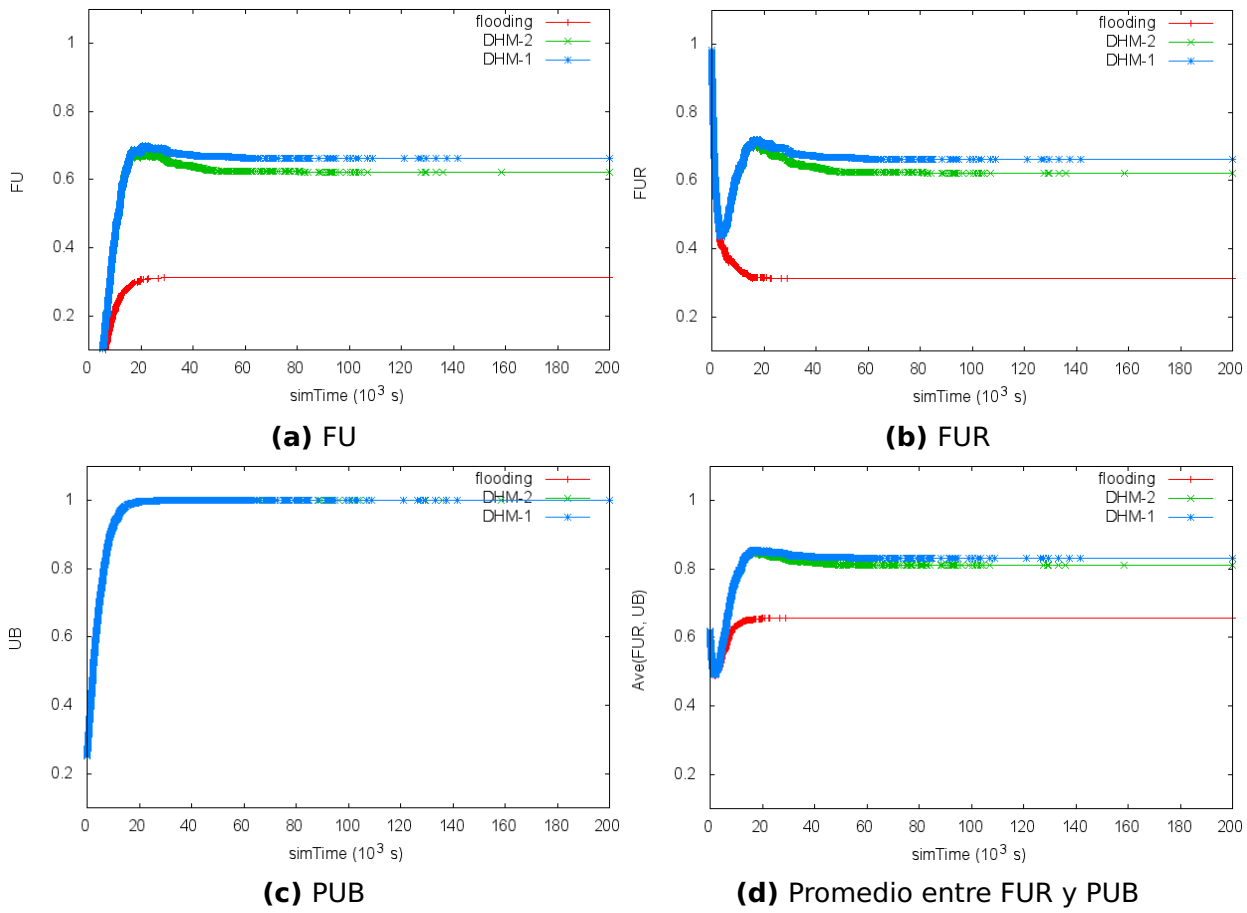
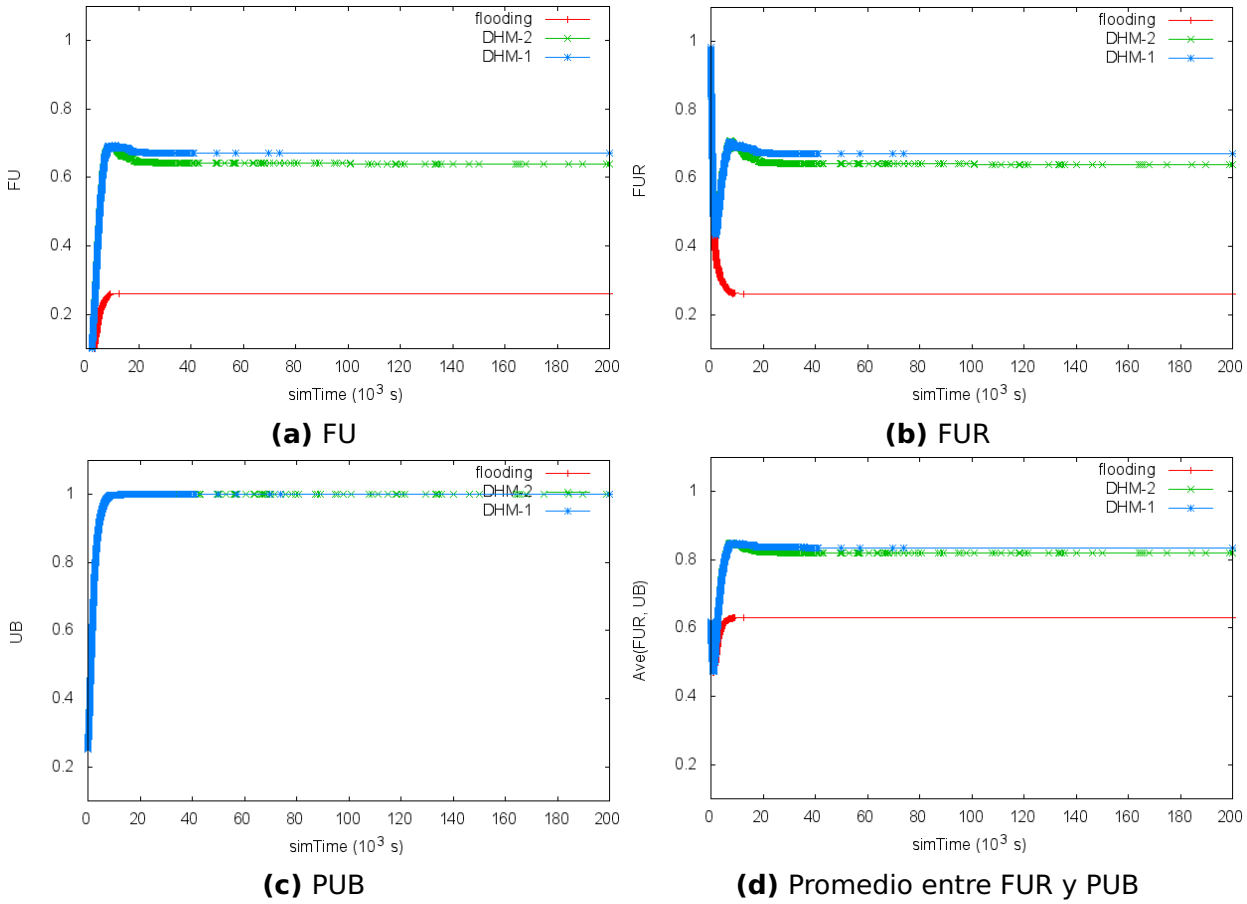


Figura 45. Resultados de las métricas FU, FUR, PUB, y el promedio de FUR y PUB durante una ejecución con $n = 8$ y utilizando el modelo de movilidad *random walk*.

Particularmente, en las figuras 45 y 46 se muestran los valores de las métricas a través del tiempo para $n = 8$ utilizando *random walk* y *random waypoint*, respectivamente. En esta figura se puede observar que las variantes del algoritmo DHM siempre obtienen valores más altos que flooding, excepto en PUB donde no hay diferencia entre ellos ya que llenan al mismo ritmo los buffers de los vértices.

Similarmente, en las figuras 47 y 48 se muestran los valores de las métricas para $n = 16$ utilizando *random walk* y *random waypoint*, respectivamente. En estas figuras se observa que, en general, se mantiene el mismo comportamiento que con $n = 8$ pero decrecen ligeramente los valores de todas las métricas, excepto para PUB.

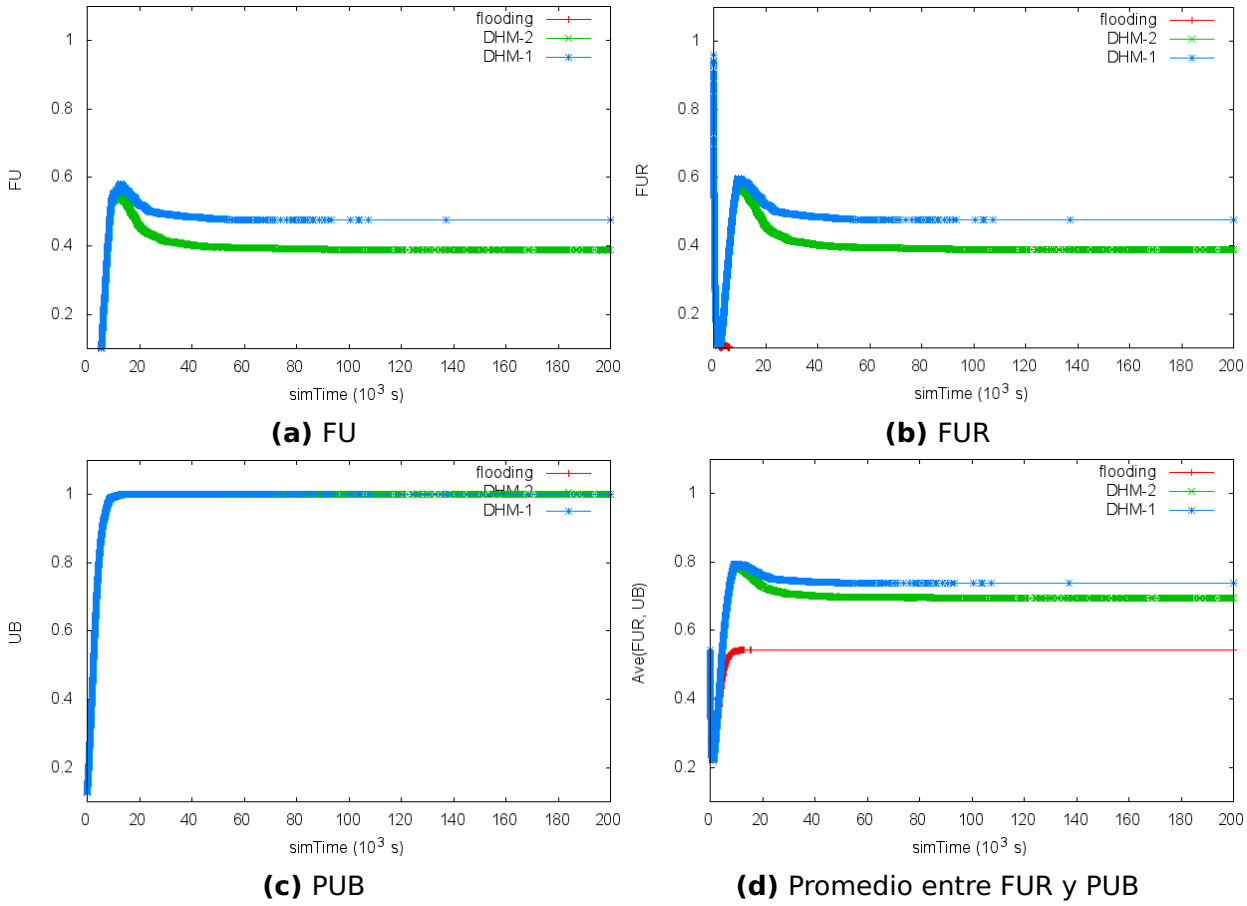
Por último, en las figuras 49 y 50 muestran los valores de las métricas para $n = 32$, utilizando *random walk* y *random waypoint*, respectivamente. En general se mantienen las tendencias que se obtienen con $n = 8$ y $n = 16$, pero decrementando ligeramente el ritmo y valor máximo de crecimiento. Esto último es más notorio para las



(a) FU **(b)** FUR

(c) PUB **(d)** Promedio entre FUR y PUB

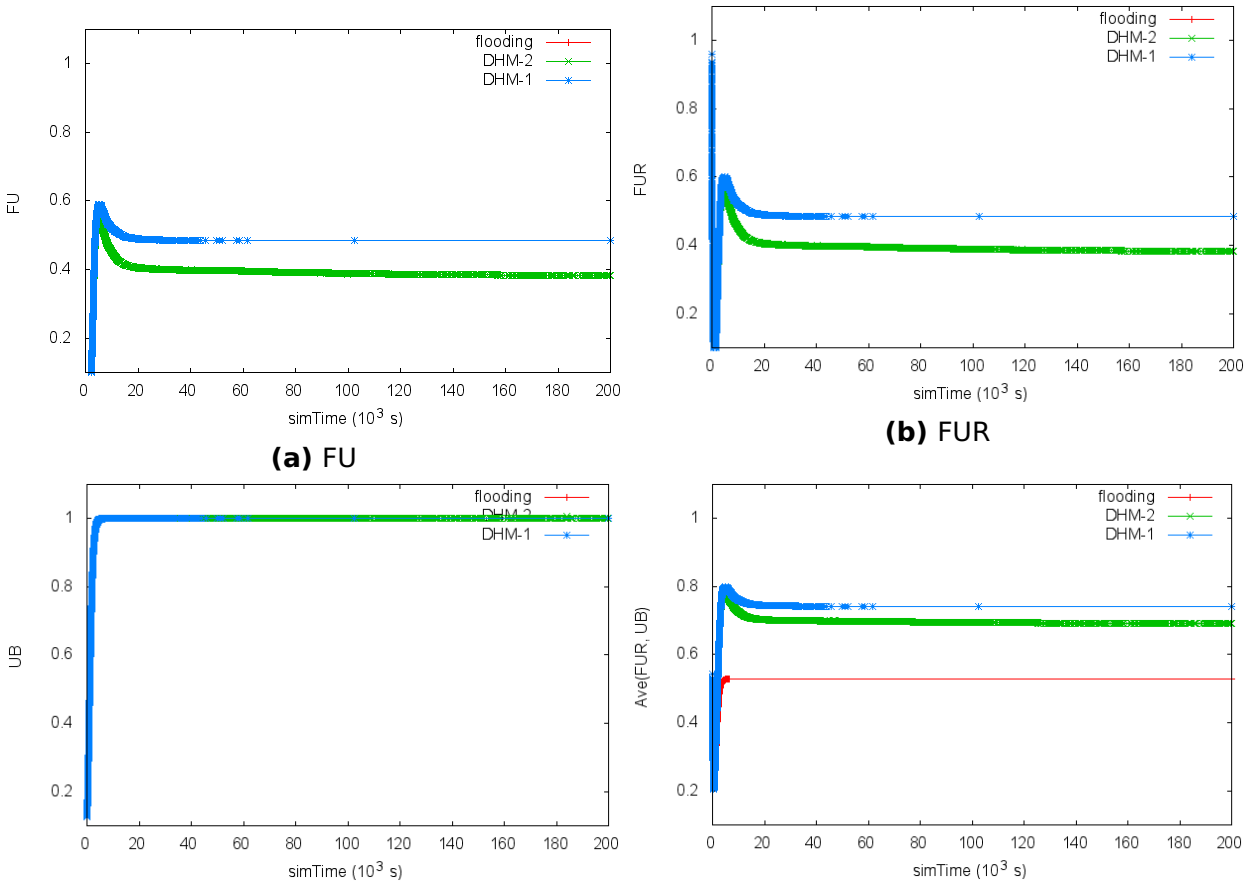
Figura 46. Resultados de las métricas FU, FUR, PUB, y el promedio de FUR y PUB durante una ejecución con $n = 8$ y utilizando el modelo de movilidad *random waypoint*.



(a) FU **(b)** FUR

(c) PUB **(d)** Promedio entre FUR y PUB

Figura 47. Resultados de las métricas FU, FUR, PUB, y el promedio de FUR y PUB durante una ejecución con $n = 16$ y utilizando el modelo de movilidad *random walk*.



(a) FU **(b)** FUR

(c) PUB **(d)** Promedio entre FUR y PUB

Figura 48. Resultados de las métricas FU, FUR, PUB, y el promedio de FUR y PUB durante una ejecución con $n = 16$ y utilizando el modelo de movilidad *random waypoint*.

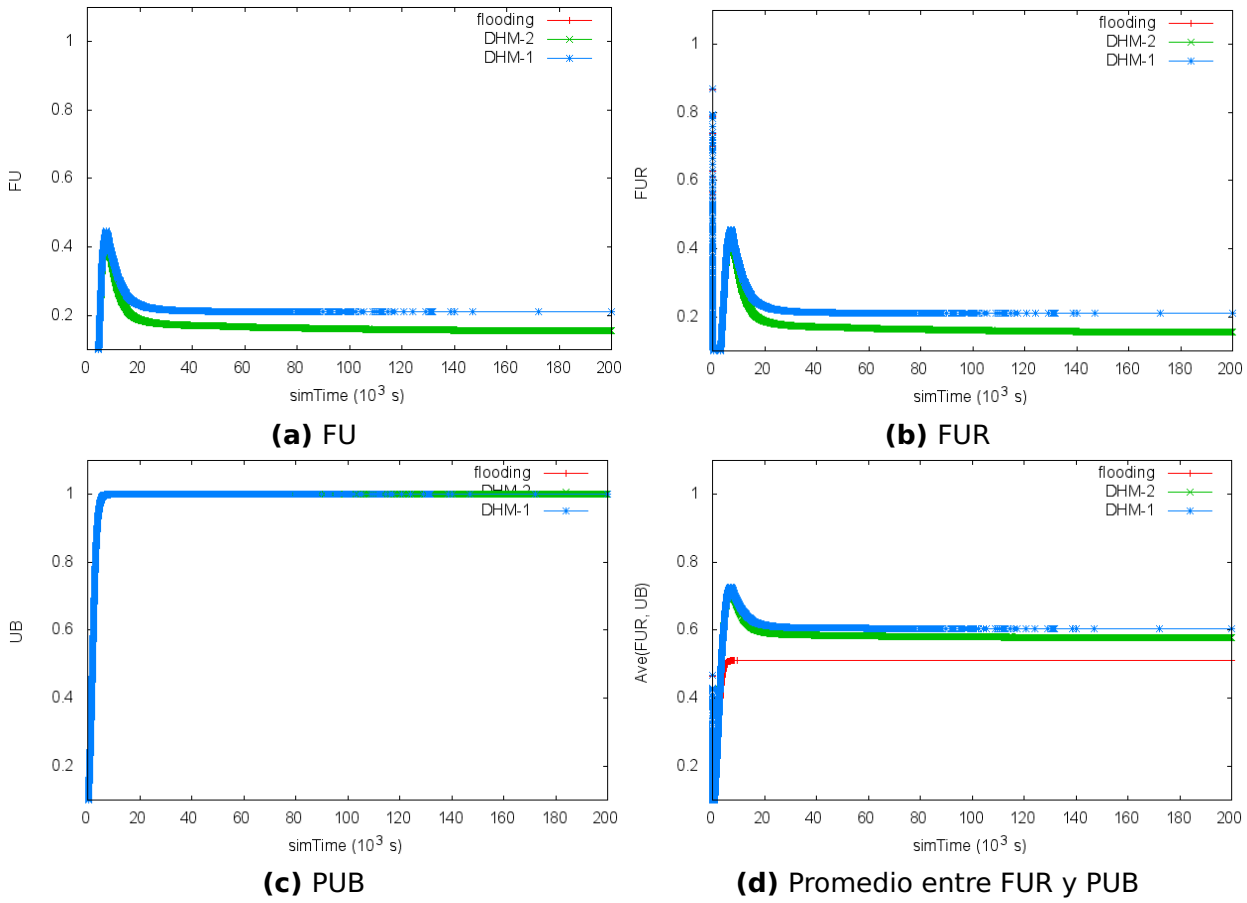


Figura 49. Resultados de las métricas FU, FUR, PUB, y el promedio de FUR y PUB durante una ejecución con $n = 32$ y utilizando el modelo de movilidad *random walk*.

métricas FU y FUR.

En general se puede decir que el algoritmo DHM es mejor que flooding para todos los casos. Todos los algoritmos empeoran su desempeño al aumentar n. El modelo de movilidad *random waypoint* permite distribuir mejor los mensajes cuando se utiliza el algoritmo DHM. Por el contrario, el modelo de movilidad *random walk* obtiene ligeramente mejores resultados al usar flooding. El algoritmo DHM-1 es mejor en la mayoría de los casos al algoritmo DHM-2. Se conjetura que el algoritmo DHM-2 elimina mensajes en una etapa temprana de la simulación lo cual afecta la distribución de mensajes posterior.

Las gráficas presentadas en esta sección son para una sola ejecución. Aunque éstas son representativas, para poder interpretar mejor los datos se podría generar una gráfica que represente múltiples ejecuciones y diferentes valores de n.

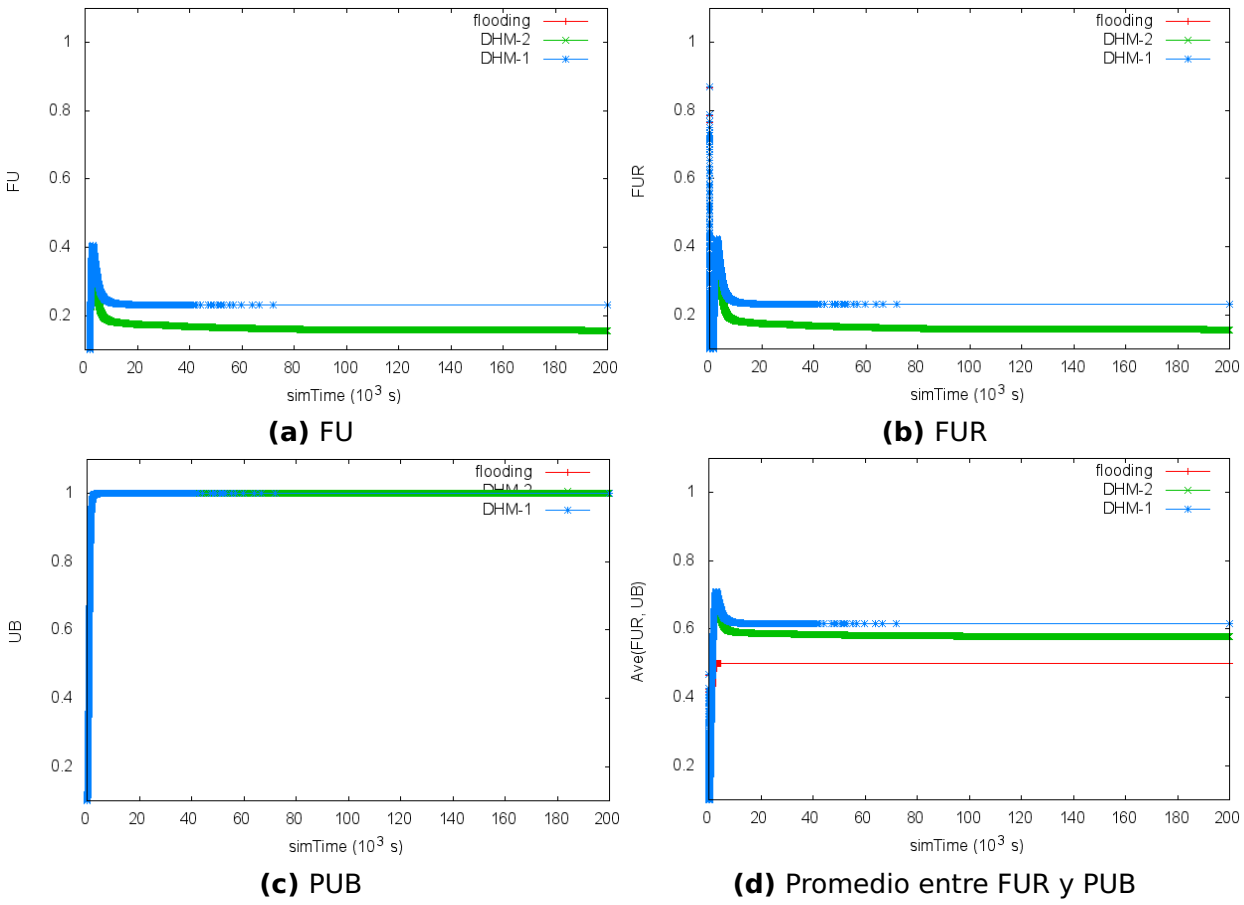


Figura 50. Resultados de las métricas FU, FUR, PUB, y el promedio de FUR y PUB durante una ejecución con $n = 32$ y utilizando el modelo de movilidad *random waypoint*.

B.3.2. Segunda ronda de experimentos

En estos experimentos se utilizan los parámetros que se muestran en la Tabla 10. La principal diferencia con los parámetros utilizados en la ronda de experimentos anterior, es que ahora se consideran cuatro diferentes capacidades de almacenamiento para el buffer de los vértices ya que este parámetro es primordial en el problema DMHM. Además, se considera otro valor de n más grande.

En la ronda de los experimentos anteriores se destaca que es necesario combinar las métricas FUR y PUB para obtener una métrica que refleje a ambas. En esa ronda se utilizó el promedio de estas métricas; sin embargo, es más apropiado utilizar el producto (FUR×PUB). En las siguientes figuras se utiliza esta métrica en gráficas que concentran múltiples ejecuciones con parámetros distintos.

Tabla 10. Parámetros para la segunda ronda de los experimentos preliminares para evaluar el algoritmo Alg-DHM.

Parámetro	Valor
Número total de vértices (n)	$8, 16, 32, 64$
Modelo de movilidad	Random waypoint, Random walk
Radio de transmisión	10 m
Velocidad de transmisión	250 kBps
Velocidad de movimiento	0.5 - 1.5 m/s
Semillas aleatorias	100
Tamaño de mensaje	1 kB
Tamaño del buffer (κ_i)	$n/2, n/4, n/8, n/16$ kB
Creación de mensajes	Un mensaje por vértice en $t = 0$
Tiempo de simulación	100,000 segundos
Esquema de energía	Sin restricciones
Tamaño del escenario	590 m × 532 m

B.3.2.1. Resultados de la segunda ronda de experimentos

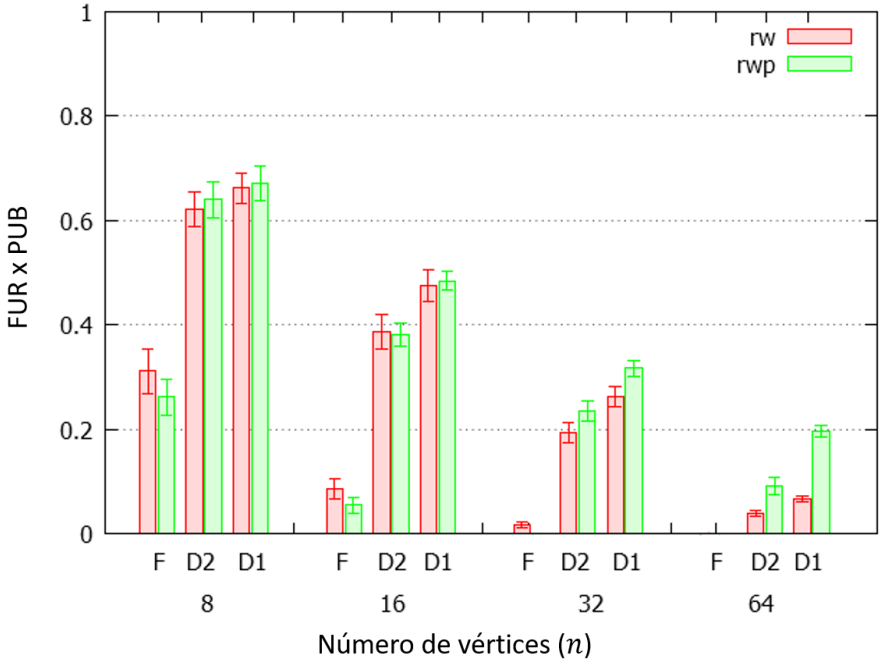
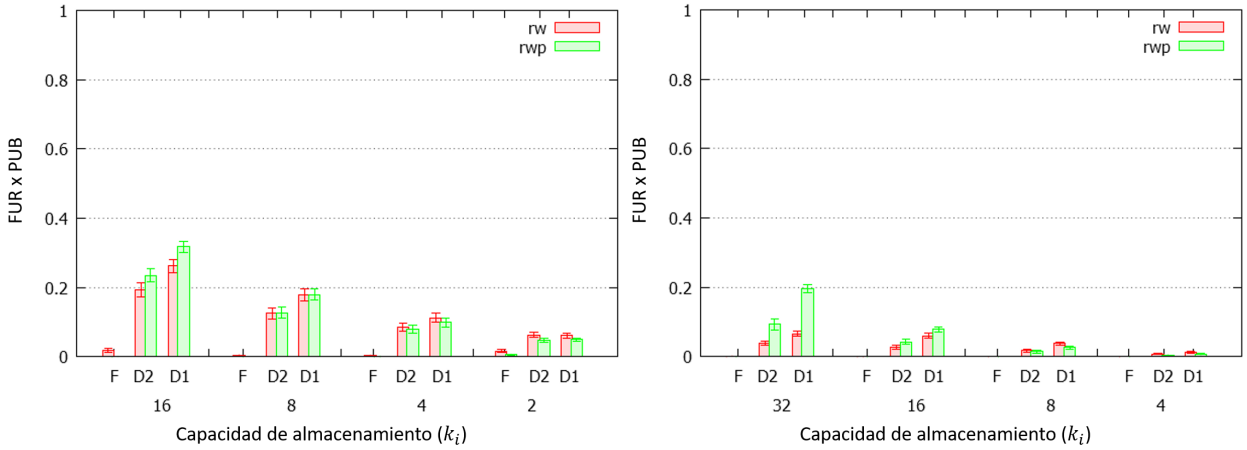


Figura 51. Resultados de la métrica FUR×PUB con $n = 8$, $n = 16$, $n = 32$, y $n = 64$, un tamaño de buffer de $n/2$, utilizando los modelos de movilidad *random walk* y *random waypoint* con los algoritmos de Flooding, DHM-1, y DHM-2.

La Figura 51 muestra los resultados de la métrica FUR×PUB con cuatro diferentes valores de n y una capacidad de almacenamiento de $n/2$ mensajes. Note que entre menor sea el valor de n, mejor es el desempeño de todos los algoritmos. En general, flooding obtiene los peores resultados para todo n. La variante DHM-1 obtiene mejores resultados que DHM-2, incluso la diferencia entre ellos es mayor al aumentar el valor de n. En la Figura 51 se varía el valor de n y se fija el tamaño de buffer. En las siguientes

figuras se fijan los valores de $n = 64$ y $n = 32$ y se varía el tamaño de buffer.



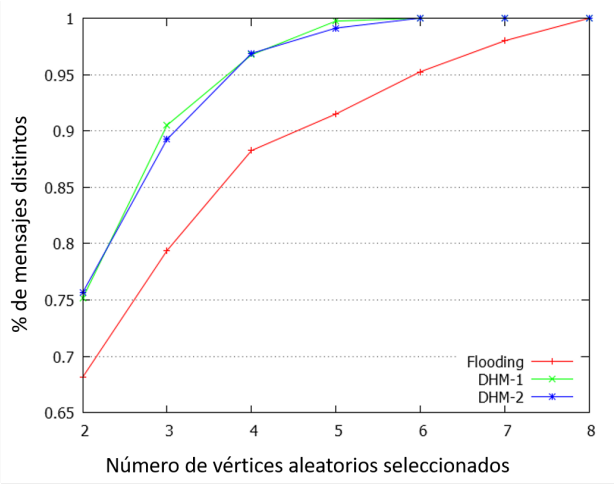
(a) FUR×PUB con $n = 32$. **(b)** FUR×PUB con $n = 64$.

Figura 52. Resultados de la métrica FUR×PUB con n = 64 y n = 32, considerando tamaño de buffer de $n/2$, $n/4$, $n/8$, y $n/16$, utilizando los modelos de movilidad *random walk* y *random waypoint* con los algoritmos de Flooding, DHM-1, y DHM-2.

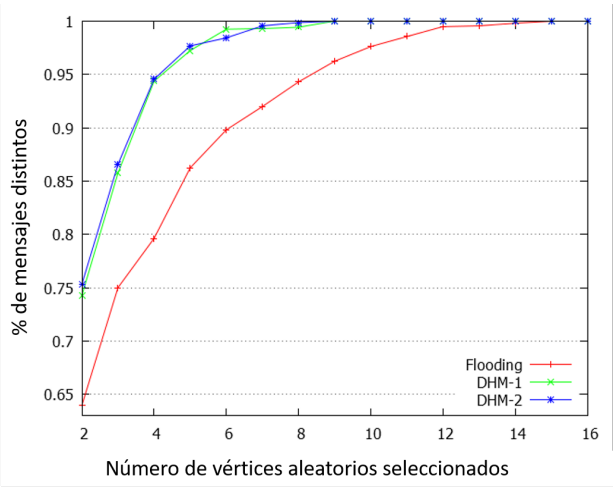
En la Figura 52 se muestran los resultados del producto de las métricas FUR y PUB para $n = 32$ y $n = 64$ con las cuatro variantes de tamaño de buffer. Como se puede observar en esta figura, las variantes del algoritmo DHM genera mejores resultados que Flooding para todos los casos. Similarmente a las gráficas anteriores, en estás gráficas la variante DHM-1 es mejor o igual a la variante DHM-2. En general, sin importar el algoritmo, el modelo de movilidad que permite mejor desempeño de los algoritmos es *random waypoint*. Otro aspecto a destacar es que a mayor capacidad de almacenamiento, mejor desempeño en las métricas. Sin embargo, es notorio que al obtener valores siempre menores a 0.4, el desempeño de todos los algoritmos para estos valores de n no es bueno.

En esta segunda ronda de experimentos se utiliza una métrica que refleja la utilidad de distribuir el número máximo y homogéneo de copias. La métrica denominada 'Porcentaje de mensajes distintos al seleccionar vértices aleatorios' (PMD) se calcula al extraer uno a uno todos los vértices en la red e ir contando el número de mensajes distintos que se obtienen después de cada extracción. Esta métrica es la que originó la métrica PNV que se define en la sección 5.1.3.

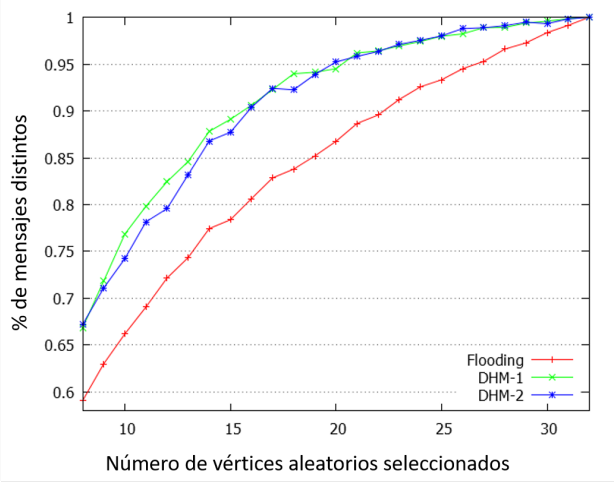
La Figura 53 muestra la métrica PMD con $n = 8$, $n = 16$, $n = 32$, y $n = 64$ utilizando *random waypoint* y cuatro diferentes tamaños de buffer. En general, en la figura se observa que, para todo n, el algoritmo Alg-DHM requiere extraer un número menor de



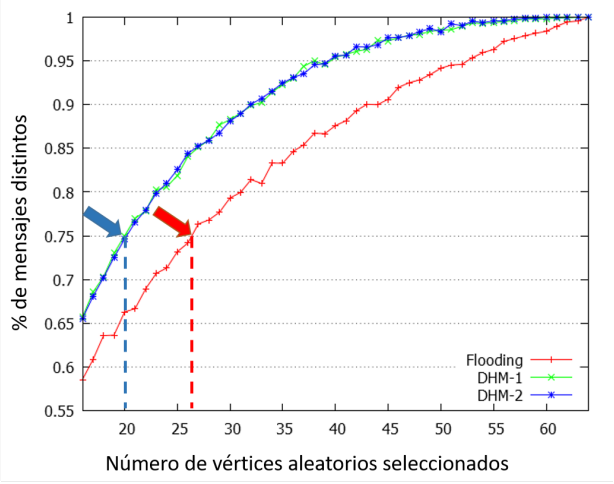
(a) PMD con $n = 8$ y $\kappa_i = n/2$.



(b) PMD con $n = 16$ y $\kappa_i = n/2$.



(c) PMD con $n = 32$ y $\kappa_i = n/8$.



(d) PMD con $n = 64$ y $\kappa_i = n/16$.

Figura 53. Resultados de la métrica PMD con $n = 8, 16, 32, 64$, utilizando *random waypoint* y cuatro distintos tamaños de buffer.

vértices para obtener el mismo porcentaje de mensajes que el algoritmo flooding. Por ejemplo, en la Figura 53d se muestra (resaltado con una flecha azul) que las variantes del algoritmo Alg-DHM requieren extraer 20 vértices en promedio para obtener el 75 % de mensajes distintos en la red. Mientras que el algoritmo flooding requiere extraer 26 vértices en promedio para obtener el mismo porcentaje (resaltado con una flecha roja).

Los resultados preliminares presentados en esta sección muestran que todos los algoritmos empeoran su desempeño al aumentar el número de vértices. Los algoritmos DHM distribuyen mejor los mensajes cuando se utiliza el modelo de movilidad *random waypoint*. El algoritmo DHM-1 es mejor en la mayoría de los casos al algoritmo DHM-2. Se conjetura que el algoritmo DHM-2 elimina mensajes en una etapa temprana de la simulación, lo cual afecta la distribución posterior de mensajes.

Aunque, en general, el algoritmo Alg-DHM es el que obtiene el mejor desempeño en estos experimentos preliminares, las métricas ilustran que sus resultados distan de aproximarse al máximo valor en las métricas. Por lo que es posible mejorar su desempeño. Se puede conjeturar que si se encuentra una forma de calcular un número máximo de copias que deben distribuirse en la red y se consigue generar un identificador único para cada mensaje se podría controlar el número de copias, mejorando considerablemente el desempeño del algoritmo Alg-DHM.

Anexo C. Procedimiento del análisis estadístico

En este anexo se describe la metodología que se sigue para los análisis estadísticos de los resultados reportados en el Capítulo 6. En general, el análisis estadístico utilizado en esta tesis comprueba de manera formal que exista una diferencia significativa entre dos muestras (conjuntos de datos) y, si existe alguna diferencia, se determina qué muestra es mayor. Para poder determinar si existe una diferencia significativa o no entre un par de muestras es necesario realizar una serie subsecuente de pruebas estadísticas, las cuales se explican en este apéndice.

Para cada prueba estadística, dependiendo lo que se quiera probar, se plantea una hipótesis nula y una hipótesis alternativa. Como resultado de aplicar una prueba se obtiene un valor p (p-*value*), el cual es la probabilidad de obtener un valor al menos tan extremo como el que se ha obtenido, suponiendo que la hipótesis nula es cierta. La hipótesis nula se rechaza si el valor p asociado al resultado observado es igual o menor que el nivel de significancia establecido, regularmente 0.05. Por lo tanto, solo se pueden obtener dos resultados, que se rechace la hipótesis nula o que la hipótesis nula no se pueda rechazar con ese nivel de significancia (que no implica que la hipótesis nula sea verdadera). A continuación se explican todas las pruebas realizadas para el análisis.

C.1. Pruebas de normalidad

En esta sección se explica la prueba de normalidad utilizada para el análisis estadistico. Existen muchos tipos de pruebas para comprobar si un conjunto de datos se comporta como una distribución normal. Kolmogorov-Smirnov es una de las más utilizadas. La prueba Kolmogorov-Smirnov es una prueba no paramétrica que se utiliza para determinar la bondad de ajuste de dos distribuciones de probabilidad entre sí. En el caso que se requiera verificar la normalidad de una distribución, la prueba Lilliefors (Lilliefors, 1967) conlleva algunas mejoras con respecto a la de Kolmogorov-Smirnov. A continuación se explican ambas pruebas.

En estadística, la prueba de Kolmogorov-Smirnov es una prueba no paramétrica de la igualdad de las distribuciones continuas; se puede utilizar para comparar una muestra con una distribución de probabilidad de referencia o para comparar dos muestras.

El estadístico de Kolmogorov-Smirnov cuantifica la distancia entre la función de distribución empírica de la muestra y la función de distribución acumulada de la distribución de referencia, o entre las funciones de distribución empírica de dos muestras.

La prueba de Kolmogorov-Smirnov se pueden modificar para servir como una prueba de bondad de ajuste, es decir, el grado de ajuste que existe entre la distribución obtenida a partir de la muestra y la distribución teórica que se supone debe seguir esa muestra. En el caso especial de la prueba de normalidad de la distribución, las muestras están estandarizadas y se compara con una distribución normal estándar. Esto equivale a establecer la media y la varianza de la distribución de referencia igual a las estimaciones de la muestra. Esta prueba se plantea como hipótesis nula de que no hay diferencia significativa entre la distribución muestral y la teórica.

En estadística, la prueba Lilliefors, llamada así por Hubert Lilliefors, profesor de estadística de la universidad de Washington, es una adaptación de la prueba Kolmogorov-Smirnov. Se utiliza para probar la hipótesis nula de que la muestra de datos proviene de una población distribuida normalmente. Esta última prueba es la que se utiliza en esta investigación.

Para esta prueba se tienen dos hipótesis:

- **H0:** (nula) Los datos siguen una distribución normal.
- **H1:** (alternativa) Los datos no siguen una distribución normal.

C.2. Prueba para determinar igualdad de varianzas

Se denomina prueba-F (*F-test*) o prueba de Fisher en honor a Ronald A. Fisher (Snedecor y Cochran, 1989). En estadística aplicada, la prueba-F se utiliza para algunas hipótesis como, la hipótesis de que las medias de múltiples poblaciones normalmente distribuidas y con la misma desviación estándar son iguales o la hipótesis de que las desviaciones estándar de dos poblaciones normalmente distribuidas son iguales.

Un requisito para aplicar esta prueba es que las dos muestras sigan una distribución normal, por lo que previamente a cada muestra se le aplica la prueba de normalidad descrita en la Sección C.1. En esta investigación la prueba-F se utiliza para comprobar

la hipótesis de que las desviaciones estándar de dos muestras son iguales. Las dos hipótesis de esta prueba son:

- **H0:** (nula) Las desviaciones estándar de las dos poblaciones normalmente distribuidas son iguales.
- **H1:** (alternativa) Las desviaciones estándar de las dos poblaciones normalmente distribuidas no son iguales.

C.3. Prueba t-Student

La prueba *t-Student* la introdujo William Sealy Gosset ('Student' era su seudónimo) (Box, 1987). Una prueba *t* es cualquier prueba de hipótesis estadística en la que la estadística de prueba sigue una distribución *t* de Student si la hipótesis nula es compatible. Existen muchas variantes de la prueba *t-Student*.

Esta prueba se utiliza en una muestra para determinar si la media de la muestra coincide con una población distribuida normalmente. Alternativamente, para dos muestras se utiliza para determinar si la media de ambas muestras son iguales. Estas pruebas se suelen llamar *t-Student*, aunque en sentido estricto este nombre sólo debe usarse si las varianzas de las dos muestras se supone que son iguales. La prueba utilizada con el supuesto de varianzas distintas se llama *t-Welch*. Estas pruebas se refieren a menudo como 'sin pareja' o 'muestras independientes', ya que generalmente se aplican cuando los datos de las dos muestras que se comparan no se traslapan. Otra variante, es cuando se supone que la diferencia entre las medias de las dos muestras es cero. Por ejemplo, suponga que se mide el tamaño del tumor de un paciente con cáncer antes y después de un tratamiento. Si el tratamiento es eficaz, se espera que el tamaño del tumor para muchos de los pacientes sean más pequeños después del tratamiento. Esto se refiere a menudo como la 'pareja' o 'medidas repetidas', para este caso se utiliza la *t-test* para datos apareados.

Un requisito para aplicar la *t-Student* es que las muestras sean normales. La prueba de *t-Student* se puede realizar para muestras independientes o dependientes (apareadas); para muestras con varianzas similiares o varianzas distintas y muestras de tamaños iguales o diferentes. Las dos hipótesis de esta prueba son:

- **H0:** (nula) Los datos normalmente distribuidos no muestran diferencia significativa, son iguales.
- **H1:** (alternativa) Los datos normalmente distribuidos muestran diferencia significativa, son diferentes.

C.4. Prueba de los signos de Wilcoxon

La prueba de los signos de Wilcoxon (*Wilcoxon Signed-rank test*) es una prueba no paramétrica para comparar la mediana de dos muestras relacionadas y determinar si existen diferencia entre ellas. Se utiliza como alternativa a la prueba *t-Student* cuando no se puede suponer la normalidad de los datos (Wilcoxon, 1945). Para esta prueba no se presupone ningún tipo de distribución particular.

Para esta prueba se tienen dos hipótesis:

- **H0:** (nula) Los datos no muestran diferencia significativa, son iguales.
- **H1:** (alternativa) Los datos muestran diferencia significativa, son diferentes.

C.5. Algoritmo de análisis estadístico

Para realizar el análisis de manera automática se utilizó el lenguaje de programación R (R Core Team, 2016). Dentro de este lenguaje se manipulan los reportes generados por el simulador ONE en archivos de texto, se aplican múltiples pruebas estadísticas a estos reportes utilizando el Algoritmo 9 y se escriben los resultados en archivos excel; todo esto por medio de las librerías rJava (Urbanek, 2016), XLConnect-Jars (Mirai Solutions GmbH, 2016b), XLConnect (Mirai Solutions GmbH, 2016a), nortest (Gross y Ligges, 2015), class (Venables y Ripley, 2002), e1071 (Meyer *et al.*, 2015) y BSDA (Arnholt, 2012). Todos los análisis en esta investigación se realizan con muestras de 100 datos, i.e., cada simulación se repitió 100 veces con los mismos parámetros iniciales.

En el Algoritmo 9 se utilizan las pruebas descritas anteriormente en forma de métodos, donde se regresa solamente el valor p; si el valor p es mayor que 0.05, no se rechaza la hipótesis nula de la prueba correspondiente. En la Línea 1 se utiliza el

análisis Lilliefors para evaluar la normalidad de las muestras. En la Línea 2 se utiliza la prueba-F para evaluar la similitud entre las varianzas. En las Líneas 3 y 6 se utiliza la prueba t-Student de 2 colas (determina si existe diferencia significativa entre las muestras) y de 1 cola (determina cuál es la muestra mayor) para varianzas similares, respectivamente. Las mismas pruebas pero para varianzas distintas se aplica en las Líneas 13 y 16. Finalmente, si las muestras no son normales se aplica la prueba de Wilcoxon en las Líneas 24 y 27, de 2 y 1 cola, respectivamente.

Tabla 11. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *A*.

Métrica	Vértices	E vs I	E vs F	E vs SnW	I vs F	I vs SnW	F vs SnW
PUB	16	I	F	E	=	I	F
	32	I	F	E	=	I	F
	64	I	F	E	=	I	F
	128	I	F	E	=	I	F
FUR	16	=	E	=	I	=	SnW
	32	I	E	SnW	I	=	SnW
	64	I	E	SnW	I	=	SnW
	128	I	E	SnW	I	=	SnW
PROD	16	I	E	E	I	I	SnW
	32	I	E	E	I	I	SnW
	64	I	E	E	I	I	SnW
	128	I	E	E	I	I	SnW
PNV	16	=	F	SnW	F	SnW	F
	32	=	F	SnW	F	SnW	F
	64	=	F	SnW	F	SnW	F
	128	=	F	SnW	F	SnW	SnW
TC	16	E	F	E	F	SnW	F
	32	E	F	E	F	SnW	F
	64	E	F	E	F	SnW	F
	128	E	F	E	F	SnW	F

Pseudocódigo 9 two-samples-statistic-analysis

Entrada: Dos muestras en los vectores números x, y.
Salida: Una cadena indicando la muestra mayor 'x', 'y' o '=' si son iguales.

```
1: if lillieFors(x) > 0.05 and lillieFors(y) > 0.05 then
   ▹ Las muestras pertenecen a una distribución normal
2:     if f_Test(x,y) > 0.05 then
       ▹ Las varianzas son similares
3:         if tStudent_2Colas_varianzasIguales(x,y) > 0.05 then
           ▹ Los datos no muestran diferencia significativa
4:             return " = "
5:         else
           ▹ Los datos muestran diferencia significativa
6:             if tStudent_1Cola_varianzasIguales(x,y) > 0.05 then
7:                 return "x"
8:             else
9:                 return "y"
10:            end if
11:        end if
12:    else
       ▹ Las varianzas son distintas
13:        if tStudent_2Colas_varianzasDiferentes(x,y) > 0.05 then
           ▹ Los datos no muestran diferencia significativa
14:            return " = "
15:        else
           ▹ Los datos muestran diferencia significativa
16:            if tStudent_1Cola_varianzasDiferentes(x,y) > 0.05 then
17:                return "x"
18:            else
19:                return "y"
20:            end if
21:        end if
22:    end if
23: else
   ▹ Las muestras no pertencen a una distribución normal
24:    if wilcoxon_2Colas(x,y) > 0.05 then
       ▹ Los datos no muestran diferencia significativa
25:        return " = "
26:    else
       ▹ Los datos muestran diferencia significativa
27:        if wilcoxon_1Cola(x,y) > 0.05 then
28:            return "x"
29:        else
30:            return "y"
31:        end if
32:    end if
33: end if
```

Tabla 12. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *B*.

Métrica	Vértices	E vs E_c	E vs I	E vs F	E vs SnW	E_c vs I	E_c vs F	E_c vs SnW	I vs F	I vs SnW	F vs SnW
PUB	16	=	I	F	E	I	F	E_c	=	I	F
	32	=	I	F	E	I	F	E_c	=	I	F
	64	E	I	F	E	I	F	E_c	=	I	F
	128	E	I	F	E	I	F	E_c	=	I	F
FUR	16	=	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	32	E	=	E	SnW	I	E_c	SnW	I	SnW	SnW
	64	E	I	E	SnW	I	E_c	SnW	I	=	SnW
	128	E	I	E	SnW	I	E_c	SnW	I	=	SnW
PROD	16	=	I	E	=	I	E_c	=	I	I	SnW
	32	E	I	E	E	I	E_c	E_c	I	I	SnW
	64	E	I	E	E	I	E_c	E_c	I	I	SnW
	128	E	I	E	E	I	E_c	E_c	I	I	SnW
PNV	16	=	=	F	=	=	F	=	F	SnW	F
	32	=	=	F	SnW	=	F	SnW	F	SnW	F
	64	=	=	F	SnW	=	F	SnW	F	SnW	F
	128	=	=	F	SnW	=	F	SnW	F	SnW	F
TC	16	E_c	E	F	E	E	F	E_c	F	SnW	F
	32	E_c	E	F	E	E	F	E_c	F	SnW	F
	64	=	E	F	E	E	F	E_c	F	SnW	F
	128	=	E	F	E	E	F	E_c	F	SnW	F

Tabla 13. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *C*.

Métrica	Vértices	E vs E_c	E vs I	E vs F	E vs SnW	E_c vs I	E_c vs F	E_c vs SnW	I vs F	I vs SnW	F vs SnW
PUB	16	=	I	F	E	I	F	E_c	=	I	F
	32	E_c	I	F	E	I	F	E_c	=	I	F
	64	E_c	I	F	E	I	F	E_c	=	I	F
	128	E_c	I	F	E	I	F	E_c	=	I	F
FUR	16	E	=	E	SnW	I	E_c	SnW	I	SnW	SnW
	32	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	64	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	128	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
PROD	16	=	I	E	E	I	E_c	E_c	I	I	SnW
	32	=	I	E	E	I	E_c	E_c	I	I	SnW
	64	E	I	E	E	I	E_c	E_c	I	I	SnW
	128	E	I	E	E	I	E_c	E_c	I	I	SnW
PNV	16	=	=	F	SnW	=	F	SnW	F	SnW	F
	32	=	E	F	SnW	E_c	F	SnW	F	SnW	F
	64	=	E	F	SnW	E_c	F	SnW	F	SnW	F
	128	E_c	E	F	SnW	E_c	F	SnW	F	SnW	F
TC	16	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	32	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	64	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	128	E_c	E	F	E	E_c	F	E_c	F	SnW	F

Tabla 14. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *D* utilizando un esquema sin restricción de energía.

Métrica	Vértices	E vs E_c	E vs I	E vs F	E vs SnW	E_c vs I	E_c vs F	E_c vs SnW	I vs F	I vs SnW	F vs SnW
PUB	16	=	I	F	E	I	F	E_c	=	I	F
	32	E_c	I	F	E	I	F	E_c	=	I	F
	64	E_c	I	F	E	I	F	E_c	=	I	F
	128	E_c	I	F	E	I	F	E_c	=	I	F
FUR	16	E	=	E	SnW	I	E_c	SnW	I	SnW	SnW
	32	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	64	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	128	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
PROD	16	=	I	E	E	I	E_c	E_c	I	I	SnW
	32	E	I	E	E	I	E_c	E_c	I	I	SnW
	64	E	I	E	E	I	E_c	E_c	I	I	SnW
	128	E	I	E	E	I	E_c	=	I	I	SnW
PNV	16	=	=	F	SnW	=	F	SnW	F	SnW	F
	32	=	E	F	SnW	E_c	F	SnW	F	SnW	F
	64	=	E	F	SnW	E_c	F	SnW	F	SnW	F
	128	=	E	F	SnW	E_c	F	SnW	F	SnW	F
TC	16	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	32	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	64	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	128	E_c	E	F	E	E_c	F	E_c	F	SnW	F

Tabla 15. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *D* utilizando un esquema con carga completa de energía.

Métrica	Vértices	E vs E_c	E vs I	E vs F	E vs SnW	E_c vs I	E_c vs F	E_c vs SnW	I vs F	I vs SnW	F vs SnW
PUB	16	E	I	F	E	I	F	SnW	=	I	F
	32	E	I	F	E	I	F	E_c	=	I	F
	64	=	I	F	E	I	F	E_c	=	I	F
	128	E_c	I	F	E	I	F	E_c	=	I	F
FUR	16	=	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	32	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	64	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	128	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
PROD	16	E	I	E	SnW	I	E_c	SnW	I	I	SnW
	32	E	I	E	E	I	E_c	=	I	I	SnW
	64	E	I	E	E	I	E_c	E_c	I	I	SnW
	128	E	I	E	E	I	E_c	=	I	I	SnW
PNV	16	=	E	=	E	E_c	=	E_c	F	SnW	F
	32	E_c	E	F	SnW	E_c	F	=	F	SnW	F
	64	=	=	F	SnW	E_c	F	SnW	F	SnW	F
	128	E_c	E	F	SnW	E_c	F	SnW	F	SnW	F
TC	16	=	E	=	E	E_c	=	E_c	F	SnW	F
	32	E_c	E	F	E	E_c	=	E_c	F	SnW	F
	64	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	128	E_c	E	F	E	E_c	F	E_c	F	SnW	F

Tabla 16. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *D* utilizando un esquema con carga aleatoria de energía.

Métrica	Vértices	E vs E_c	E vs I	E vs F	E vs SnW	E_c vs I	E_c vs F	E_c vs SnW	I vs F	I vs SnW	F vs SnW
PUB	16	E	I	F	E	I	F	SnW	=	I	F
	32	E	I	F	E	I	F	=	=	I	F
	64	E	I	F	E	I	F	E_c	=	I	F
	128	E_c	I	F	E	I	F	E_c	=	I	F
FUR	16	E_c	I	E	SnW	E_c	F	E_c	I	SnW	SnW
	32	E	I	E	SnW	I	=	SnW	I	SnW	SnW
	64	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
	128	E	I	E	SnW	I	E_c	SnW	I	SnW	SnW
PROD	16	E_c	I	E	SnW	I	E_c	SnW	I	I	SnW
	32	E	I	E	=	I	=	SnW	I	I	SnW
	64	E	I	E	E	I	E_c	=	I	I	SnW
	128	E	I	E	E	I	E_c	=	I	I	SnW
PNV	16	=	E	=	E	E_c	=	E_c	F	SnW	F
	32	E_c	E	F	=	E_c	=	E_c	F	SnW	F
	64	E_c	E	F	SnW	E_c	F	=	F	SnW	F
	128	E_c	E	F	SnW	E_c	F	SnW	F	SnW	F
TC	16	E	E	F	E	=	F	SnW	F	SnW	F
	32	E_c	E	F	E	E_c	F	E_c	F	SnW	F
	64	=	E	F	E	E_c	F	E_c	F	SnW	F
	128	E_c	E	F	E	E_c	F	E_c	F	SnW	F

Tabla 17. Resultados del análisis estadístico de las métricas obtenidas por pares de algoritmos con el escenario *E*, con diferentes tiempos de llegada de los vértices, para varios algoritmos y para $n = 128$.

Métrica	Tiempo de llegada (s)	E vs E_c	E vs F	E vs SnW	E_c vs F	E_c vs SnW	F vs SnW
PUB	Distribución uniforme[0, 0]	E_c	F	E	F	E_c	F
	Distribución uniforme[0, 10k]	E_c	F	E	F	E_c	F
	Distribución uniforme[0, 20k]	E_c	F	E	F	E_c	F
	Distribución uniforme[0, 40k]	E_c	F	E	F	E_c	F
FUR	Distribución uniforme[0, 0]	E	E	SnW	E_c	SnW	SnW
	Distribución uniforme[0, 10k]	E	E	SnW	E_c	SnW	SnW
	Distribución uniforme[0, 20k]	E	E	SnW	E_c	SnW	SnW
	Distribución uniforme[0, 40k]	E	E	SnW	E_c	SnW	SnW
PROD	Distribución uniforme[0, 0]	E	E	E	E_c	=	SnW
	Distribución uniforme[0, 10k]	E	E	E	E_c	=	SnW
	Distribución uniforme[0, 20k]	E	E	E	E_c	=	SnW
	Distribución uniforme[0, 40k]	E	E	E	E_c	=	SnW
PNV	Distribución uniforme[0, 0]	=	F	SnW	F	SnW	F
	Distribución uniforme[0, 10k]	=	F	SnW	F	SnW	F
	Distribución uniforme[0, 20k]	E_c	F	SnW	F	SnW	F
	Distribución uniforme[0, 40k]	E_c	F	SnW	F	SnW	F
TC	Distribución uniforme[0, 0]	E_c	F	E	F	E_c	F
	Distribución uniforme[0, 10k]	E_c	F	E	F	E_c	F
	Distribución uniforme[0, 20k]	E_c	F	E	F	E_c	F
	Distribución uniforme[0, 40k]	E_c	F	E	F	E_c	F

Anexo D. Gráficas de los resultados

En este anexo se muestran todas las figuras que se omitieron en el Capítulo 6, con una descripción de las tendencias presentadas en cada una de ellas. Las figuras se omitieron con la finalidad de presentar un capítulo de resultados experimentales (Capítulo 6) conciso, ya que las figuras omitidas presentan tendencias similares a pesar de utilizar parámetros de configuración diferentes entre sí.

En la sección 5.4 se definen los distintos escenarios de simulación. Las gráficas que presentan los resultados de las métricas utilizan diagramas de caja con muesca, los cuales se describen en la Figura 14.

En todos los escenarios se omitieron las figuras que muestran las métricas de FUR, PUB y el tiempo de convergencia del escenario, excepto el escenario *A* en donde se presentaron las gráficas de todas las métricas. A continuación se muestra cada una de las figuras que se omitieron, desde el escenario *B* al escenario *E*.

D.1. Resultados de simulación para el escenario B

En esta sección se presentan las figuras con los resultados de las métricas FUR, PUB, y el tiempo de convergencia al utilizar el escenario *B*.

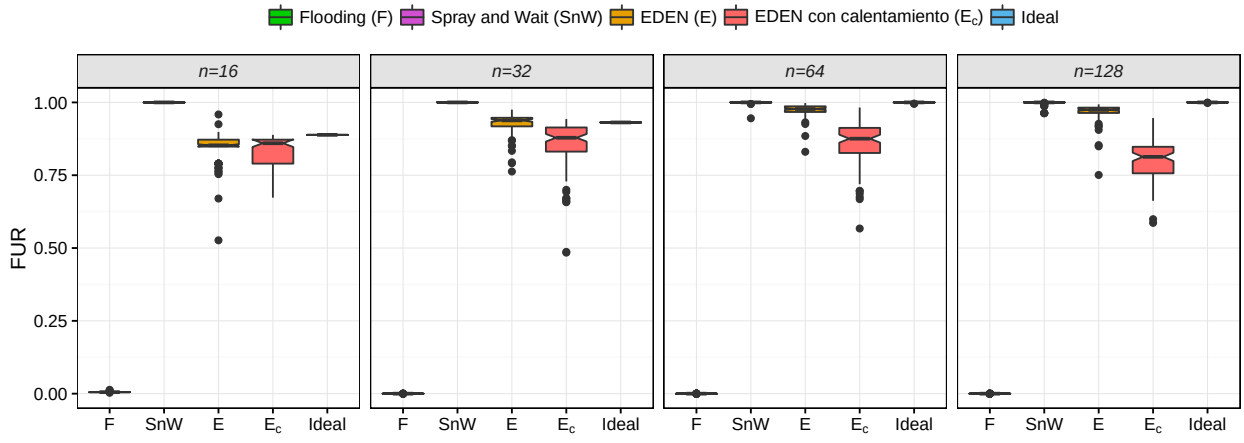


Figura 54. Factor de Uniformidad Relativo (FUR) para el escenario *B*, para todos los algoritmos y valores de *n*.

La Figura 54 muestra el FUR para todos los algoritmos y valores de *n*. El FUR para *epidemic flooding* es muy cercano a cero y es por mucho el más bajo en comparación

con los otros algoritmos para todo n. Por otro lado, SnW obtiene un FUR de prácticamente uno, siendo el mejor de todos los algoritmos para todo n. EDEN es mejor que EDEN_C para todo n, excepto para $n = 16$, donde no hay diferencia significativa.

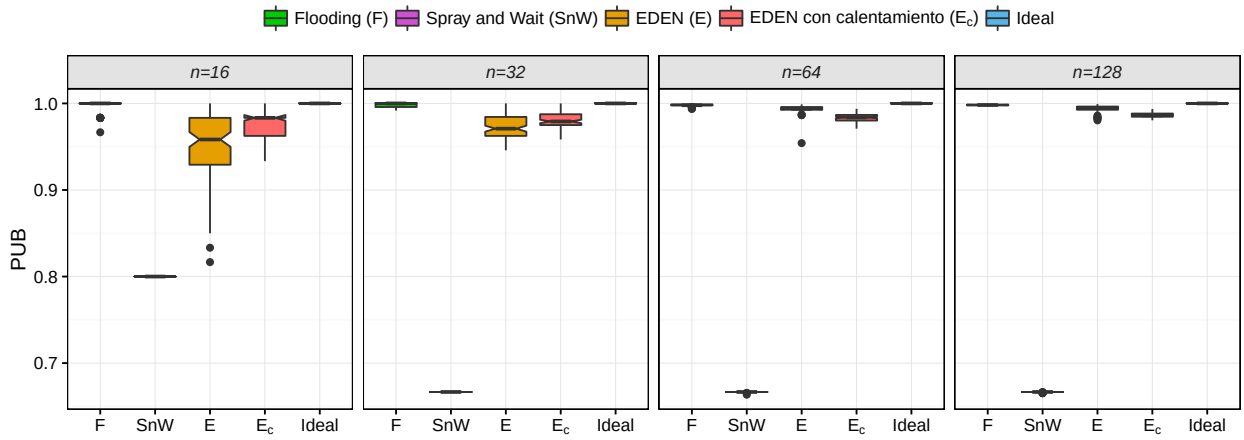


Figura 55. Porcentaje de Utilización del Buffer (PUB) para el escenario *B*, para todos los algoritmos y valores de n.

La Figura 55 muestra el PUB para todos los algoritmos y valores de n. El PUB para SnW es el más bajo para todo n, solo para $n = 16$ mejora considerablemente, pero aún así al compararlo con los otros algoritmos es el que presenta los peores resultados para todo n. El PUB de *epidemic flooding*, EDEN y EDEN_C son muy cercanos a uno, pero *epidemic flooding* muestra siempre el mejor rendimiento. Al comparar EDEN y EDEN_C, se puede observar que para $n = 16$ y $n = 32$, son muy similares; mientras que, para $n = 64$ y $n = 128$, EDEN es mejor.

La Figura 56 muestra el tiempo de convergencia de la red para todos los algoritmos y valores de n. *Epidemic flooding* produce los peores resultados para todo n, incluso se puede observar que la red no converge ya que el tiempo de simulación final es igual al tiempo de convergencia, es decir, la red no converge utilizando este algoritmo para este escenario. Después, esta EDEN_C, el cual en general es ligeramente peor que EDEN. Finalmente, SnW obtiene los mejores resultados para todo n al obtener siempre el tiempo de convergencia más pequeño. Esto sucede porque SnW transmite muy pocos mensajes en comparación con los otros algoritmos.

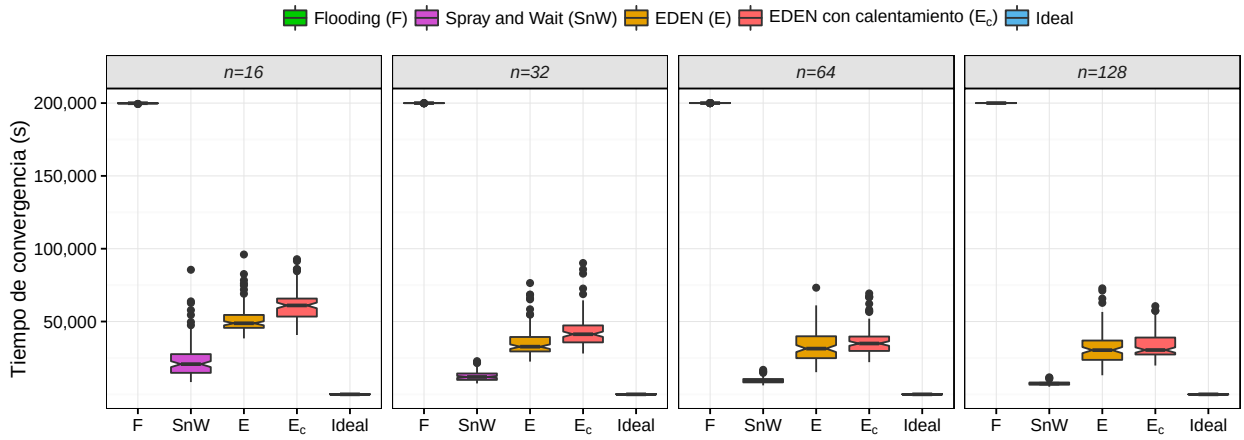


Figura 56. Tiempo de convergencia de la red para el escenario *B*, para todos los algoritmos y valores de *n*.

D.2. Resultados de simulación para el escenario C

En esta sección se presentan las figuras con los resultados de las métricas FUR, PUB, y el tiempo de convergencia al utilizar el escenario *C*.

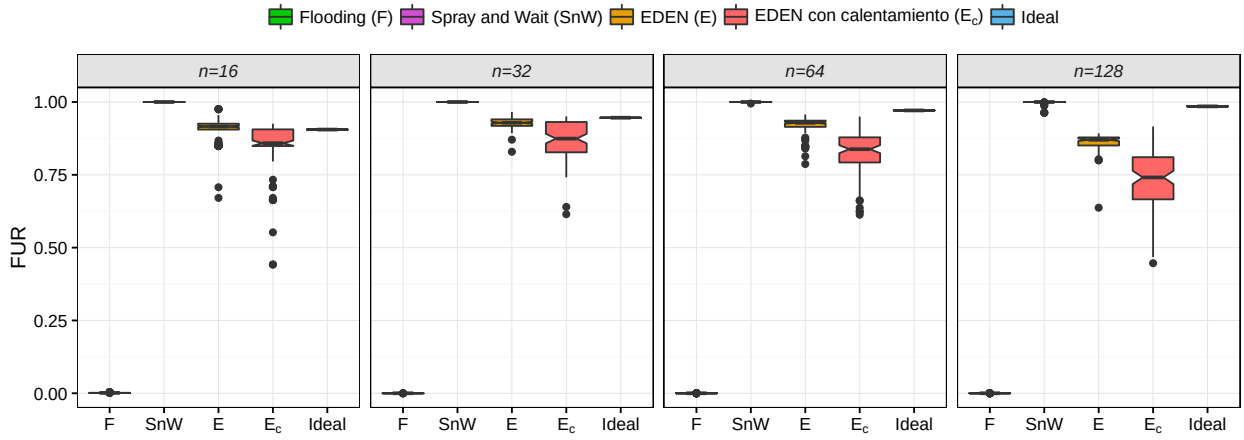


Figura 57. Factor de Uniformidad Relativo (FUR) para el escenario *C*, para todos los algoritmos y valores de *n*.

La Figura 57 muestra el FUR para todos los algoritmos y valores de *n*. El FUR para *epidemic flooding* es muy cercano a cero y es por mucho el más bajo en comparación con los otros algoritmos para todo *n*. Por otro lado, SnW obtiene un FUR de prácticamente uno siendo el mejor de todos los algoritmos para todo *n*. Entre los algoritmos propuestos, EDEN es siempre mejor que EDEN_C para todo *n*.

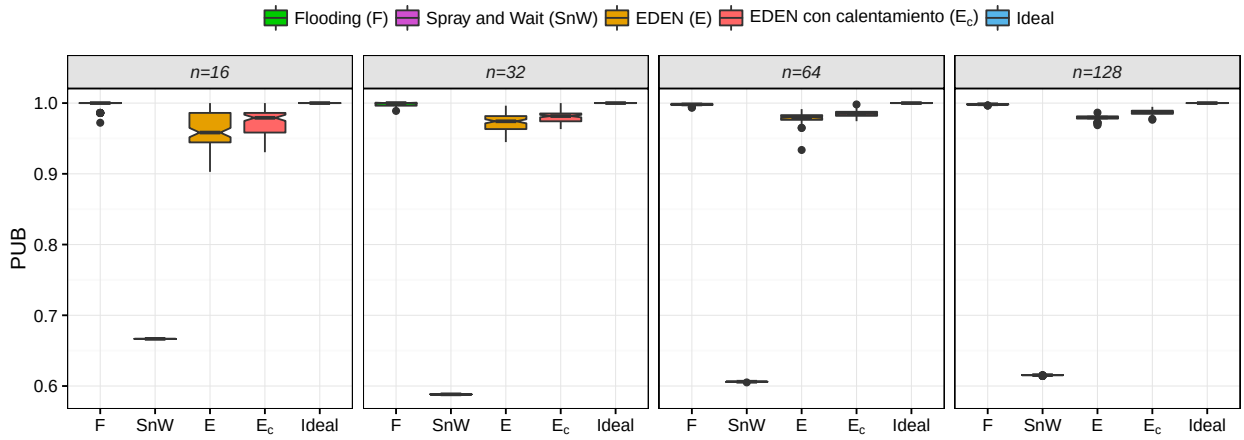


Figura 58. Porcentaje de Utilización del Buffer (PUB) para el escenario *C*, para todos los algoritmos y valores de *n*.

La Figura 58 muestra el PUB para todos los algoritmos y valores de *n*. El PUB para SnW es muy bajo para todo *n*, solo para $n = 16$ mejora ligeramente, pero aún así es el que presenta peores resultados en comparación con los otros algoritmos. El PUB de *epidemic flooding*, EDEN y EDEN_C son muy cercanos a uno, pero el primero siempre obtiene el mejor resultado entre los tres. Al comparar EDEN y EDEN_C, se puede observar que EDEN_C es ligeramente mejor para todo *n*.

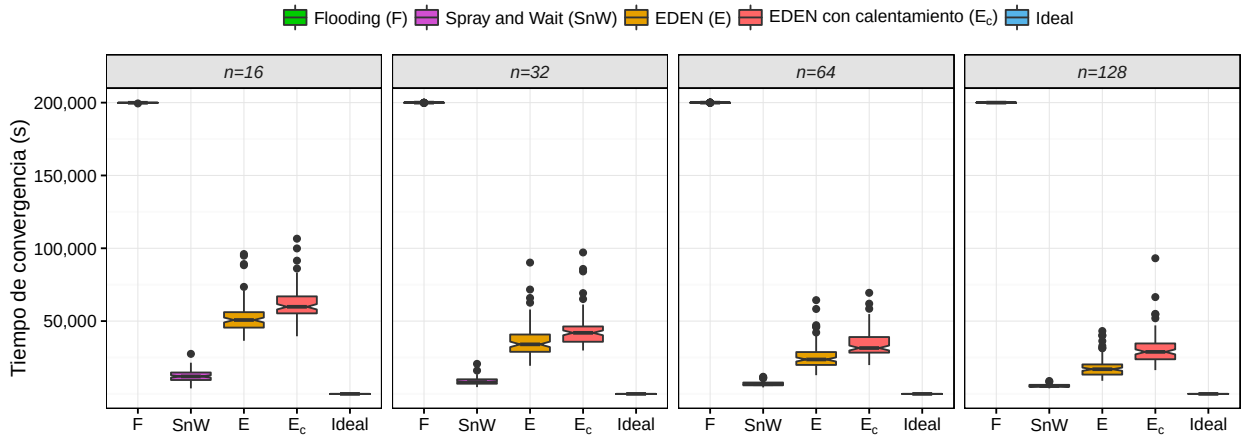


Figura 59. Tiempo de convergencia de la red para el escenario *C*, para todos los algoritmos y valores de *n*.

La Figura 59 muestra el tiempo de convergencia de la red para todos los algoritmos y valores de *n*. *Epidemic flooding* produce los peores resultados para todo *n*, con valores iguales o cercanos a $200,000$ segundos (el límite del tiempo de simulación), ya

que la red no converge utilizando este algoritmo para este escenario. Por el contrario, SnW obtiene los mejores resultados para todo n, al obtener siempre el tiempo de convergencia más pequeño; mientras que EDEN_c siempre es ligeramente peor que EDEN para todo n.

D.3. Resultados de simulación para el escenario D

En esta sección se presentan todas las figuras con los resultados de la simulación al utilizar el escenario *D* que se omitieron en el Capítulo 6. Para este escenario en particular se utilizan tres diferentes esquemas de energía para los vértices en la red descritos en la sección 5.3.7.

D.3.1. Esquema sin restricción de energía

Las siguientes figuras muestran los resultados de las métricas FUR, PUB, FUR×PUB, y el tiempo de convergencia utilizando el primer esquema de energía, que considera que los vértices en la red cuentan con energía sin restricciones.

La Figura 60 muestra el FUR para todos los algoritmos y valores de n. El FUR para *epidemic flooding* es muy cercano a 0 y es por mucho el más bajo en comparación con los otros algoritmos para todo n. Por el contrario, SnW obtiene un FUR de prácticamente uno siendo el mejor de todos los algoritmos para todo n. EDEN es siempre mejor que EDEN_c para todo n. Para ambos algoritmos se nota una tendencia decreciente conforme aumenta n.

La Figura 61 muestra el PUB para todos los algoritmos y valores de n. El PUB para SnW es muy bajo para todo n, sólo para $n = 16$ mejora ligeramente, pero aún así es el que presenta peores resultados en comparación con los otros algoritmos. El PUB de *epidemic flooding*, EDEN y EDEN_c son muy cercanos a uno, pero el primero siempre obtiene el mejor resultado entre los tres. Entre EDEN y EDEN_c, el segundo es ligeramente mejor para todo n.

La Figura 62 muestra el FUR×PUB para todos los algoritmos y valores de n. El producto de *epidemic flooding* es cero siendo el más bajo para todo n. El segundo peor producto lo genera SnW variando con valores entre 0.7 y 0.65. EDEN y EDEN_c son los que obtienen mejores resultados en esta métrica, incluso muy cercanos al Ideal. Sin

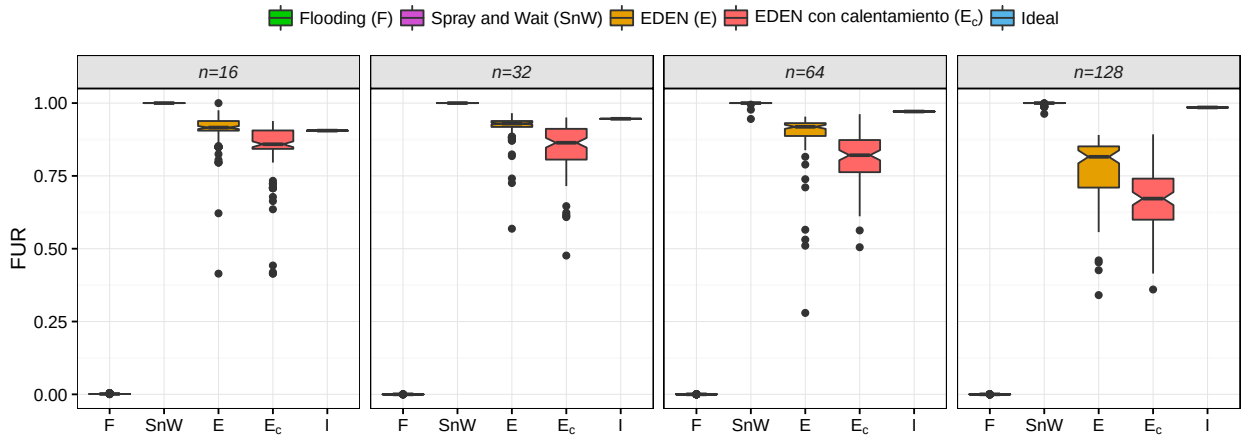


Figura 60. Factor de Uniformidad Relativo (FUR) para el escenario *D*, sin restricción de energía, para todos los algoritmos y valores de *n*.

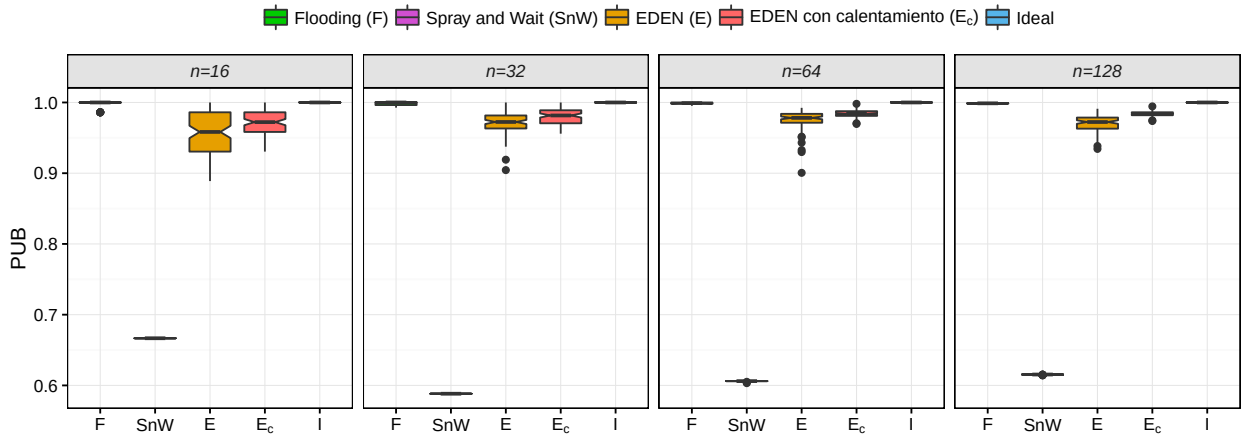


Figura 61. Porcentaje de Utilización del Buffer (PUB) para el escenario *D*, sin restricción de energía, para todos los algoritmos y valores de *n*.

embargo, a medida que aumenta *n*, mayor es la diferencia de estos dos algoritmos con el Ideal.

La Figura 63 muestra el tiempo de convergencia de la red para todos los algoritmos y valores de *n*. *Epidemic flooding* produce los peores resultados para todo *n*. Por el contrario, SnW obtiene los mejores resultados para todo *n*, al obtener siempre el tiempo de convergencia más pequeño; mientras que $EDEN_C$ siempre es ligeramente peor que EDEN para todo *n*.

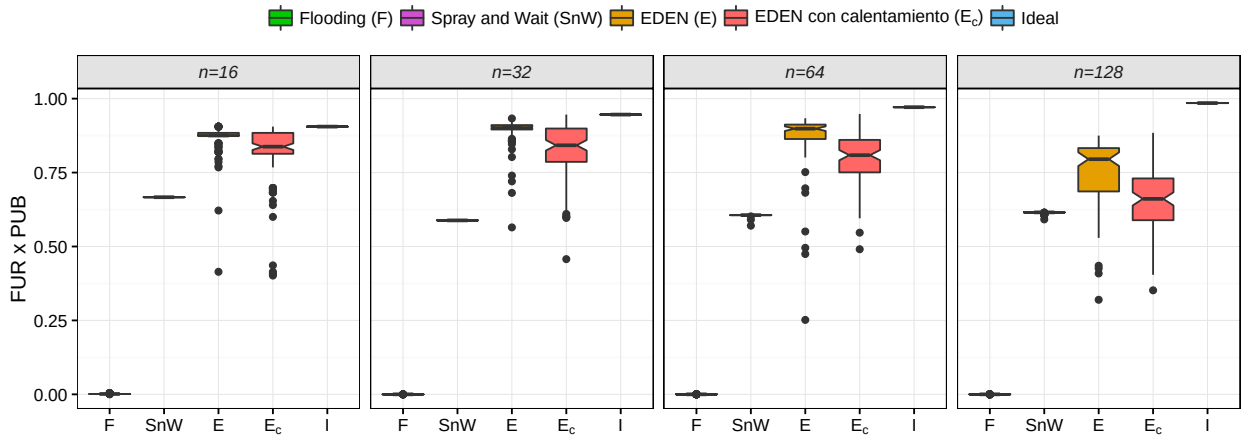


Figura 62. Producto del FUR y PUB (FUR×PUB) para el escenario *D*, sin restricción de energía, para todos los algoritmos y valores de *n*.

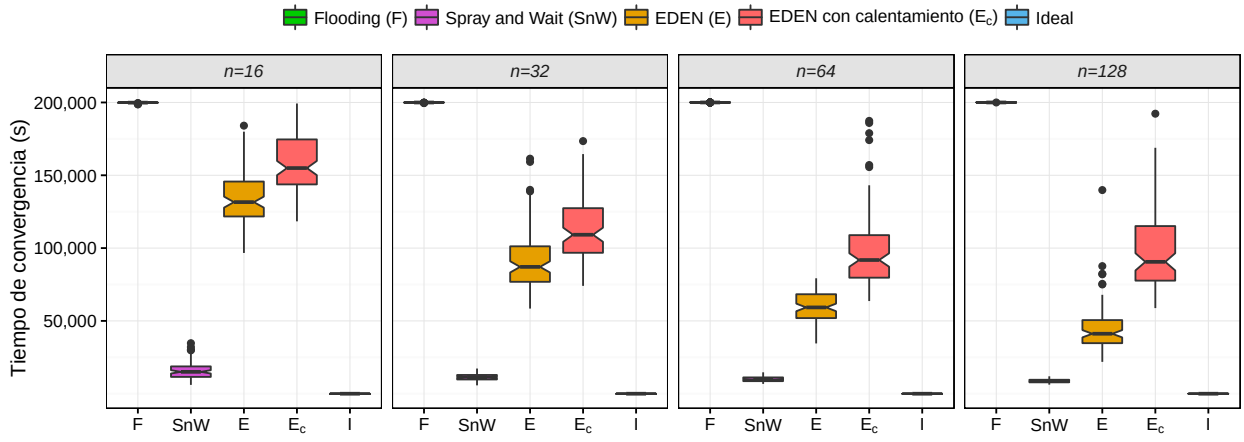


Figura 63. Tiempo de convergencia de la red para el escenario *D*, sin restricción de energía, para todos los algoritmos y valores de *n*.

D.3.2. Esquema con carga de energía completa

Las siguientes figuras muestran los resultados de las métricas FUR, PUB, FUR×PUB, y el tiempo de convergencia utilizando el segundo esquema de energía, que considera que todos los vértices en la red cuentan con carga completa de energía al iniciar la simulación.

La Figura 64 muestra el FUR para todos los algoritmos y valores de *n*. El FUR para *epidemic flooding* es muy cercano a cero y es el algoritmo con peor desempeño para todo *n* por un amplio margen. Por el contrario, SnW obtiene un FUR de prácticamente

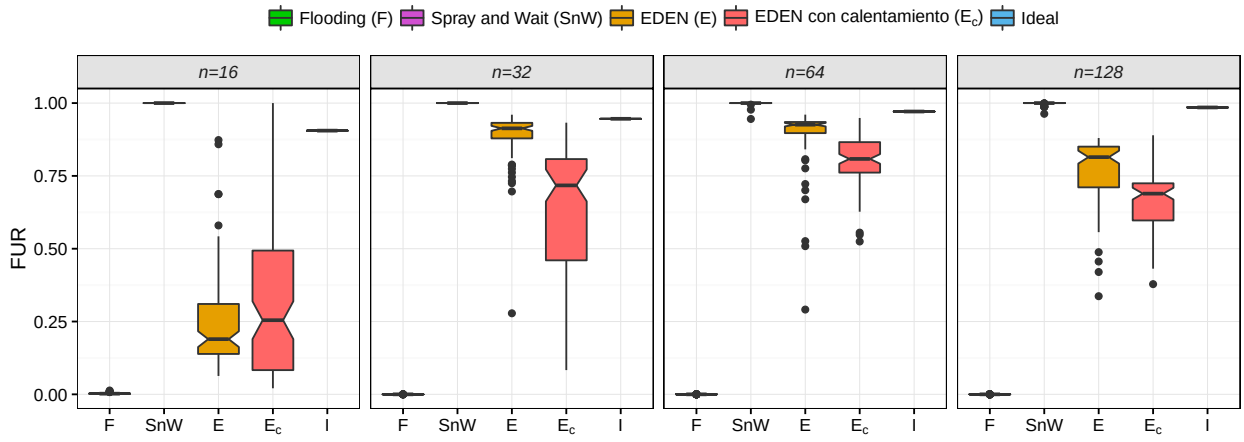


Figura 64. Factor de Uniformidad Relativo (FUR) para el escenario *D*, con carga de energía completa, para todos los algoritmos y valores de *n*.

uno, siendo el mejor de todos los algoritmos para todo *n*, incluso superando los valores del Ideal. Esto se debe a que SnW utiliza un número de copias por mensaje muy por debajo al número de copias óptimo utilizado en el algoritmo Ideal. Por lo tanto, para SnW la uniformidad en el número de copias no entra en conflicto con el espacio de almacenamiento, mientras que para el Ideal sí. Finalmente, EDEN siempre es mayor a $EDEN_C$ para todo *n*, excepto para $n = 16$, donde no hay diferencia significativa entre ellos.

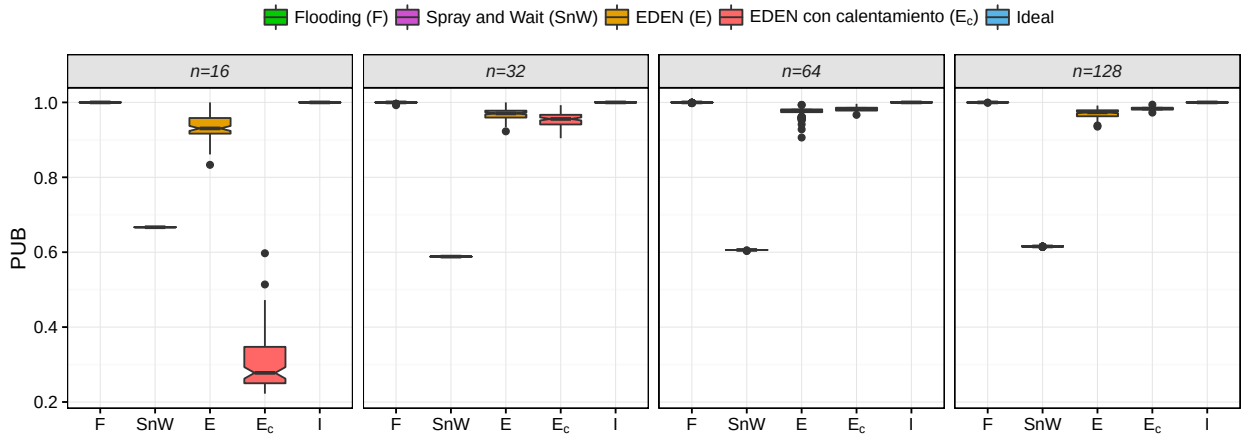


Figura 65. Porcentaje de Utilización del Buffer (PUB) para el escenario *D*, con carga de energía completa, para todos los algoritmos y valores de *n*.

La Figura 65 muestra el PUB para todos los algoritmos y valores de *n*. El PUB para SnW es el más bajo para todo *n*, excepto para $n = 16$, donde $EDEN_C$ obtiene el peor

resultado. En general, el PUB de *epidemic flooding*, EDEN y $EDEN_c$ son muy cercanos a uno, pero el primero siempre obtiene el mejor resultado entre los tres. Entre EDEN y $EDEN_c$, el primero es ligeramente mejor para todo n. Particularmente, para $n = 16$, el desempeño de $EDEN_c$ decrece considerablemente debido a que la energía de los vértices no dura lo suficiente para que $EDEN_c$ termine de repartir todas las copias requeridas.

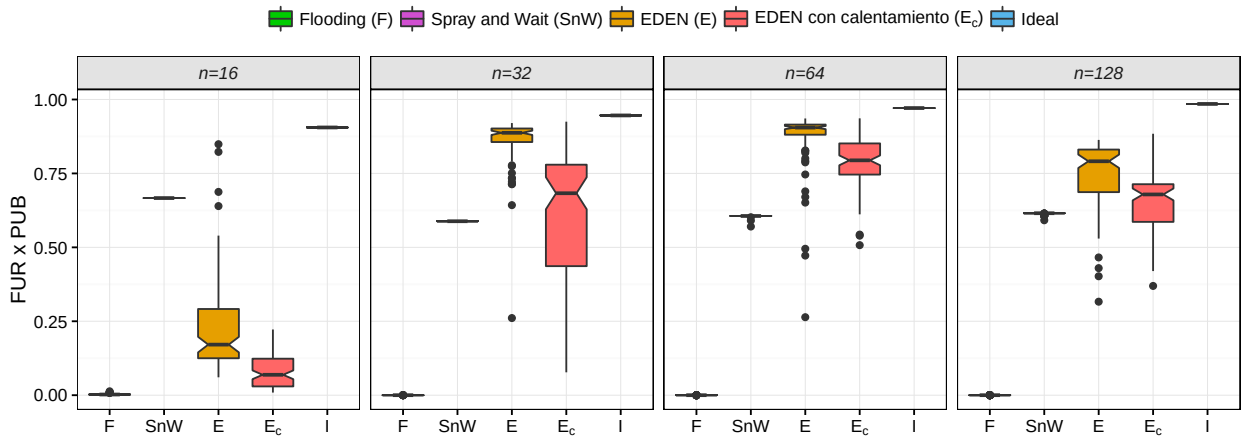


Figura 66. Producto del FUR y PUB (FUR×PUB) para el escenario *D*, con carga de energía completa, para todos los algoritmos y valores de n.

La Figura 66 muestra el FUR×PUB para todos los algoritmos y valores de n. El producto de *epidemic flooding* es el peor para todo n. El producto para SnW es el más alto solamente para $n = 16$. Note que al combinar ambas métricas SnW nunca es mejor que el Ideal a diferencia de lo que ocurre cuando solo se considera el FUR. EDEN y $EDEN_c$ decrementan su desempeño considerablemente para $n = 16$. En general, para los demás valores de n siempre son mejores que SnW. EDEN es el mejor de los algoritmos propuestos, incluso generando valores muy cercanos al Ideal para $n = 32$ y $n = 64$.

La Figura 67 muestra el tiempo de convergencia de la red para todos los algoritmos y valores de n. Una carga de energía completa dura aproximadamente a lo más 75,000 segundos. En general, el tiempo de convergencia para *Epidemic flooding*, EDEN y $EDEN_c$ es igual a este tiempo para todo n. En estos casos, los algoritmos no lograron llegar a un estado de convergencia en la red antes de agotar la energía de los vértices. De estos tres algoritmos, solamente EDEN para $n = 64$ y $n = 128$ logró una convergencia de la red en menor tiempo. En particular, SnW generó los tiempos de convergencia

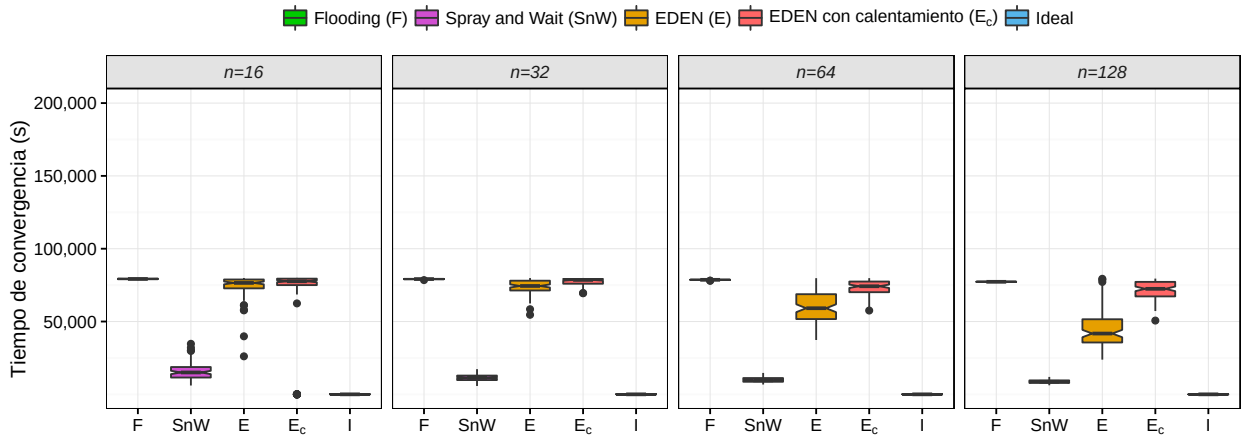


Figura 67. Tiempo de convergencia de la red para el escenario *D*, con carga de energía completa, para todos los algoritmos y valores de *n*.

más pequeños, en promedio de 25,000 segundos, siendo el mejor para todo *n*.

D.3.3. Esquema de energía con carga aleatoria

Las siguientes figuras muestran los resultados de las métricas FUR, PUB, FUR×PUB, y el tiempo de convergencia utilizando el tercer esquema de energía, que considera que los todos los vértices en la red cuentan con carga aleatoria de energía al iniciar la simulación.

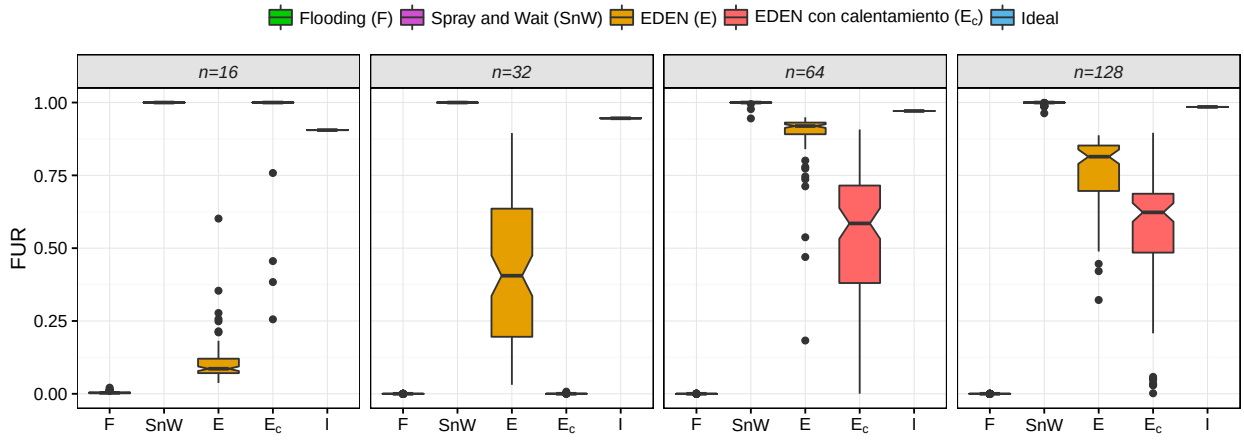


Figura 68. Factor de Uniformidad Relativo (FUR) para el escenario *D*, con carga de energía aleatoria, para todos los algoritmos y valores de *n*.

La Figura 68 muestra el FUR para todos los algoritmos y valores de *n*. El FUR para

epidemic flooding es muy cercano a cero y es el algoritmo con peor desempeño para todo n. Por el contrario, SnW obtiene un FUR de prácticamente uno, generando los mejores valores de entre los algoritmos para todo n. EDEN siempre es mayor a $EDEN_c$ para todo n, excepto para $n = 16$ donde lo inverso sucede. En este esquema de energía es notorio como EDEN y $EDEN_c$ deterioran su desempeño para $n = 16$ y $n = 32$.

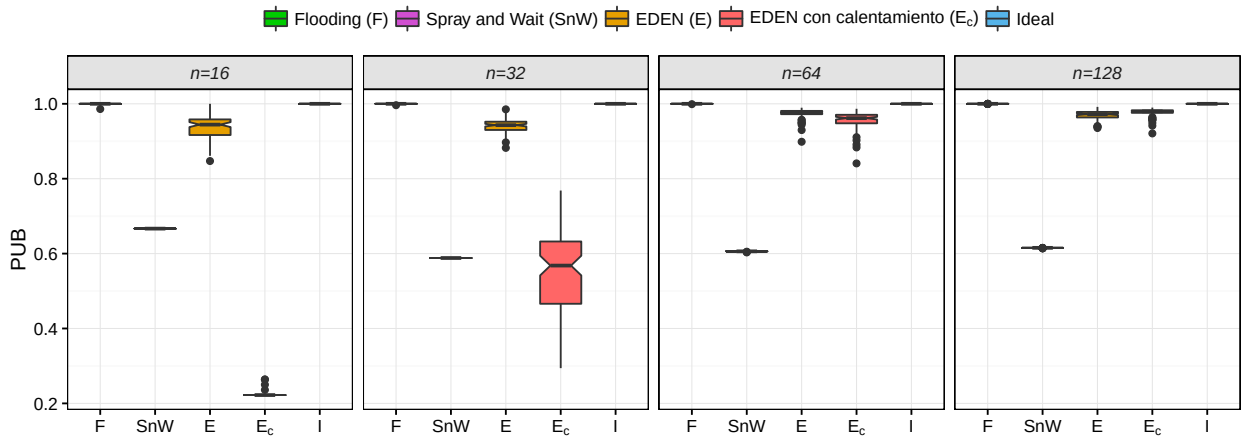


Figura 69. Porcentaje de Utilización del Buffer (PUB) para el escenario *D*, con carga de energía aleatoria, para todos los algoritmos y valores de n.

La Figura 69 muestra el PUB para todos los algoritmos y valores de n. El PUB para SnW es el más bajo para $n = 64$ y $n = 128$. Similarmente, el PUB para $EDEN_c$ es el más bajo para $n = 16$ y $n = 32$. Pero en contraste, los valores de SnW oscilan entre 0.65 y 0.58 para todo n, mientras que los valores de $EDEN_c$ van desde 0.2 para $n = 16$, hasta uno para $n = 128$. En general, el PUB de *epidemic flooding* es el mejor con valores muy cercanos a uno para todo n. Seguido por EDEN que obtiene valores a lo menos de 0.9 para todo n. Por lo tanto, entre EDEN y $EDEN_c$, el primero es mejor para todo n, excepto para $n = 64$ y $n = 128$ donde ambos obtienen valores muy similares.

La Figura 70 muestra el FUR×PUB para todos los algoritmos y valores de n. El producto de *epidemic flooding* es el peor para todo n. El producto de SnW es el más alto solamente para $n = 16$ y $n = 32$. Similarmente, el producto generado por EDEN es el más alto solamente para $n = 64$ y $n = 128$. En general, EDEN es mejor que $EDEN_c$ para todo n, excepto para $n = 16$, donde lo inverso ocurre. Para este esquema de energía, el decremento en el desempeño de ambos algoritmos es considerable para los casos con menor número de vértices.

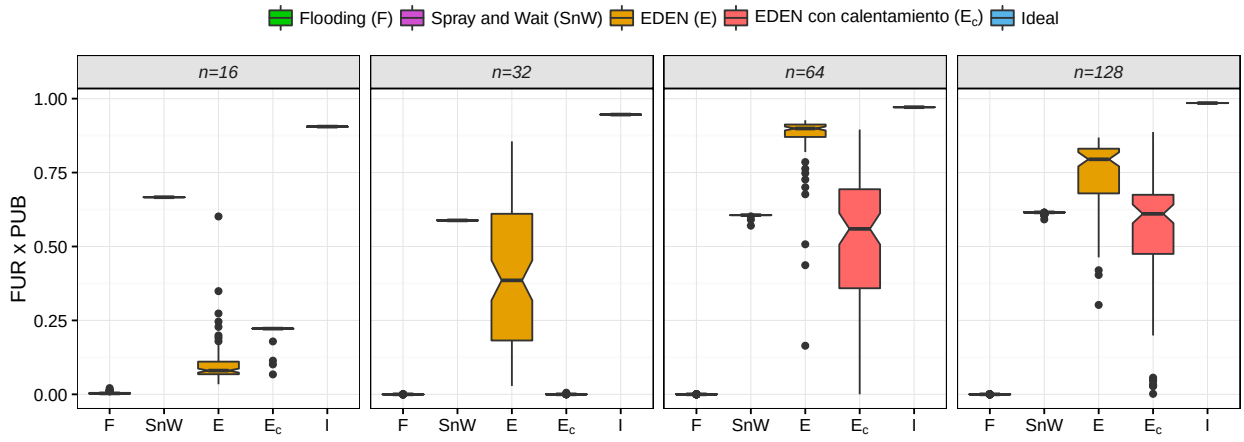


Figura 70. Producto del FUR y PUB (FUR×PUB) para el escenario *D*, con carga de energía aleatoria, para todos los algoritmos y valores de *n*.

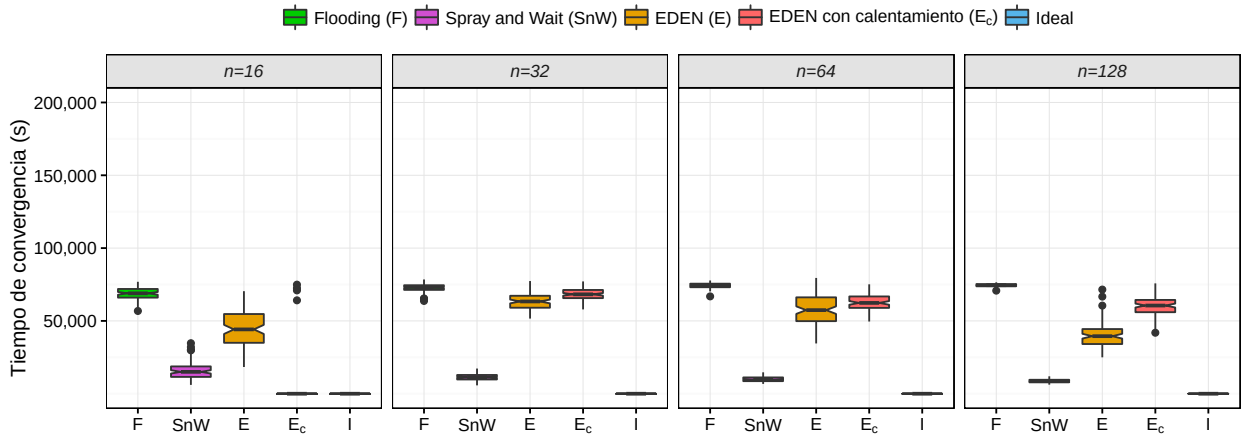


Figura 71. Tiempo de convergencia de la red para el escenario *D*, con carga de energía aleatoria, para todos los algoritmos y valores de *n*.

La Figura 71 muestra el tiempo de convergencia de la red para todos los algoritmos y valores de *n*. El tiempo de convergencia para *Epidemic flooding* es aproximadamente $75,000$ segundos para todo *n*, siendo el algoritmo con el peor desempeño. SnW produce los tiempos de convergencia más pequeños, en promedio de 20,000 segundos, siendo el mejor para todo *n*. EDEN_C converge más rápido que EDEN para $n = 16$, lo contrario sucede para $n = 128$. Para $n = 32$ y $n = 64$ los tiempos son similares entre sí.

Aunque en promedio EDEN_C para $n = 16$ muestra tiempos de convergencia muy pequeños, esto es probablemente consecuencia de dos factores. Primero, el número de vértices es muy pequeño en este tamaño de escenario; segundo, los vértices agotan

su batería aleatoriamente. Por lo tanto, la frecuencia entre contactos es muy grande. Por lo que no ocurren cambios en el buffer de los vértices por largos periodos de tiempo generando estos tiempos de convergencia engañosamente pequeños. Lo anterior se puede observar en la Figura 27, donde se contrastan los tiempos de convergencia de la red con los tiempos de agotamiento de la energía.

D.4. Resultados de simulación para el escenario E

En esta sección se presentan las figuras con los resultados de las métricas FUR, PUB, y FUR×PUB al utilizar el escenario *E*. Para este escenario se fija $n = 128$ y se varía el tiempo de llegada de los vértices a la red utilizando cuatro distribuciones uniformes de los tiempos de llegada (ver sección 5.3.8).

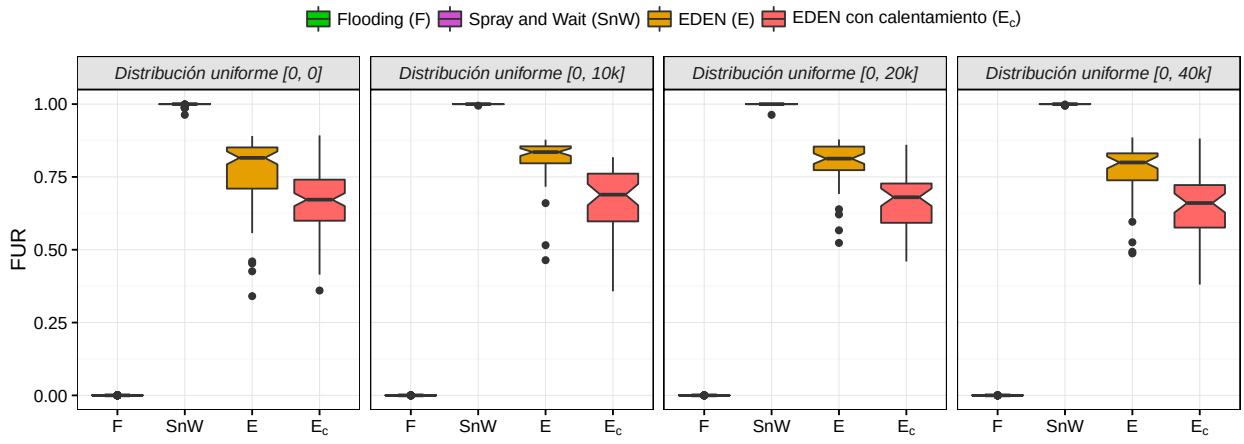


Figura 72. Factor de Uniformidad Relativo (FUR) para el escenario *E*, para varios algoritmos y para $n = 128$.

La Figura 72 muestra el FUR para todos los algoritmos y para $n = 128$. El FUR para *epidemic flooding* es muy cercano a cero y es el algoritmo con peor desempeño para las diferentes distribuciones de los tiempos de llegada de los vértices a la red. Por el contrario, SnW obtiene un FUR de prácticamente uno, generando los mejores valores de entre los algoritmos para todas las distribuciones. EDEN siempre es mayor a EDEN_C para todas las distribuciones.

La Figura 73 muestra el PUB para todos los algoritmos y para $n = 128$. El PUB para SnW es el más bajo, en promedio 0.63, para todas las distribuciones. El PUB de *epidemic flooding* es el mejor con valores muy cercanos a uno para todas las distribuciones.

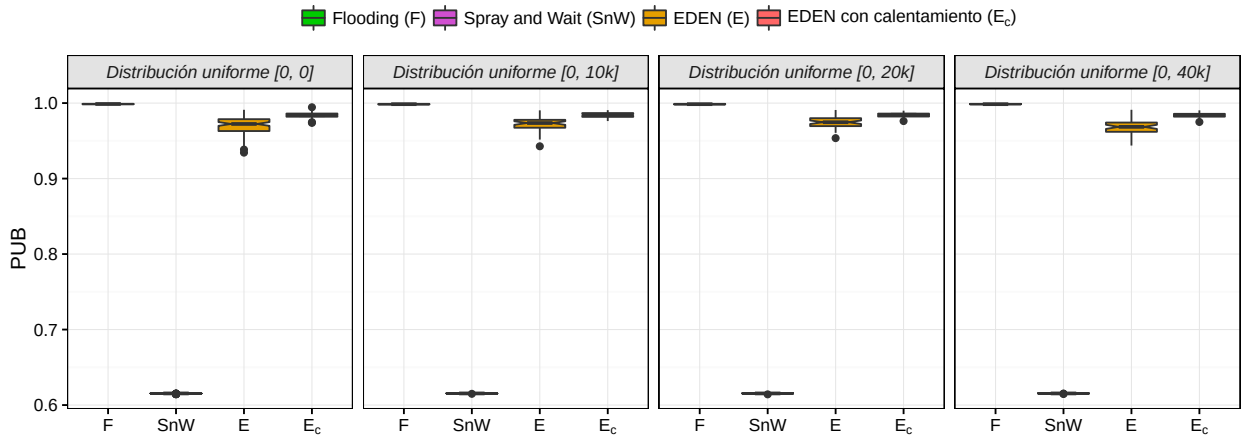


Figura 73. Porcentaje de Utilización del Buffer (PUB) para el escenario *E*, para varios algoritmos y para $n = 128$.

EDEN_C es ligeramente mejor que EDEN para todas las distribuciones. El PUB de ambos algoritmos nunca es menor a 0.98.

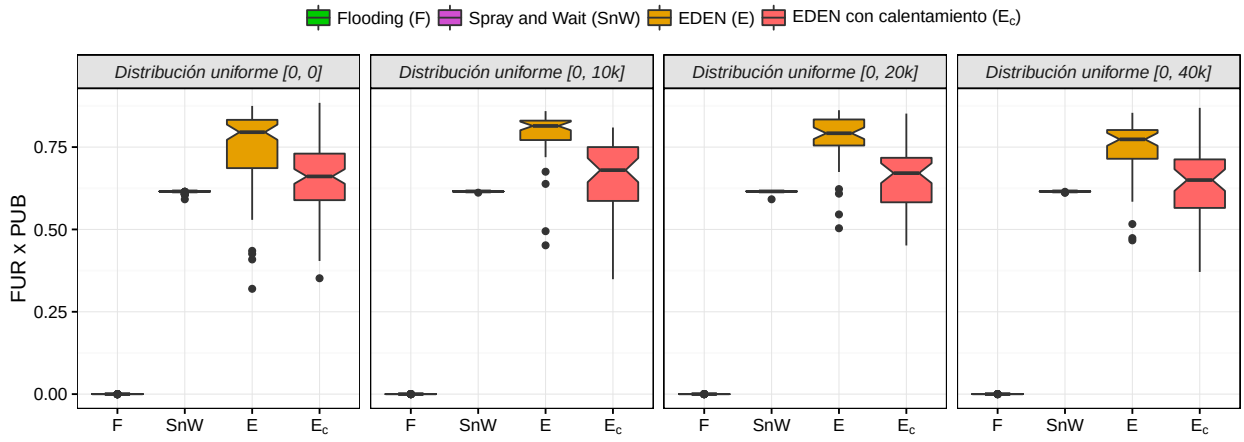


Figura 74. Producto del FUR y PUB (FUR×PUB) para el escenario *E*, para varios algoritmos y para $n = 128$.

La Figura 74 muestra el FUR×PUB para todos los algoritmos y para $n = 128$. El producto de *epidemic flooding* es el peor para todas las distribuciones con valores practicamente de cero. El producto de SnW es el segundo peor para todas las distribuciones con valores en promedio de 0.63. Entre EDEN_C y EDEN, el segundo siempre es el mejor. Incluso, el producto generado por EDEN, en promedio 0.82 es el más alto para todas las distribuciones. EDEN_C por su parte obtiene valores en promedio de 0.70 para todas las distribuciones pero con un alta variabilidad en sus valores.