FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Tesis de Maestría

Evaluación del algoritmo Contact Graph Routing en el caso de constelaciones satelitales LEO.

Autor:

Ing. Juan Pablo Astudillo León

Director:

Dr. Ing. José Ignacio Alvarez-Hamelin

Tesis realizada como parte de los requisitos para la obtención del título de Mágister en Ingeniería en Telecomunicaciones

Grupo de redes complejas y comunicación de datos (CoNexDat)

Facultad de Ingeniería

4 de septiembre de 2015

UNIVERSIDAD DE BUENOS AIRES

Resumen

Grupo de redes complejas y comunicación de datos (CoNexDat)
Facultad de Ingeniería

Mágister en Ingeniería en Telecomunicaciones

Evaluación del algoritmo Contact Graph Routing en el caso de constelaciones satelitales LEO.

por Ing. Juan Pablo Astudillo León

El presente trabajo estudia el algoritmo Contact Graph Routing (*CGR*) en el caso de constelaciones satelitales *LEO*. Al principio se analiza el panorama de las comunicaciones por satelite en redes basadas en *TCP/IP*, y posteriormente su solución la arquitectura *DTN*. Más adelante, se estudia las variables y consideraciones que utiliza *CGR* para el cálculo de las rutas de los paquetes. Finalmente, se plantean ambientes de simulación con el software *OMNet++*, para evaluar el rendimiento del algoritmo ante modificaciones del plan de contactos y el tráfico configurado ...

Agradecimientos

El desarrollo de este proyecto no habría sido posible sin el apoyo y el estímulo del Dr. Ing. José Ignacio Alvarez-Hamelin, bajo cuya supervisión fue posible el desarrollo del proyecto.

De igual manera doy gracias al apoyo brindado por el Dr. Ing. Jorge M. Finochietto, Ing. Juan Fraire y Ing. Pablo Madoery, que en reiteradas ocasionadas nos ayudaron a encaminar el proyecto en una forma adecuada.

Asimismo quiero reconocer el apoyo brindado por la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (Senescyt), institución que financió mediante una beca mis años de estudios e investigación en la Universidad de Buenos Aires.

También quiero agradecer a los miembros del jurado que aceptaron revisar el resultado del presente trabajo.

No puedo terminar sin agradecer a mi familia, en cuyo estímulo constante y amor he confiado a lo largo de mis años estudiantiles...

Índice general

Índice de figuras

Índice de cuadros

Abreviaturas

ARQ	**A**utomatic **R**epeat re-**Q**uest
BP	**B**undle **P**rotocol
CGR	**C**ontact **G**raph **R**outing
CLA	**C**onvergence **L**ayers **A**dapters
CRP	**C**ontact **R**eview **P**rocedure
DTN	**D**elay **T**olerant **N**etwork
ECC	**E**stimated **C**apacity **C**onsumption
FBP	**F**orward **B**undle **P**rotocol
GEO	**G**eosynchronous **S**atellite **O**rbit
GUI	**G**raphical **U**ser **I**nterface
IP	**I**nternet **P**rotocol
IPN	**I**nterplanetary **N**etwork
LEO	**L**ow **E**arth **O**rbit
LTP	**L**icklider **T**ransmission **P**rotocol
PEPs	**P**erformance **E**nhancing **P**roxies
QoS	**Q**uality **o**f **S**ervice
RTT	**R**ound **T**rip **T**ime
TCP	**T**ransmission **C**ontrol **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
UTC	**C**oordinated **U**niversal **T**ime
TTL	**T**ime **T**o **L**ive

Capítulo 1

Introducción

1.1. Introducción

Las comunicaciones en redes satelitales deben superar ciertos obstáculos que no son afrontados por redes basadas en *IP*. La movilidad de los nodos, escasa conectividad, recursos restringidos, y retardos significativos de propagación representan importantes desafíos para el enrutamiento. Las estaciones satelitales se comunican entre ellas a través de frecuencias de radio que son propagadas aproximadamente a la velocidad de la luz [1]. Por otra parte, los retardos de propagación entre dos nodos pueden llegar a ser mayores al tiempo que estos dos tienen línea de vista directa (ver figura 1.1), lo cual incrementa aún más el problema de comunicación. No obstante, esto se solucionaría si dicho nodo supiera la posición futura del nodo al cual él quiere transmitir.

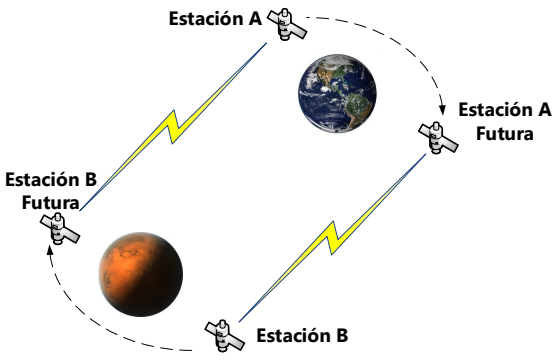

FIGURA 1.1: Si la estación satelital A tiene que transmitir a una estación B sobre la base de que hay una línea de vista existente, por el tiempo que la señal atraviesa la distancia entre ellos, la estación espacial A estará en un lugar diferente.

Las comunicaciones satelitales se consideran como redes tolerantes a demoras [2] o *DTN* (Delay Tolerant Network, sus siglas en inglés), por ello el reenvío de paquetes a través de este tipo de redes difiere en varios aspectos de la transmisión de paquetes a través de una red basada en *IP*. En una red basada en *IP* la conectividad entre nodos se basa en el intercambio de paquetes continuamente en toda la red. Las fallas en la conectividad son anómalas y pueden ser interpretados como cambios en la topología de la red. Por otra parte, las latencias en la transmisión de cada uno de los paquetes son muy pequeñas. Este hecho, junto con la conectividad continua, asegura que la velocidad a la cual la información que contiene cambios sobre la conectividad a través de la red supere la velocidad a la cual se producen dichos cambios.

Sin embargo, en una red *DTN* esto no se cumple, ya que no hay ninguna expectativa de conectividad continua a lo largo de la red. Las fallas en la conectividad pueden ser rutinarias, de muy larga duración, y recurrentes; que no deben ser interpretadas como cambios en la topología. Además, las latencias de propagación de la señal pueden ser grandes, lo cual significa que la velocidad a la que la información propagada sobre los cambios en la conectividad a través de la red puede ser mucho más baja que la velocidad a la que se producen los cambios [2].

Contact Graph Routing (*CGR*) [3], está diseñado para el uso en las redes donde se planifican los cambios en la conectividad de la red. Es decir, el algoritmo se basa en la capacidad de realizar un precálculo de la topología de la red. Donde se asume que los cambios en la topología se producen en menor frecuencia que el tiempo para sincronizar las configuraciones. Por lo que ante cambios programados, un plan de contactos global de todos estos eventos puede ser distribuido de antemano a todos los nodos, permitiendo a cada nodo tener un conocimiento teóricamente exacto de la conectividad en la red.

1.2. Organización de la tesis

El presente trabajo está organizado en cuatro capítulos: Estado del arte (Capítulo 2), Simulación de la red satelital (Capítulo 3), Escenarios estudiados y análisis de los resultados (Capítulo 4) y Conclusiones (Capítulo 5).

El capítulo 2 introduce el marco teórico del trabajo. Al principio se realiza una breve descripción sobre las comunicaciones por satélite considerando los inconvenientes que

involucran al trabajar con un esquema tradicional *TCP/IP* en un entorno espacial. Posteriormente, se presenta *DTN* como una arquitectura pensada para redes con entornos desafiantes, como por ejemplo el espacio, en donde se discute acerca del protocolo *Bundle* y esquemas de enrutamiento en *DTN*. Finalmente, es explicado el algoritmo Contact Graph Routing (*CGR*) haciendo hincapié sobre los pasos de procesamiento, variables y algoritmos que emplea *CGR* para el cálculo de las rutas.

En el capítulo 3 se explica el ambiente de trabajo de *OMNeT++* y las pautas consideradas para generar un modelo de red. Adicionalmente, se detalla la importancia de utilizar *C++* para dar funcionalidad a cada uno de los nodos satelitales y las herramientas que ofrece *OMNeT++* para analizar los datos generados. Después, se procederá a explicar las variables empleadas para generar el tráfico entre los nodos, el plan de contactos y el análisis de los datos recolectados mediante *Python*.

Descritas las herramientas y plataformas utilizadas, se pasa en el capítulo 4 a realizar las simulaciones para evaluar el desempeño de *CGR* donde se explican las variables configuradas en cada uno de los escenarios con el objetivo de evaluar el rendimiento de este algoritmo ante cambios en el tráfico y el plan de contactos.

Por último, se culmina el presente trabajo con las Conclusiones (Capítulo 5), además de un apartado que incluye los algoritmos programados (Anexo A) y la Bibliografía.

Capítulo 2

Estado del Arte

El presente capítulo introduce el marco teórico del trabajo. Al principio se realiza una breve descripción sobre las comunicaciones por satélite con énfasis en los tipos de órbita y los problemas que involucran al mover un esquema tradicional *TCP/IP* en un entorno espacial. Más adelante se presenta *DTN* como una arquitectura genérica desarrollada para acoplar redes tradicionales con redes que trabajan sobre entornos desafiantes como el espacio profundo. Posteriormente se menciona el protocolo Bundle *(BP)* como un protocolo superpuesto a la capa de transporte, y además se nombran algunos algoritmos de enrutamiento desarrollados para *DTN*. Finalmente, es explicado el algoritmo Contact Graph Routing (*CGR*) haciendo hincapié sobre las pasos de procesamiento que emplea, desde la construcción del plan de contactos hasta los algoritmos que componen *CGR* encargados de encontrar una ruta de un paquete a destino.

2.1. Comunicaciones satelitales

2.1.1. Introducción

Los satélites artificiales nacieron durante la guerra fría, entre los Estados Unidos y la Unión Soviética, que pretendían ambos llegar a la luna y a su vez lanzar un satélite a la órbita espacial. El primer satélite artificial colocado en órbita fue el Sputnik 1, lanzado el 4 de octubre de 1957 por la Unión Soviética. Mientras que el Telstar fue el primer satélite de telecomunicaciones comercial del mundo, y este fue puesto en órbita terrestre por los Estados Unidos el 10 de julio en 1962 [4].

Los satélites presentan muchas ventajas, dado que son dispositivos adecuados para trasmitir información en zonas poco desarrolladas [5]. Los satélites son clasificados en función de la misión y el tipo de órbita.

Los satélites clasificados por misión realizan tareas en específico, como por ejemplo los satélites de comunicaciones, navegación, meteorológicos, reconocimiento, y entre otros. Sin embargo, para el presente trabajo la clasificación por el tipo de órbita requiere un mayor estudio [4].

2.1.2. Tipos de satélites por órbita

El período orbital del satélite esta en función de su distancia con respecto a la Tierra, es decir, cuanto más cerca este se encuentre, más reducido es su período. En la figura 2.1 se indica un diagrama de las diferentes regiones de altura que definen el tipo de órbita.

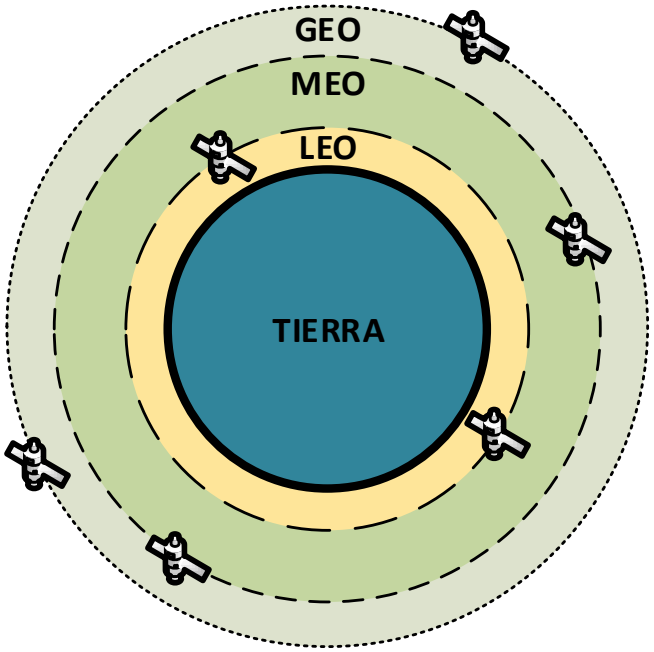

FIGURA 2.1: Diagrama de las diferentes regiones de altura que definen el tipo de órbita. **Órbita LEO:** 160 a 2.000 km de altitud. **Órbita MEO:** 10.075 a 20.150 Km y **Órbita GEO:** 35.786 km sobre el Ecuador.

Cabe mencionar que cuando el período orbital del satélite no coincide con el de la Tierra produce que éstos tengan un movimiento en el cielo, lo que dificulta el apuntamiento de las antenas desde la Tierra, y por lo tanto la comunicación sea interrumpida por períodos de tiempo [6].

2.1.2.1. Satélites geoestacionarios (GEO)

La altitud de 35.786,04 kilómetros permite que el período orbital del satélite coincida exactamente con el de la rotación de la Tierra. La órbita correspondiente se conoce como el cinturón de Clarke[1]; por lo tanto, los satélites que giran en esta órbita parecen inmóviles o estacionarios en el cielo para un observador ubicado en la Tierra, por lo que se los llama satélites geoestacionarios [7]. Esta característica permite el uso de antenas fijas, debido a que su apuntamiento no cambia y garantiza el permanente contacto con el satélite.

Los satélites comerciales emplean tres bandas de frecuencias, llamadas *C*, *Ku* y *Ka* (ver tabla 2.1). Cuando se trabaja en una misma banda de frecuencia no es recomendable colocar muy contiguos a los satélites dado que puede producir interferencias entre ellos [8].

Banda	Frecuencia ascendente (Ghz)	Frecuencia descendente (Ghz)
C	5,925 - 6,425	3,7 - 4,2
Ku	14,0 - 14,5	11,7 - 12,2
Ka	27,5 - 30,5	17,7 - 21,7

Tabla 2.1: Bandas de frecuencias utilizadas por satélites comerciales.

2.1.2.2. Satélites de órbita baja (LEO)

Los satélites *LEO* están localizados a una distancia reducida con relación a la Tierra (160-2.000 km). Por otra parte, ofrecen la ventaja de reducir la pérdida por propagación y el retardo, debido a un camino más corto para la propagación de las señales.

Para un observador en la Tierra estos satélites no están fijos en el cielo pero parecen moverse muy rápidamente (con un período orbital típico de alrededor de 90-100 minutos), esto representa una ventaja para la observación de la Tierra, dado que toda la Tierra

[1] Ver `http://www.itu.int/itunews/manager/display.asp?lang=es&year=2008&issue=03&ipage=Arthur-Clarke&ext=html`, accedido el 8/7/2015.

puede ser explorada por un solo satélite, pero requiere constelaciones de decenas de satélites para una cobertura global continua, por ejemplo para proporcionar servicio de telefonía o acceso a Internet [9].

2.1.3. Desafíos en las comunicaciones por satélite

Las comunicaciones por satélite presentan algunas de las características distintivas que merecen ser analizadas brevemente. En el lado positivo, ofrecen una forma muy efectiva para proporcionar rápidamente cobertura en grandes áreas. Los satélites pueden ofrecer acceso ubicuo a Internet a un costo razonable en los países en desarrollo y especialmente en zonas poco pobladas. Por otro lado, las comunicaciones por satélite son esenciales para los equipos de rescate en caso de desastres naturales, como terremotos o inundaciones, cuando la infraestructura de comunicación terrestre se ve afectada seriamente [10].

No obstante, este tipo de comunicaciones tienen que hacer frente a una serie de diferentes desafíos en cada una de las capas de red. En particular, si nos centramos en la capa de transporte y las capas superiores, el rendimiento se ve desafiado por las siguientes peculiaridades [11]:

i. Largos tiempos de ida y vuelta (RTT), especialmente para los sistemas GEO (alrededor de 600 ms).

ii. Posibles interrupciones del canal, especialmente para terminales móviles, debido a las obstrucciones como edificios, túneles, etc.

iii. Otros problemas surgen desde el entorno espacial, como la distancia de separación entre dos estaciones satelitales que en algunos de los casos puede llegar a ser muy grande (ver figura 1.1).

Una posible solución a estos problemas proviene de la modificación de la capa de transporte, es decir, el uso de un protocolo de transporte especializado en ambos nodos finales [10]. Con el fin de aplicar variantes de protocolo de transporte adecuado para el enlace por satélite se utilizan en algunos casos los *PEPs*[2] (Performance-enhancing proxies) [12], [13]. Los *PEPs* son agentes de red diseñados para mejorar el rendimiento extremo

[2]Ver `https://tools.ietf.org/html/rfc3135`, accedido el 9/5/2015.

a extremo de algunos protocolos de comunicaciones (ver figura 2.2). Existen diferentes tipos de *PEPs* y cada uno es usado para resolver un problema en especifico.

Para mejorar el rendimiento en enlaces satelitales donde existen tiempos muy altos de *RTT* es empleado *Split-TCP* [14],[15].

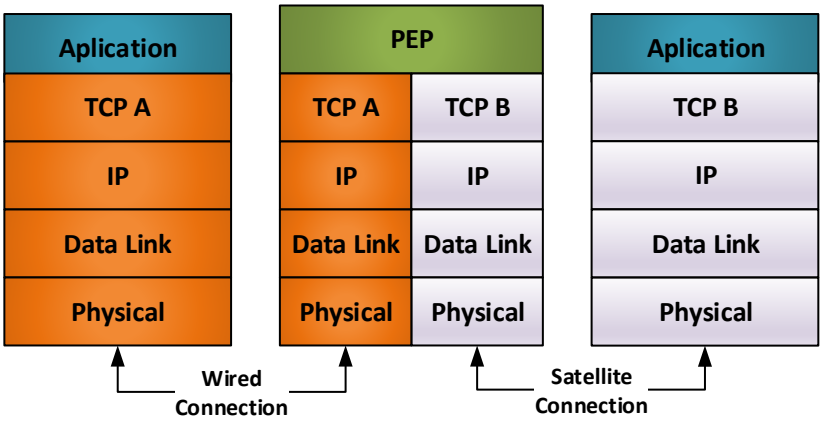

FIGURA 2.2: Pila de protocolos para un esquema de red que utiliza PEPs. Para este caso existen dos conexiones para la capa de transporte, la primera una conexión cableada y la segunda un enlace satelital. PEPs es transparente para los nodos finales, pero no en los nodos intermedios que tienen que trabajar en conjunto con la capa de transporte y aplicación [10].

El método *Split-TCP* divide la conexión de extremo a extremo *TCP* en múltiples conexiones y utiliza diferentes parámetros para transferir los datos a través de los diferentes segmentos [14]. Las aplicaciones localizadas en los extremos utilizan el protocolo *TCP* estándar sin ninguna modificación, y no necesitan saber de la existencia de *PEPs* localizados en el medio. De esta manera permiten aislar las características de enlace de satélite desde el resto de la red [16]. Sin embargo, tienen algunos inconvenientes graves, relacionados con la seguridad, ya que la división de *TCP* es incompatible con mecanismos estándar de seguridad en Internet, tales como *IPsec*[3] [17], [18], que cifra las cabeceras *TCP* que el *PEP* debe leer para mejorar el rendimiento en el enlace por satélite [10].

En el siguiente apartado se introduce *DTN* como una arquitectura para solucionar los inconvenientes que presentan las comunicaciones satelitales.

[3]IPsec, es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet *(IP)* autenticando y/o cifrando cada paquete *IP* en un flujo de datos.

2.2. Redes tolerantes a demoras (DTNs)

El origen del concepto *DTN* (Delay Tolerant Networks [19], [20]) se origina por la necesidad de crear redes interplanetarias (*IPN*, Interplanetary Networks), en particular, para las misiones en el planeta Marte donde la nave espacial en órbita tendría la capacidad de actuar como un nodo repetidor para la retransmisión de datos a los módulos de aterrizaje. En tales situaciones, uno se enfrenta a latencias en el orden de las decenas de minutos, así como el ancho de banda limitado y altamente asimétrico [21].

2.2.1. Diferencias entre redes DTN y TCP/IP

El reenvío de paquetes a través de una red basada en *DTN* difiere en varios aspectos de la transmisión de paquetes a través de una red basada en *IP*. Básicamente, en el enrutamiento *IP* cada router puede basarse en una comprensión local de la conectividad actual en la red. Es decir, la ruta a un host destino dado una vez calculada, puede ser almacenada en una tabla de enrutamiento para referencia futura y sólo tendrá que ser cambiado a la llegada de nueva información de conectividad. Esta es transmitida mediante mensajes propios del protocolo que son generados inmediatamente en respuesta a cambios detectados en la conectividad que invalida esa ruta. Por otra parte, el enrutamiento en *DTN* no goza de tales ventajas. El potencial retraso en la llegada de la información relativa a los cambios de conectividad hace que toda esta información sea potencialmente obsoleta; un nodo que se basó únicamente en este flujo de información nunca puede tener una comprensión totalmente exacta de la conectividad actual en la red [3].

2.2.1.1. Protocolo Bundle

El protocolo Bundle *(BP)* [22] ha sido diseñado como una implementación de la arquitectura *DTN*, en donde la unidad básica de datos es un *bundle*[4]. Con el objetivo de soportar las comunicaciones en entornos desafiantes, *BP* se basa en la inserción en los puntos finales y algunos nodos intermedios de una nueva capa, que se encuentra entre la capa de aplicación y las capas inferiores llamada *Bundle Layer* [20] como se muestra en

[4]Bundle, es un mensaje que lleva unidades de datos de los protocolos de capa de aplicación.

la figura 2.3. El *BP* puede interactuar con capas inferiores diferentes a través de *CLA* (Convergence Layers Adapters)[5].

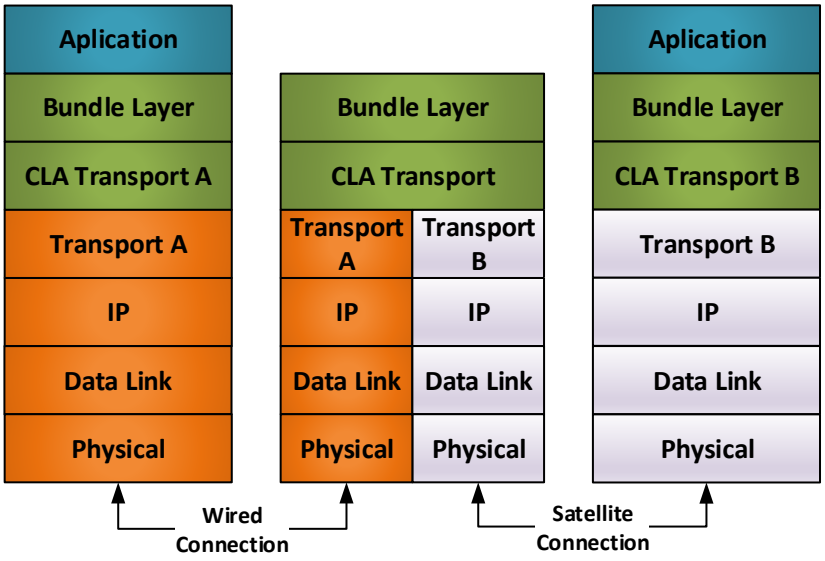

FIGURA 2.3: Arquitectura *DTN*. *DTN* no es transparente a la aplicación debido a que es instalado en los nodos de los extremos a diferencia del esquema empleado por *PEPs*. Sin embargo, *DTN* no opera en las capas de trasporte y aplicación en los nodos intermedios debido a la inserción de *BP*. Finalmente, el uso de diferentes *CLAs* permite utilizar variantes de protocolos de transporte a diferencia de los *PEPs* [10].

En particular, la arquitectura *DTN* es adecuada para actuar como superposición en la parte superior de una red heterogénea que consiste en diferentes segmentos, como las redes inalámbricas de sensores / ad hoc, Internet por cable, redes de área local inalámbricas (LAN) o enlaces satelitales mediante *CLAs*[6]. Por otro lado las características de protocolo de transporte extremo a extremo se limitan a un salto de *DTN*, mientras que la transferencia de datos de extremo a extremo a través de la red heterogénea es proporcionada por el *Bundle Layer* [27].

2.2.2. Enrutamiento en DTN

El objetivo en los esquemas de enrutamiento tradicionales es seleccionar una ruta que minimice alguna métrica sencilla como el número de saltos. Por otra parte, para redes

[5]Un *CLA* envía y recibe paquetes utilizando los servicios de algún enlace, red o protocolo de internet nativo.

[6]Cabe mencionar que varios *CLAs* se han definido, incluso para *TCP* [23], *UDP* [24], *LTP* [25], [26] y *CLAs* adicionales incluyen Bluetooth y Ethernet.

DTN, el objetivo es aumentar la probabilidad de transmitir un paquete a destino, pero reducir los retardos de entrega es igualmente importante para este tipo de redes [28]. Algunos autores han realizado diferentes clasificaciones de algoritmos de enrutamiento para *DTN*. Por ejemplo, Jain *et al.* [28] clasifica los algoritmos de enrutamiento en función de la cantidad de datos que utilizan éstos para calcular las rutas, es decir, información sobre los contactos, estado de la cola, y la demanda de tráfico.

Otra clasificación proviene de Balasubramanian *et al.* [29] y Demmer and Fall *et al.* [30], donde los algoritmos de enrutamiento se clasifican en dos esquemas: reenvío y replicación. Por un lado, los esquemas de reenvío son adecuados para la optimización del almacenamiento, ancho de banda y energía, ya que envían solo una copia del paquete al nodo vecino. Por otro lado, los esquemas de replicación proporcionan un mayor rendimiento en términos de probabilidad de entregar un paquete debido a que replican múltiples copias a los vecinos con la esperanza de que uno de ellos alcance al destino.

2.2.2.1. Algoritmos de enrutamiento en DTN

Caini *et al.* [10] en su trabajo realiza una descripción de algunos algoritmos de enrutamiento que son empleados en *DTN*:

i. **Epidemic Routing** [31]: los nodos replican de forma continua y transmiten paquetes a nuevos contactos que aún no poseen una copia de éste. Aunque es altamente confiable, este algoritmo consume muchos recursos, ya que no hace ningún intento por eliminar réplicas que no mejoran la probabilidad de la entrega del paquete.

ii. **ProPHET** [32], [33]: utiliza la no aleatoriedad de contactos, para replicar paquetes sólo si hay una probabilidad dada para entregarlo.

iii. **Spray-and-wait** [34]: que envía un número limitado de copias en la red, y luego espera hasta que uno de estos nodos cumpla con el contacto de destino.

iv. **MaxProp** [35]: se basa en priorizar tanto la programación de los paquetes transmitidos a los nodos *DTN* y la programación de los paquetes que se cayeron. Las prioridades se basan en probabilidades de ruta de acuerdo a los datos históricos y mecanismos complementarios.

v. RAPID [36], [29]: utiliza una variable aleatoria que representa el contacto entre dos nodos *DTN* y replica paquetes en función de esta.

La aleatoriedad que presentan estos algoritmos, no está presente en redes satelitales donde los contactos entre nodos se programa con anticipación [37].

2.3. Algoritmo Contact Graph Routing (CGR)

El algoritmo *CGR* permite el cálculo de rutas en redes *DTN* entre un nodo fuente y destino, donde se asume que la conectividad entre nodos es programada e intermitente [38]. La figura 2.4 muestra los tres pasos de procesamiento que emplea *CGR* para el enrutamiento de paquetes.

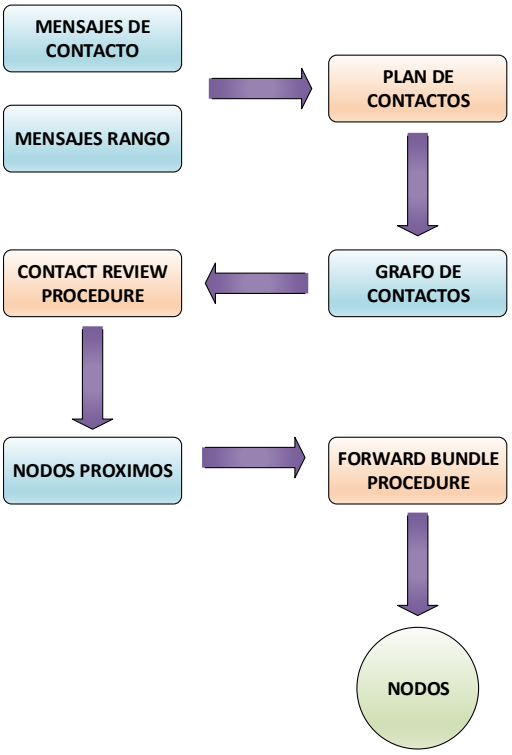

FIGURA 2.4: El algoritmo *CGR* tiene tres pasos de procesamiento [39]. **1.** Plan de Contactos, **2.** Algoritmo *Contact Review Procedure* y **3.** Algoritmo *Forward Bundle Procedure*.

i. Crea un plan y un grafo de contactos en función de los mensajes de contacto y rango.

ii. *CGR* ejecuta recursivamente el algoritmo *CGR-CRP* (Contact Review Procedure), el cual genera un listado de nodos próximos de un salto capaces de entregar el paquete a destino a partir de la información del gráfico de contactos.

iii. El algoritmo *CGR-FBP* (Forward Bundle Procedure) selecciona un nodo del listado de nodos próximos en función de las métricas para el envío de los paquetes.

2.3.1. Plan de Contactos

Cuando se programan cambios en la conectividad, un *plan de contactos* global de todos estos eventos puede ser distribuido de antemano a todos los nodos, permitiendo a cada nodo tener un conocimiento teóricamente exacto de la conectividad en la red en cualquier momento especificado.

Los mensajes del plan de contactos son de dos tipos: mensajes de contacto y mensajes de rango. Estos mensajes contienen el tiempo de inicio y finalización del contacto, el número del nodo transmisor y receptor, y la tasa de transmisión de datos programada entre los dos nodos durante este intervalo.

2.3.2. Grafo de contactos

Un nodo puede construir un grafo de contactos a partir de un plan de contactos completo. Este grafo construido localmente en un nodo dado contiene, por cada nodo en la red [3]:

i. Una lista de objetos llamadas *xmit* que encapsulan el tiempo de inicio y finalización del contacto, el número del nodo de transmisión y la velocidad de transmisión, derivados todos de los mensajes de contacto. Esta lista es ordenada por el tiempo de finalización.

ii. Una lista de objetos llamadas *origin* que encapsulan el número del nodo de transmisión y la distancia actual de ese nodo desde el receptor.

Es importante recalcar que esta información debe ser actualizada a medida que pasa el tiempo, es decir, los objetos *xmit* y *origin* cuyos tiempos de finalización se han alcanzado deben ser eliminados del grafo.

2.3.3. Características de enrutamiento de CGR

Para el enrutamiento, *CGR* asume que todos los nodos tienen la información global de todos los contactos (duración y volúmenes). Por ende, cada nodo utiliza la información de contacto en su poder para calcular localmente las rutas del paquete. Es importante mencionar que por cada paquete *CGR* calcula la ruta completa hacia destino y selecciona el mejor camino. Sin embargo, esto da lugar a la selección del mejor nodo próximo al que se enviará el paquete como se observa en la figura 2.5. De hecho, el nodo próximo seleccionado recalcula nuevamente la ruta cuando el paquete arriba [40]. Cabe recalcar que este algoritmo no almacena las rutas ya calculadas para cálculos posteriores.

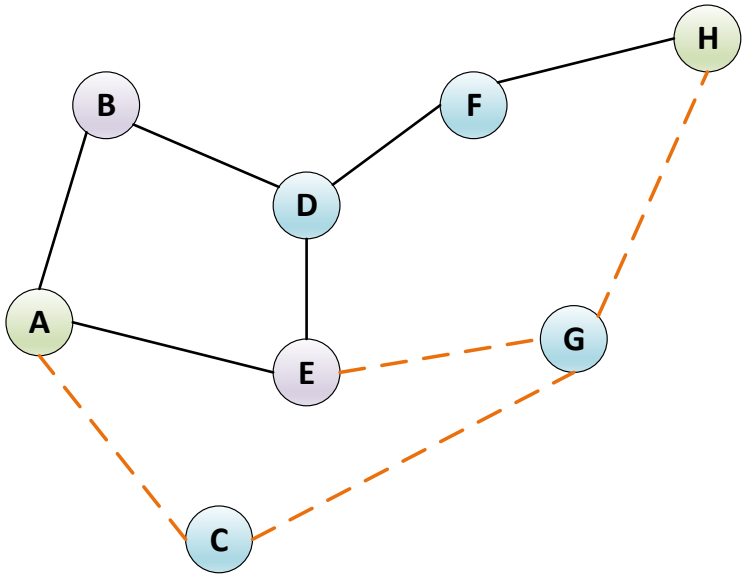

FIGURA 2.5: El algoritmo *CGR* produce un listado de nodos vecinos de un salto. Por ejemplo, si el nodo *A* desea enviar un paquete al nodo *H*, *CGR* produce un conjunto de nodos vecinos $\{B, E\}$, de este conjunto *CGR* selecciona el nodo que tiene la menor métrica para transmitir el paquete. Las lineas entrecortadas representan enlaces que no están disponibles por un período de tiempo [39].

2.3.4. Terminología y conceptos importantes

2.3.4.1. Ventanas de contacto y volúmenes de contacto

Los enlaces entre nodos DTN están activos sólo durante intervalos de tiempo limitados conocidos como ventanas de contacto. Durante cada ventana sólo una cantidad limitada de datos se puede transferir. El valor máximo es llamado volumen de contacto, el cual

es el producto de la velocidad del enlace (bps) y la ventana de contactos (segundos). Las ventanas de contacto y los volúmenes de contacto se conocen con anterioridad, porque están estrechamente relacionado con el movimiento de los satélites o la asignación de ancho de banda programada de los enlaces en el espacio [40].

2.3.4.2. Último Momento y tiempo de expiración

El "último momento.[es] el tiempo límite para enviar un paquete durante un contacto dado. Por otro lado, el "tiempo de expiración.[es] el período de tiempo después del cual el paquete está sujeto a la destrucción si no se ha entregado a su destino. La fecha de expiración de un paquete se calcula como la hora de creación más su tiempo de vida (Time-to-live, *TTL*) [3].

2.3.4.3. Consumo estimado de capacidad y capacidad residual

El tamaño de un paquete es la suma de los tamaños de todos los bloques incluyendo el bloque de datos. El consumo estimado de capacidad (*ECC*, Estimated Capacity Consumption) para un paquete puede ser calculado como la suma del tamaño del paquete y un estimado del tamaño de las cabeceras de las capas superiores. Por otra parte, la capacidad residual de un contacto dado entre el nodo local y uno de sus vecinos, es la suma de las capacidades de ese contacto y todos los contactos programados anteriormente entre el nodo local y ese vecino, menos la suma de los *ECC* de todos los paquetes que están actualmente en la cola para la transmisión a ese vecino [3].

2.3.5. Algoritmos de CGR

El algoritmo *CGR* se descompone a su vez en dos algoritmos: *Contact Review Procedure.* y *Forward Bundle Procedure*. La tabla 2.2 identifica las variables y valores iniciales utilizados por ambos algoritmos, los cuales están en función del plan de contactos y el tráfico configurado. Dado que *CGR* es ejecutado de forma recursiva implica que algunas variables se han configurado con valores mínimos y máximos al inicio del programa tales como: la variable de destino actual, la distancia de red o número de saltos, listado de nodos vecinos y excluidos.

Variable	Valor Inicial	Descripción
B		Paquete a ser enviado.
$\mathbf{B_{Dest}}$		Destino del paquete.
X		Tiempo de expiración del paquete.
D	B_{Dest}	El destino actual.
$\mathbf{D_{Transmits}}$	{ }	Contactos cuyo nodo receptor es D.
$\mathbf{D_{PDT}}$	∞	El tiempo mas próximo en el cual B puede alcanzar a D.
$\mathbf{D_{PND}}$		El menor número de saltos para el cual B puede llegar a D desde N.
ExclNodes	{}	Conjunto de nodos que no deben ser considerados para el envio de un paquete dado.
m		Una entrada de contacto del gráfico de contactos.
$\mathbf{m_{StartTime}}$		El tiempo de inicio del contacto.
$\mathbf{m_{EndTime}}$		El tiempo de finalización del contacto.
$\mathbf{m_{TransmitNode}}$		El nodo transmisor sobre el contacto.
$\mathbf{m_{TransmitRate}}$		La tasa de transmisión sobre el contacto.
$\mathbf{m_{ResCap}}$		La capacidad residual sobre el contacto.
N		El nodo local que estará enviando el paquete.
ND	0	El número de salto anticipados cuando se envía el paquete.
ProxNodes	{}	Conjunto de nodos próximos al nodo local que son capaces de enviar el paquete.
$\mathbf{D_{PDT}}$	0	El tiempo de entrega más próximo previsto.
T	0	El ultimo tiempo que B puede ser enviado sobre m.

Tabla 2.2: Definición de las variables CGR y sus valores de inicialización [39].

Cabe destacar que la lista de nodos excluidos son nodos a los cuales no se calculará una ruta para enviar el paquete, esta lista es inicialmente configurada con el nodo desde el que se recibió el paquete, de esta forma se evita un bucle entre este nodo y el nodo de análisis.

2.3.5.1. Algoritmo Contact Review Procedure (CGR-CRP)

El algoritmo *CGR-CRP* toma como valores de entrada: el nodo de destino, el paquete, la lista de nodos excluidos y el tiempo de vida del paquete (ver algoritmo 1). De estos valores se genera un conjunto de nodos vecinos de un salto capaces de entregar el paquete desde el nodo de análisis hacia el destino [3].

Algoritmo 1 Algoritmo CGR-CRP [39]

Input: Nodo destino D, bundle B, tiempo de vida X y el listado de nodos excluidos ExclNodes.

Output: Construye un listado de nodos vecinos *ProxNodes* con distancia de un salto.

```
1: Store(PDT, ND)
2: ExclNodes = ExclNodes ∪ {D}
3: foreach m ∈ D_Transmits
4:    T = Last_Moment(m, B, X)
5:    if current_time ≤ T and m_StartTime ≤ T then
6:       if D = B_Dest then
7:          PDT = m_EndTime
8:       end if
9:       S = m_TransmitNode
10:      if S==N then
11:         if m_ResCap ≥ ECC(B, X) then
12:            if D ∈ ProxNodes then
13:               if PDT ≤ D_PDT then
14:                  D_PDT = PDT
15:                  D_PND =_MIN [D_PND, ND]
16:               else
17:                  ND = D_PND
18:                  PDT = D_PDT
19:                  ProxNodes = ProxNodes ∪ {D}
20:               end if
21:            else if S ∉ ExclNodes then
22:               PDT =_MIN [PDT, m_EndTime]
23:               L = m_EndTime − Latency(B, m)
24:               S_ND = D_PND + 1
25:               ND = ND + 1
26:               CGR − CRP(S,_MIN [T, L], B, ExclNodes)
27:            end if
28:         end if
29:      end if
30:   end if
31: end for
32: ExclNodes = ExclNodes − {D}
```

De este algoritmo podemos resaltar lo siguiente:

i. El nodo D se anexa a la lista de nodos excluidos con el objetivo de evitar bucles de enrutamiento.

ii. Para el cálculo se toma en cuenta todos los contactos cuyo nodo receptor es el nodo D y de éste se calcula el último momento para enviar un paquete durante un contacto dado, tal que llegará al nodo receptor antes del tiempo de expiración.

iii. Si la capacidad residual de un contacto es mayor o igual al *ECC* (consumo de capacidad estimado para un paquete) se realiza el cálculo para encontrar el conjunto de nodos vecinos capaces de enviar el paquete considerando los tiempos y la distancia de red.

iv. Se calcula la latencia, este valor se utiliza para permitir que durante un tiempo el nodo reciba el paquete desde su origen, ponerlo en cola para su transmisión, y para después reenviarlo.

v. Las últimas líneas ejecutan nuevamente el algoritmo *CGR-CRP* con el objetivo de buscar más nodos que puedan enviar el paquete de análisis.

2.3.5.2. Algoritmo Forward Bundle Procedure (CGR-FBP)

En este punto se asume que cada nodo de una lista de nodos próximos es un nodo vecino al que se puede enviar el paquete, con la esperanza de que uno de los contactos previstos de ese nodo permita el transporte del paquete en una ruta aceptable hacia su destino final [3]. El seudocódigo de *CGR-FBP* es descrito en el algoritmo 2, y de este podemos destacar:

i. Si la lista de nodos próximos está vacío, esto significa que *CGR* no fue capaz de calcular una ruta que pueda transmitir este bundle a su nodo destino final antes del tiempo de expiración.

ii. Se selecciona un único nodo próximo para la transmisión. En teoría se selecciona la ruta empezando por el nodo próximo que tiene asociado el tiempo de entrega previsto como más próximo. En caso de que ocurra un empate entre varios nodos próximos, se selecciona el nodo que tiene la distancia de red más pequeña. En el caso de que múltiples nodos próximos aún estén empatados se elige arbitrariamente el que tiene el número de nodo más pequeño [3].

Algoritmo 2 Algoritmo CGR-FBP [39]

Input: Bundle a enviar B

Output: Selecciona uno de los nodos próximos ProxNodes basados en métricas.

```
1: CGR − CRP(B_Dest, B_EspirationTime, B, ExclNodes)
2: if ProxNodes ≠ {} then
3:    if B_Priority==Critical then
4:       foreach N ∈ ProxNodes
5:          Transmitir B a N
6:       end for
7:    else
8:       N = POP(SORT_NODES(ProxNodes))
9:       Transmitir B a N
10:   end if
11: else
12:   No se encontró ruta
13: end if
```

Capítulo 3

Simulación de la red satelital

La presente sección detalla las herramientas utilizadas para la implementación de Contact Graph Routing y el análisis de los datos generados en las simulaciones. Inicialmente se discute brevemente sobre el área de aplicación de *OMNeT++* y las partes que contiene un modelo de red generado en este simulador. Más adelante, se expone la jerarquía modular de *OMNeT++* y las clases principales empleadas para dar funcionalidad a los nodos empleando la utilización de mensajes, paquetes y señales que viajan a través de la red.

A continuación, se mencionan las consideraciones empleadas para generar los planes de contacto mediante el programa *Python* en base a los tiempos de conexión entre los satélites y los períodos de descarga de los archivos en alguna estación terrena. Posteriormente, se especifica cada uno de los parámetros empleados para generar el tráfico en cada uno de los nodos, en función de los volúmenes de contacto, almacenamiento de los nodos, y el tamaño y tiempo de generación de las imágenes. En el siguiente apartado, se describe la implementación de algoritmo *CGR* tomando en cuenta las clases que brinda *OMNeT++* para generar la funcionalidad de cada uno de los nodos y los mensajes de intercambio entre ellos.

Finalmente, es explicado el código programado en *Python* para separar los datos recolectados y las funciones escritas para el análisis de los resultados de las simulaciones.

3.1. Simulador OMNeT++

OMNeT++ es una herramienta pública de simulación de eventos discretos de redes basado en componentes modulares, orientado a objetos y con un fuerte soporte de *GUI*[1] [41]. Su área de aplicación principal es la simulación de redes de comunicaciones, pero debido a su arquitectura genérica y flexible, se utiliza con éxito en otras áreas como la simulación de sistemas complejos de *TI*, redes fijas o inalámbricas, y arquitecturas de hardware [42], [43].

Detallaremos a continuación la arquitectura de OMNeT++.

3.1.1. Módulos jerárquicos

Un modelo de simulación en *OMNeT++* está compuesto por un conjunto de módulos que se comunican entre ellos a través de mensajes (ver figura 3.1). Los módulos activos en la simulación son los módulos simples y son implementados con *C++*. Por otra parte, los módulos simples pueden ser combinados en módulos compuestos, lo que permite al usuario crear modelos de red más complejos utilizando instancias previamente definidas en el módulo simple de red, ya que cuando un módulo es usado como un bloque compuesto, no existe distinción entre éste y los módulos simples, lo que brinda transparencia en la comunicación durante la simulación [44].

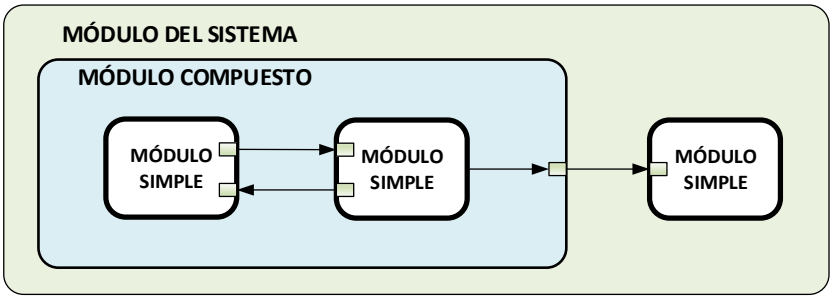

FIGURA 3.1: Un modelo de red en *OMNeT++* puede ser representado con la combinación de módulos simples y compuestos

[1]Esta herramienta de simulación está disponible tanto para sistemas operativos basados en UNIX como para Windows. Fue creado por András Varga en el año 2001 en la Universidad Técnica de Budapest. Su versión comercial, OMNEST ha sido desarrollada por Simulcraft Inc.

3.1.2. Partes que conforman un modelo de red en OMNeT++

La figura 3.2 indica el proceso de construcción de un modelo en *OMNeT++*, y éste se compone de las siguientes partes [44]:

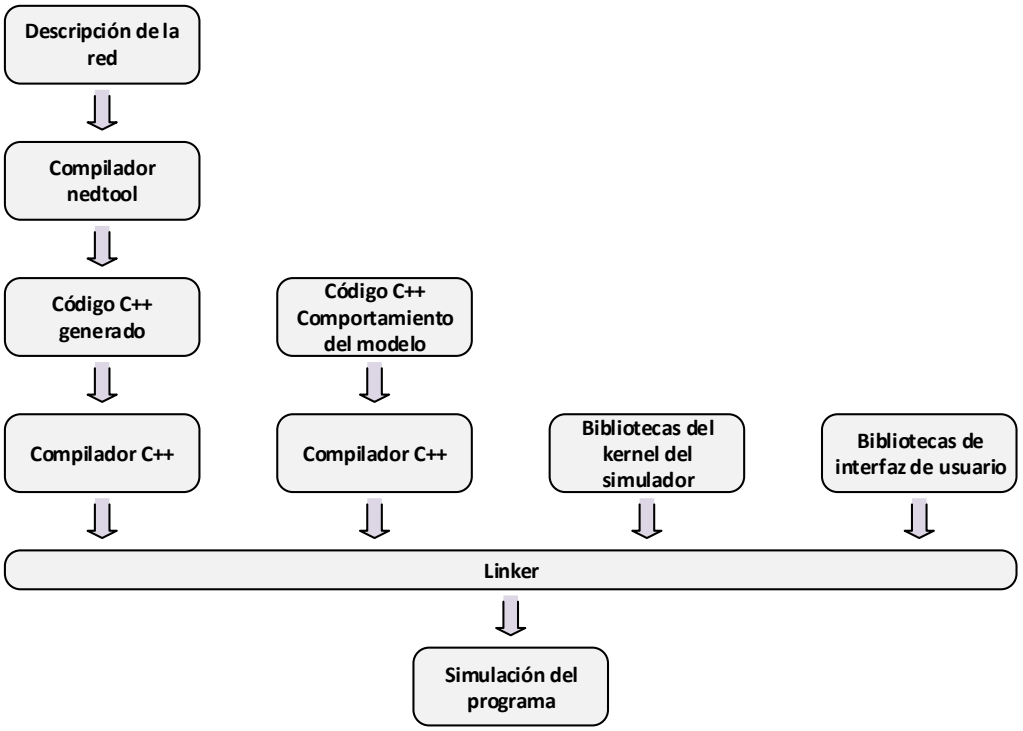

FIGURA 3.2: Proceso de construcción de un modelo en *OMNeT++*.

i. **Archivos de descripción de red (.ned):** los archivos *NED* describen la estructura del modelo empleando parámetros, conexiones, canales, submódulos, entre otros. Además pueden ser generados mediante líneas de comando o usando la herramienta de diseño (*GNED*). Una ventaja que ofrece *OMNeT++* es que los componentes creados pueden ser reutilizados en otros modelos de red.

ii. **Definiciones de mensaje (.msg):** estos archivos permiten modelar varios elementos característicos de una red, por ejemplo: eventos, mensajes, paquetes, tramas, celdas y señales. Mediante la herramienta *nedtool* al momento de compilar el proyecto estos elementos son generados como clases en *C++*.

iii. **Código fuente (.h/.cc):** como se mencionó anteriormente, los módulos simples son los componentes activos de la red, por lo cual éstos son los únicos elementos programados en *C++* mediante la biblioteca *omnetpp.h*.

iv. **Archivos de configuración (.ini):** para ejecutar una simulación, *OMNeT++* requiere de un archivo de inicialización. Éste se divide en diferentes secciones, que incluyen información sobre los parámetros generales, parámetros específicos de la ejecución, almacenamiento de los vectores de salida y valores de configuración para ciertos módulos.

3.1.3. Clases principales de OMNeT++

Presentamos las *clases* que posee *OMNeT++*.

3.1.3.1. Clase cSimpleModule

Esta clase y sus subclases derivadas implementan el comportamiento de los módulos simples descritos en el modelo de red. Las funciones definidas en la clase *cSimpleModule* tienen como objetivo la generación de eventos y la respuesta del modelo ante la presencia de ellos [43].

Las funciones virtuales que componen esta clase son:

i. **initialize():** esta función es la encargada de obtener los parámetros de inicialización configurados en el archivo escrito en lenguaje NED o descritos en el archivo de inicialización, para después asignarlos a los módulos correspondientes en lenguaje *C++*.

ii. **handleMessage():** en esta función se implementa el comportamiento de los módulos ante la llegada de información.

iii. **finish():** su implementación es opcional y es llamada cuando las simulaciones finalizan satisfactoriamente. Por lo general, es empleada para la recopilación de los resultados de la simulación.

3.1.3.2. Clase cMessage

Esta clase es utilizada para representar unidades de datos como: mensajes, paquetes, tramas, etc; los cuales pueden ser enviados de un módulo a otro o al mismo módulo.

En las siguientes secciones se explica la implementación del modelo de red.

3.2. Plan de Contactos

La topología de red consiste en un tren de cinco satélites *LEO* en donde se asume que todo el compartimiento del tráfico es conocido de antemano, es decir, el volumen de datos, origen/destino de las imágenes y su tiempo de generación es programado. En la figura 3.3 se observa que cada nodo se comunica con los otros nodos a razón de un nodo a la vez dependiendo de los tiempos definidos por el plan de contactos.

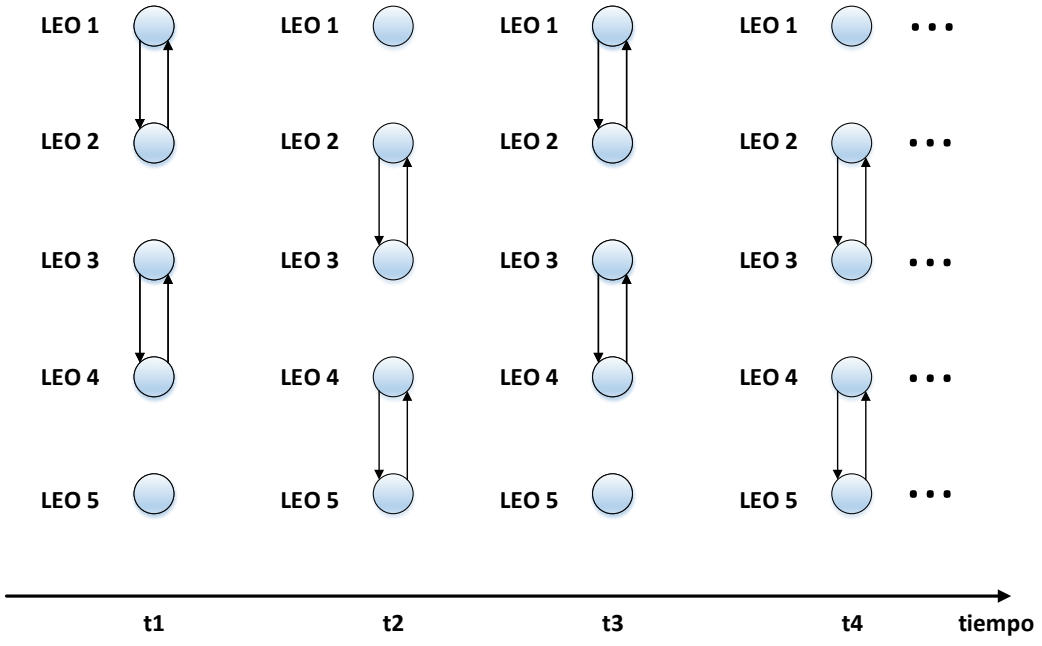

FIGURA 3.3: Estructura de los nodos satelitales. Esta gráfica indica las oportunidades de contacto para enviar un paquete entre cada uno de los nodos con respecto a una escala de tiempo definida por el plan de contactos. Los círculos representan cada uno de los nodos y estos son denotados mediante las siglas *LEO*.

Los planes de contacto generados toman como dato que para una altitud de 520 km el período orbital de una estación es de 90 minutos [10]. Esto implica que un satélite puede

proporcionar solamente conectividad intermitente con alguna estación terrena, mientras que para brindar una conectividad continua se requiere una constelación de satélites.

El programa que genera los planes está dividido en dos etapas: la primera genera un plan de contactos que incluye solamente las estaciones satelitales en función al tiempo de conexión entre ellos, mientras que la segunda etapa consiste en generar cada una de las descargas de datos hacia las estaciones terrenas (ver figura 3.4). Es importante recalcar que en los primeros escenarios de simulación se calculará un plan de contactos para un caso ideal en donde por período orbital cada nodo tendrá la posibilidad de tener conexiones con alguna estación terrena. Esto se debe a que en un escenario real en donde se necesite descargar la información por cada período orbital es necesario emplear un mayor número de estaciones terrenas. Sin embargo, en el último escenario (ver sección 4.7) se utilizará un plan de contactos calculado en función de parámetros orbitales reales, en donde para un tiempo de 24 horas todas las estaciones espaciales tendrán varias oportunidades para bajar sus imágenes.

Los planes de contacto son generados como archivos de texto plano (ver figura 3.5). El archivo de salida tiene diferentes *id"* que identifican a cada contacto y los nodos involucrados (denotados mediante las siglas *LEO*). Asimismo cada contacto contiene el tiempo de inicio de conexión entre dos nodos y su duración en segundos para una oportunidad de comunicación. Estos datos son importados posteriormente al simulador mediante manejo de ficheros en *C++*.

3.3. Cálculo de tráfico generado por nodo

En este apartado se explica el procedimiento utilizado para generar el tráfico en cada uno de los nodos. Este cálculo hace referencia a la cantidad y tiempo de generación de las imágenes creadas en los nodos satelitales para ser posteriormente enviados hacia alguna estación terrena.

3.3.1. Consideraciones para generar el tráfico en los nodos

Una serie de archivos de imágenes tiene que ser transferido como sea posible desde el satélite a un centro de control para su posterior análisis. Debido a la baja órbita del

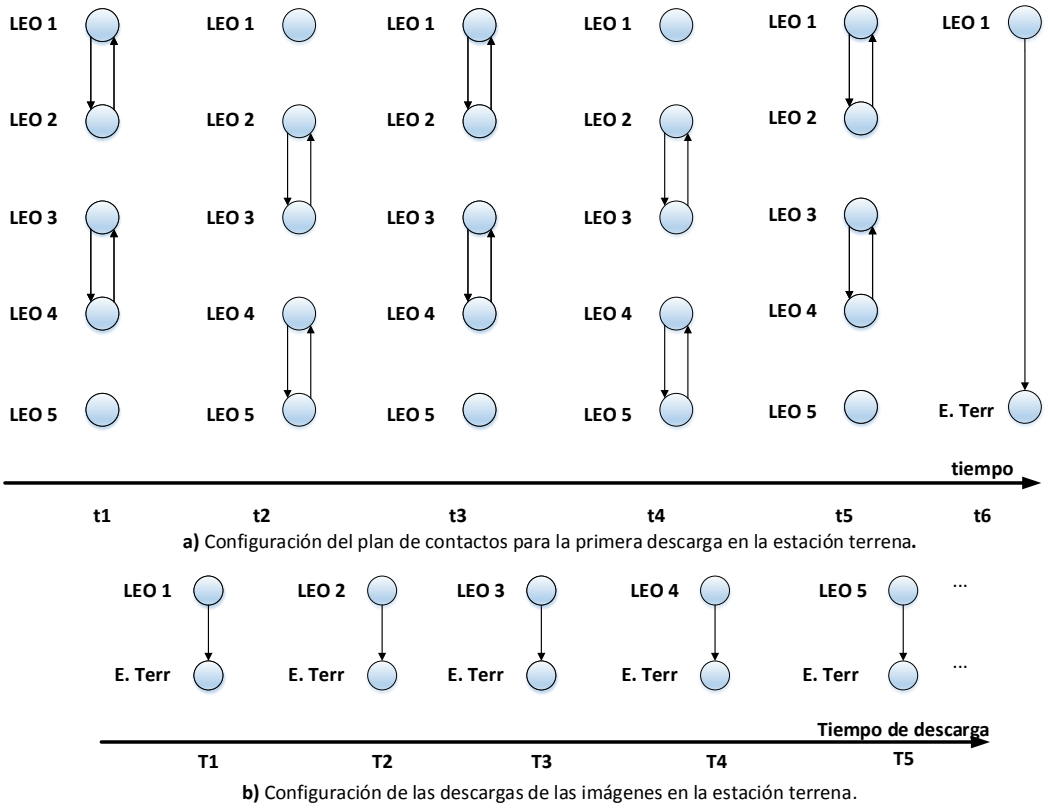

FIGURA 3.4: Los planes de contactos son generados de tal manera que solamente un nodo de la constelaciones de satélites puede descargar datos en alguna estación terrena por cada período orbital **(Caso Ideal)**. Como se ilustra en la gráfica inferior el proceso de descarga de archivos desde los nodos en órbita es continuo, es decir, comienza el nodo *LEO1* y lo termina el nodo *LEO5*, y este proceso se repite hasta el período de tiempo configurado en el plan de contactos.

satélite, la descarga de estos datos es posible cuando el satélite tiene línea de vista con la estación receptora terrestre. Por otra parte, las ventanas de contacto cortas y bajas velocidades de transmisión imponen un límite en el volumen de contactos, es decir, sobre el número de imágenes que se pueden descargar en un solo paso. Si los datos para ser descargados superaron el volumen de contactos, múltiples pasadas o el uso de otros nodos son necesarios.

Para el cálculo de los parámetros de tráfico se tomó en cuenta lo siguiente:

i. El período orbital es una variable del programa. En las simulaciones se emplea como período orbital un valor de 90 minutos.

ii. Suponemos una velocidad de transmisión de datos (bps) entre cada nodo satelital y la estación terrena, como también una ventana de contacto (segundos), y resultado

```
# Archivo que contiene las ventanas para cada segmento del sistema.
# Formato de los Datos:
# id      SegmA  SegmB  Tiempo_de_inicio(seg)    Duracion_ventana(seg)

1         LEO1   LEO2   1          899
2         LEO3   LEO4   1          899
3         LEO2   LEO3   900        900
4         LEO4   LEO5   900        900
5         LEO1   LEO2   1800       900
6         LEO3   LEO4   1800       900
7         LEO2   LEO3   2700       900
8         LEO4   LEO5   2700       900
9         LEO1   LEO2   3600       900
10        LEO3   LEO4   3600       900
11        LEO2   LEO3   4500       900
12        LEO4   LEO5   4500       900
```

FIGURA 3.5: Formato del plan de contactos generado mediante Python. La primera columna representa el identificador de cada uno de los contactos. La segunda y tercera contiene los nodos involucrados en ese contacto. Finalmente, las últimas columnas indican el tiempo de inicio y su duración medido en segundos [45].

del producto de éstos, un volumen de contactos (bits). Este valor en particular limita la cantidad de información que puede ser transmitida desde los nodos por cada oportunidad de descarga a Tierra.

iii. Consideramos un ancho de banda asimétrico desde los nodos en órbita hacia las estaciones terrenas dado que el objetivo es solamente la descarga de imágenes desde los nodos en órbita.

iv. La configuración del plan de contactos puede ocasionar que un nodo en un determinado tiempo utilice otros nodos para descargar sus archivos en el destino. Cabe recordar que *CGR* selecciona la mejor ruta desde el satélite a la estación terrena destino, teniendo en cuenta la secuencia temporal de los contactos y sus volúmenes de contacto que son valores limitados. Esto implica que la capacidad de almacenamiento de los nodos sea un factor importante a tener en cuenta en las simulaciones.

v. La cantidad y el tamaño de las imágenes son valores a definir, ya que posteriormente estos valores permitirán generan el archivo de tráfico que será cargado en los nodos. Estos valores permiten calcular el tiempo de generación de las imágenes y la cantidad de paquetes por imagen.

vi. El tamaño de cada paquete es medido en bytes y su tiempo de generación es de cero segundos.

La figura 3.6 muestra el archivo de tráfico generado, el cual contiene los siguientes parámetros: tasa de transmisión de las estaciones satelitales y terrena, la capacidad de almacenamiento de cada uno de los nodos, cantidad de imágenes y paquetes, y finalmente los tiempos de generación [45].

Archivo que contiene los parametros de trafico a cargar en cada uno de los nodos

1	20	2	1430	214	5670	0.05	10000	2000	25	2892
2	20	2	1430	214	5670	0.05	10000	2000	25	2892
3	20	2	1430	214	5670	0.05	10000	2000	25	2892
4	20	2	1430	214	5670	0.05	10000	2000	25	2892
5	20	2	1430	214	5670	0.05	10000	2000	25	2892

FIGURA 3.6: Formato del archivo de trafico generado mediante *Python*. La primera columna representa al nodo en cuestión. La segunda y tercera contiene la tasa de transmisión de los enlaces de los nodos satelitales y las estaciones terrenas en (Mbps). Las siguientes dos columnas representan la capacidad de almacenamiento de los nodos en (MB). A continuación, la quinta y sexta columna describen la cantidad de paquetes por imagen y su tiempo de generación. Finalmente, las últimas cuatro columnas indican el tiempo de vida de los paquetes, su tamaño en bytes, la cantidad de imágenes y su tiempo de generación en segundos [45].

3.4. Implementación del modelo de red en OMNeT++

Los pasos a tomar en cuenta para implementar una simulación es crear un directorio de trabajo, y después describir la red mediante la creación de un archivo de topología de extensión **.ned** como se muestra en la figura 3.7. La red se ensambla a partir de submódulos, es decir cada nodo en la red es una instancia del módulo **Node**. De esta manera, cada uno de los submódulos **node** hereda las características del módulo principal.

El siguiente paso es dar la funcionalidad de cada uno de los nodos mediante *C++*. Los nodos están representados por la clase de *C++* **node**, que a su vez es una subclase de **cSimpleModule**, y registrada en *OMNeT++* mediante **DefineModule**.

En la sección 3.1.3 se mencionó que la clase *cSimpleModule* tiene tres métodos virtuales: *initialize*, *handleMessage* y *finish*. Cada uno cumple un rol en específico y son ejecutados desde el *kernel* de la simulación: el primero se ejecuta una sola vez, el segundo cada vez

```
package CgrNet;
simple Node
{
    parameters:
        string planContactosFile = default("planContactos.txt");
        string traficoGen = default("traficoGen.txt");
    gates:
        inout gate @loose;
}

network CgrNet
{
    parameters:
        int nodeNum = default(5);
    submodules:
        node[nodeNum]: Node;
}
```

FIGURA 3.7: Los nodos son submódulos del módulo principal **Node**. Los parámetros que comparten cada uno de los nodos son el archivo del plan de contactos y el volumen de tráfico a generar.

que un mensaje llega al módulo, y el tercero al finalizar la simulación. La función de cada uno de ellos en el proyecto se describe en los siguientes apartados.

3.4.1. Método initialize

El método *initialize* es invocado al inicio de la simulación, y permite cargar los valores de configuración en los nodos. Para este caso, mediante las clases y funciones de *C++* que permiten manipular ficheros se importa el plan de contactos y el archivo que contiene los parámetros del tráfico en cada uno de los módulos.

El plan de contactos es cargado como estructuras en *C++* y como mensajes en *OMNeT++*. Donde cada mensaje aloja el contacto y la velocidad de transmisión del enlace. Cabe recordar que los objetos *cMessage* permiten crear mensajes, paquetes y eventos como temporizadores. Esto ayuda a determinar cuándo existe una oportunidad de contacto con cierto nodo en particular para un tiempo de simulación determinado.

Finalmente, el archivo de tráfico carga en los nodos los valores de la velocidad de transmisión de los enlaces, capacidades de almacenamiento y los tiempos de generación de los paquetes e imágenes.

3.4.2. Método handleMessage

Éste contiene a su vez otros métodos que cumplen el objetivo de verificar si el paquete llegó a su destino. El paquete tiene como parámetros un identificador, la información del nodo que lo generó y el tiempo de creación. Este tiempo es manejado mediante funciones dedicadas a comprobar que el paquete sea generado con respecto al tiempo de generación de cada imagen programado previamente en *Python*. Después de su creación, es enviado a otro método que verifica si llegó a destino, si es que no es el caso, se programa su envío hacia otro nodo considerando el vecino calculado mediante *CGR*.

No obstante, el envío de paquetes está sujeto a las oportunidades de comunicación configuradas en el plan de contactos. Por lo que el método handleMessage se encarga de verificar los tiempos de inicio y fin de contacto en cada uno de los nodos para habilitar la conexión hacia el nodo programado, es decir encender el enlace satelital. Finalmente, si las condiciones lo permiten el paquete puede ser enviado y recibido por el siguiente contacto temporal.

Es importante mencionar, que se están utilizando las funciones y clases que ya tenía programadas *Totsim*[2] para crear y manipular cada uno de los contactos. Como base se empleó la clases programadas en *C++*, que implementan los algoritmos de *Contact Graph Routing*.

3.4.3. Método finish

Este método en algunos casos es opcional, y es utilizado para almacenar los valores generados en las simulaciones mediante variables escalares o vectores. Sin embargo, en las simulaciones realizadas al utilizar esta función en conjunto con el registro de eventos influyeron para que las simulaciones duraran una gran cantidad de tiempo por la cantidad de datos generados. En respuesta a ello se decidió utilizar archivos de texto para almacenar los datos generados que posteriormente serán analizados mediante Python.

[2]Software desarrollado en el Laboratorio de Comunicaciones Digitales de Universidad Nacional de Córdoba-Argentina.

3.5. Recolección y análisis de los datos

Los datos generados en la simulación son almacenados en un archivo de texto plano que sigue una determinada estructura (ver figura 3.8). Estos archivos por cada línea describen: el identificador del paquete, los nodos origen, destino y el nodo actual en el que se encuentra el paquete. Asimismo, se almacena los valores del tiempo de vida, el identificador de la imagen, y la capacidad de almacenamiento del nodo en un determinado tiempo de simulación. En otras palabras, por cada nodo se crea un archivo el cual almacena todos los eventos de cada paquete al pasar por cada uno de los nodos. Es importante mencionar que la implementación de los algoritmos que procesan la información generada en *OMNeT++* se explica en detalle en el Anexo A [45].

```
400 , 1 , 6 , 1 , 1.45 , 10000 , 18000 , 1
401 , 1 , 6 , 1 , 1.50 , 10000 , 20000 , 1
402 , 1 , 6 , 1 , 1.55 , 10000 , 22000 , 1
403 , 1 , 6 , 1 , 1.60 , 10000 , 24000 , 1
404 , 1 , 6 , 1 , 1.65 , 10000 , 26000 , 1
405 , 1 , 6 , 1 , 1.70 , 10000 , 28000 , 1
406 , 1 , 6 , 1 , 1.75 , 10000 , 30000 , 1
407 , 1 , 6 , 1 , 1.80 , 10000 , 32000 , 1
408 , 1 , 6 , 1 , 1.85 , 10000 , 34000 , 1
409 , 1 , 6 , 1 , 1.90 , 10000 , 36000 , 1
410 , 1 , 6 , 1 , 1.95 , 10000 , 38000 , 1
```

FIGURA 3.8: Formato de los archivos generados en *OMNeT++* para el nodo *LEO1*. El archivo compuesto por ocho columnas contiene en la primera columna el identificador del paquete, las siguientes cuatro columnas describen los nodos origen, destino, el nodo actual y el tiempo de simulación (seg). La columna seis y siete almacenan el tiempo de vida del paquete (seg) y la capacidad de almacenamiento del nodo (bytes) en ese instante de tiempo. Finalmente, la última columna especifica el identificador de la imagen [45].

El siguiente paso consiste en importar todos estos datos y analizarlos con *Python*. Python tiene muchas ventajas[3], entre la cuales se destacan el manejo de estructuras de datos como diccionarios, lo cual facilita en gran medida el procesamiento de la información, además de la gran cantidad de bibliotecas que permiten el análisis estadístico de los datos.

El análisis de los datos se dividió en dos etapas que son descritas en los siguientes apartados.

[3]Ver `http://www.maestrosdelweb.com/ventajas-python/`, accedido el 8/07/2015

3.5.1. Procesamiento de los datos

Esta etapa procesa los seis archivos creados en *OMNeT++* (incluyendo a la estación terrena), con el propósito de generar cinco nuevos archivos los cuales almacenan los paquetes generados por cada nodo y su trayecto del origen al destino como se indica en la figura 3.9. Esto se debe a que los ficheros generados en *OMNeT++* registran el paso de los paquetes por el nodo pero no su trayecto [45].

```
400 , 1 , 6 , 1 , 1.45, 10000 , 18000.0 , 1
400 , 1 , 6 , 6 , 5410.01, 10000 , 20000.0 , 1
401 , 1 , 6 , 1 , 1.5 , 10000, 20000 , 1
401 , 1 , 6 , 6 , 5410.01 , 10000 , 22000 , 1
402 , 1 , 6 , 1 , 1.55 , 10000 , 22000.0 , 1,
402 , 1 , 6 , 6 , 5410.01 , 10000 , 24000.0 , 1
403 , 1 , 6 , 1 , 1.6 , 10000 , 24000.0 , 1
403 , 1 , 6 , 6 , 5410.01 , 10000 , 26000.0 ,1
```

FIGURA 3.9: Formato de los archivos procesados con Python para el nodo LEO 1. Este fichero sigue la misma estructura descrita previamente en la figura 3.8. La diferencia radica en que este nuevo fichero se almacena el trayecto de cada uno de los paquetes. En este caso, el trayecto de los paquetes 400, 401, 402 y 403 desde el nodo origen 1 hacia el nodo destino 6 que identifica la estación terrena [45].

3.5.2. Análisis de las imágenes

El objetivo de esta etapa es analizar todas las imágenes generados por cada nodo. Las funciones implementadas permiten calcular los siguientes parámetros: cantidad de imágenes y paquetes entregados correctamente al destino, tiempo de entrega de cada imagen, número de saltos promedio del paquete y su tiempo de vida por imagen, y finalmente el nodo del último salto, que en el caso de existir más de uno se registrará la cantidad de paquetes entregados al destino.

El resultado del análisis de cada imagen es almacenado en un archivo tal como se muestra en la figura 3.10, esta figura es un extracto de las primeras 10 imágenes del archivo que contiene todos los resultados obtenidos de la simulación.

```
1,1,686,686,1,1.00,0.00,5388.17,6.17,0.0,5412.75,1,686,2,0,3,0,4,0,5,0,
1,2,686,686,1,1.00,0.00,2176.73,8.03,3235.3,5432.39,1,686,2,0,3,0,4,0,5,0,
1,3,686,686,1,2.00,0.00,4319.58,6.17,6469.6,10812.78,1,0,2,686,3,0,4,0,5,0,
1,4,686,686,1,3.00,0.00,6508.13,8.03,9703.9,16232.39,1,0,2,0,3,686,4,0,5,0,
1,5,686,686,1,3.00,0.00,3283.37,6.17,12938.2,16245.17,1,0,2,0,3,686,4,0,5,0,
1,6,686,686,1,4.00,0.00,5439.53,8.03,16172.5,21632.39,1,0,2,0,3,0,4,686,5,0,
1,7,686,686,1,5.00,0.00,7624.66,8.03,19406.8,27051.82,1,0,2,0,3,0,4,0,5,686,
1,8,686,686,1,5.00,0.00,4396.84,8.03,22641.1,27058.3,1,0,2,0,3,0,4,0,5,686,
1,9,686,686,1,1.00,0.00,6510.72,8.03,25875.4,32406.48,1,686,2,0,3,0,4,0,5,0,
1,10,686,686,1,1.00,0.00,3289.38,8.03,29109.7,32419.43,1,686,2,0,3,0,4,0,5,0,
```

FIGURA 3.10: Extracto del archivo que contiene los resultados del análisis de cada una de las imágenes. En este caso se muestra las 10 primeras imágenes del nodo *LEO1*. El formato del archivo es el siguiente: las dos primeras columnas identifican el nodo y la imagen, la tercera y cuarta columna representan la cantidad de paquetes generados y entregados correctamente al destino. De la quinta a la novena columna se indica el número de saltos promedio y el tiempo de vida de los paquetes por imagen. El tiempo de creación y entrega de la imagen al destino son mostrados en las siguientes dos columnas. Las últimas columnas representan el nodo del último salto, en este ejemplo el nodo 1 fue el encargado de entregar la totalidad de los paquetes al destino [45].

Capítulo 4

Escenarios estudiados y análisis de los resultados

En el presente capítulo se describen los escenarios empleados para evaluar el rendimiento del algoritmo *Contact Graph Routing* (*CGR*). La topología de red consiste de un tren de cinco satélites *LEO* tal como se indica en la figura 4.1. Donde el rango de comunicación para los enlaces entre satélites es de 300 km mientras que el enlace con las estaciones terrenas es de 1.000 km.

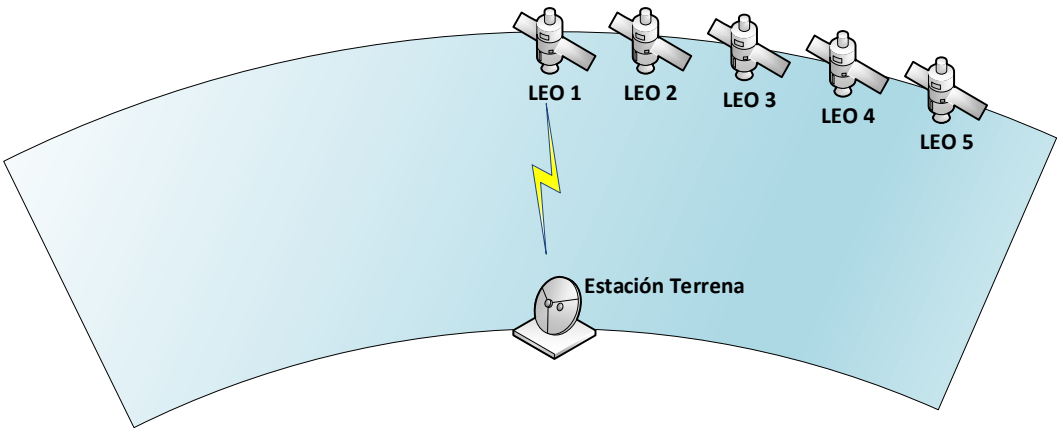

FIGURA 4.1: La red utilizada en las simulaciones consiste de una constelación de cinco satélites los cuales tienen conexión con alguna estación terrena.

Los parámetros de inicialización para habilitar el funcionamiento de *CGR* en cada uno de los nodos son descritos al comienzo del capítulo. Posteriormente, en las siguientes

secciones se detallan cada uno de los escenarios implementados, tomando en cuenta en cada uno, el objetivo y los parámetros de simulación, los cuales son calculados en función del plan de contactos y el tráfico configurado.

4.1. Inicialización de los parámetros en OMNeT++

El simulador *OMNeT++* requiere de un archivo de configuración para ejecutar una simulación. Este se divide en secciones y contiene información tales como: parámetros generales y específicos de ejecución, especificación de valores para ciertos módulos, entre otros [44].

La figura 4.2 muestra el archivo de configuración para este proyecto. Éste contiene los siguientes parámetros: el número de nodos, los archivos que almacenan el plan de contactos y el tráfico generado, las banderas que habilitan el enrutamiento de *CGR* en cada nodo, y el destino de los paquetes.

```
[General]
network = CgrNet
CgrNet.nodeNum = 6        # Numero de nodos
CgrNet.*.planContactosFile = "planContactos.txt"
CgrNet.*.traficoGen = "traficoGen.txt"

###################Parametros a considerar para realizar el ruteo

CgrNet.node[1].habilitaRuteo=true      # true, habilita ruteo
CgrNet.node[2].habilitaRuteo=true
CgrNet.node[3].habilitaRuteo=true
CgrNet.node[4].habilitaRuteo=true
CgrNet.node[5].habilitaRuteo=true
CgrNet.node[6].habilitaRuteo=true


CgrNet.node[1].bundleNodeDest=6        # Nodo destino
CgrNet.node[2].bundleNodeDest=6
CgrNet.node[3].bundleNodeDest=6
CgrNet.node[4].bundleNodeDest=6
CgrNet.node[5].bundleNodeDest=6
```

FIGURA 4.2: Archivo de configuración para las simulaciones en *OMNeT++*. En la parte superior se especifica el número de nodos, y el directorio de los archivos que albergan el plan de contactos y tráfico generado. La siguiente sección de este archivo contiene las banderas que habilitan el enrutamiento en los nodos y del destino de los paquetes.

4.2. Antecedentes

Tal como se describió en la sección 3.3.1, una serie de imágenes tiene que ser transferida desde los nodos en órbita hacia alguna estación terrena, y la descarga de estos archivos es posible cuando el satélite tiene línea de vista con la estación receptora. Además, las ventanas de contacto y las velocidades de transmisión en cada uno de los enlaces imponen una capacidad límite de información para transmitir por el nodo de último salto designado en cada pasada por la estación terrena.

4.2.1. Descripción general de los escenarios

En el primer escenario se realiza una simulación base donde las capacidades de almacenamiento de los nodos son lo suficientemente grandes para no descartar paquetes provenientes de otros nodos. Además los valores son configurados con el objetivo de que todas las imágenes lleguen completas a la estación terrena. Posteriormente, en el segundo escenario se limita la capacidad de almacenamiento de los nodos satelitales y observaremos algunos inconvenientes de *CGR*, ya que éste no considera el estado en el que se encuentran los nodos a los cuales se les va a enviar información. Para el tercer escenario se analizan dos puntos. Por un lado se plantea un escenario en el cual el sistema trabaja a su máxima capacidad teórica, y por otro, se analiza cuál es el valor óptimo del sistema para que se transmitan todos los paquetes de forma correcta. Más adelante, en el cuarto escenario se asignan probabilidades de pérdida de contactos entre los enlaces. Finalmente, en el último escenario se realiza una simulación utilizando un plan de contactos generado mediante parámetros orbitales reales.

4.2.2. Descripción de las variables de simulación

Las variables empleadas en cada una de las simulaciones son descritas en la tabla 4.1. En particular, el tiempo de generación de cada imagen *(timeGenImage)* describe cada cuanto tiempo se generá una imagen en cada estación espacial. Además, este valor esta en función de la cantidad de imágenes a generar por nodo *(cantImage)* y el tiempo de simulación *(simulationTime)*.

Variable	Descripción
orbitalTime	Período de tiempo, en el cual un satélite da una vuelta a la tierra [minutos].
datarateTerr	Velocidad de transmisión del enlace entre los satélites y la estación terrena [Mbps].
datarateSat	Velocidad de transmisión entre los enlaces satelitales [Mbps].
bufferSizeTerr	Capacidad de almacenamiento de la estación terrena [MB].
bufferSizeSat	Capacidad de almacenamiento de las estaciones satelitales [MB].
simulationTime	Tiempo de simulación [horas].
timeSat	Ventana de contacto entre las estaciones satelitales [minutos].
timeTerr	Ventana de contacto entre los satelitales y la estación terrena [minutos].
cantBundles	Cantidad de bundles ó paquetes por imagen.
bundleSize	Tamaño de los paquetes [bytes].
bundleTTL	Tiempo de vida de cada paquete [segundos].
cantImagen	Cantidad de imágenes a generar en cada nodo.
imagenSize	Tamaño de cada imagen [MB].
timeGenImage	Tiempo de generación entre cada imagen [segundos]

Tabla 4.1: Descripción de las variables empleadas en las simulaciones.

4.3. Escenario 1: Simulación Base

En este escenario partiremos de unas condiciones base, es decir, los nodos tendrán una capacidad de almacenamiento lo suficientemente grande como para evitar que éstos eliminen los paquetes entrantes cuando su capacidad de almacenamiento es superada. Cabe recordar que *CGR* calcula las rutas en función de la capacidad residual de los contactos y el tiempo de vida de los paquetes, pero no toma en cuenta el estado actual en el que se encuentran los nodos a los cuales se les va a enviar datos.

Los valores de configuración de este escenario son descritos en la tabla 4.2, en donde se ha considerado que cada nodo transmitirá 75 imágenes. Como resultado, cada nodo generará un tráfico total de 750 MB que equivale a 51.450 paquetes, y por lo tanto en la simulación todos los nodos generarán un total de 3,66 GB de información. Por otra parte, el plan de contactos calculado contiene 497 ventanas de contacto para la conexión entre los satélites y las estaciones terrenas (ver tabla 4.3). La configuración del plan de contactos está en función del tiempo que dura la simulación y los valores de las ventanas de contacto entre los satélites y de éstos con la estación terrena.

Variable	Valor configurado
orbitalTime	90 minutos
datarateTerr	20 Mbps
datarateSat	2 Mbps
bufferSizeTerr*	∞
bufferSizeSat*	∞
simulationTime	67hs 40min
timeSat	15 minutos
timeTerr	10 minutos
cantBundles	686
bundleSize	14,9 KB
bundleTTL	10.800 segundos
cantImagen	75
imagenSize	10 MB
timeGenImage	3.200 segundos

Tabla 4.2: Valores de configuración para el Escenario 1.
***** Se configuró la capacidad de almacenamiento en los nodos con valores grandes (100.000 MB), ya que no es posible establecer como valor numérico infinito en las simulaciones.

Enlace	Cantidad de contactos
Nodos $\leftrightarrow$ Estaciones terrenas *	9
LEO1 $\leftrightarrow$ LEO2	113
LEO2 $\leftrightarrow$ LEO3	113
LEO3 $\leftrightarrow$ LEO4	113
LEO4 $\leftrightarrow$ LEO5	113

Tabla 4.3: Ventanas de contacto entre nodos.
***** Cada nodo tiene 9 oportunidades de contacto para enviar la información generada en todas las estaciones satelitales a las estaciones terrenas. Estos valores fueron calculados para un caso ideal en donde por cada período orbital cada nodo tendrá la posibilidad de tener conexiones con alguna estación terrena.

El tiempo de generación de cada imagen se estableció en 3.200 segundos. Este valor fue calculado con el propósito de que la última imagen generada por el nodo *LEO1* (t=240.000s) posea una oportunidad de contacto para transmitir sus imágenes a la estación terrena receptora mediante el nodo de último salto *LEO5* que se encuentra al extremo de la constelación como se indica en la figura 4.3.

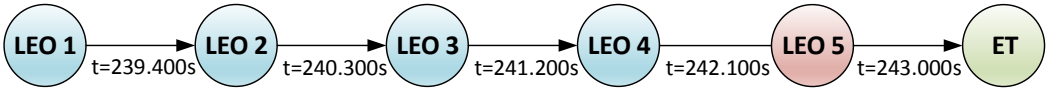

Figura 4.3: Configuración del plan de contactos para la última descarga de datos en la estación terrena por el nodo *LEO5*. El valor de tiempo entre dos nodos representa el tiempo de inicio de contacto con una duración de 900 segundos.

4.3.1. Cálculo de Contact Graph Routing en los nodos

El cálculo de enrutamiento que realiza *CGR* es por cada paquete; además, éste es ejecutado de forma recursiva hasta que éste encuentra una lista de vecinos de un salto que sean capaces de entregar la información al nodo de destino. Por ello algunos paquetes necesitan mayores cómputos de *CGR* que otros, dado que en el proceso del cálculo del algoritmo éste puede encontrarse con ventanas de contactos que expiraron o enlaces ya saturados, y por lo tanto necesitará realizar un nuevo cálculo para ese paquete. Cabe recordar que toda esta información se maneja desde un punto de vista local en cada nodo. En este contexto, la figura 4.4 describe la cantidad de veces que *CGR* fue calculado por paquete en cada uno de los nodos. En esta simulación el algoritmo fue ejecutado 6.664.573 veces.

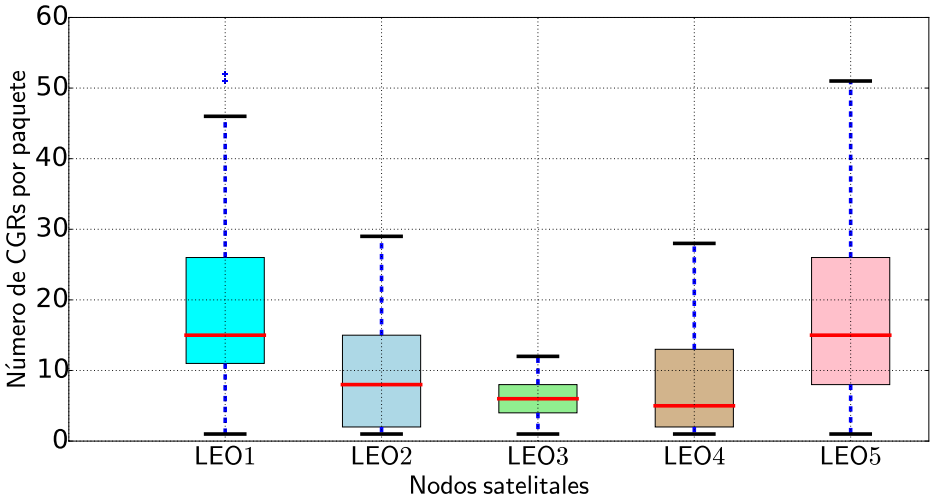

FIGURA 4.4: Éste diagrama de cajas representa la cantidad de veces que *CGR* es calculado para encontrar una ruta de un paquete, estos resultados fueron obtenidos con los valores de configuración de la tabla 4.2. Como se observa por lo menos una vez fue calculado el algoritmo en cada uno de los nodos para el envío de los paquetes.

El valor mínimo en todos los nodos coincide en un valor de 1, esto significa que por lo menos una vez fue calculado *CGR* para el enrutamiento de los paquetes. Adicionalmente, los nodos ubicados en los extremos del tren de satélites exhiben un mayor cálculo del algoritmo.

Es importante recalcar que *CGR* calcula toda la ruta por cada paquete que llega al nodo, y esta información no es almacenada en una tabla de enrutamiento para cálculos posteriores. Para ello se implementó una variante de *CGR* de *Totsim* la cual permite

almacenar las rutas y éstas son reutilizadas para el envío de los siguientes paquetes. De hecho estas rutas son recalculadas si es que suceden dos condiciones. Por un lado, no existe suficiente capacidad residual del contacto ya calculado para enviar ese paquete (ver sección 2.3.4.3), y por otro lado, el tiempo de las ventanas de contacto expiraron.

Con esta modificación se realizó una nueva simulación, y como resultado se disminuyó el cálculo de este algoritmo en un 63,71 %, en otras palabras *CGR* fue calculado 2.418.207 veces. Estos valores obtenidos involucran una gran mejora energética en los nodos satelitales en cuestión de procesamiento, considerando que los recursos que éstos manejan en el espacio son escasos. Por otro lado, en la figura 4.5 se muestra que la mayoría de las veces no se ejecuta *CGR*, ya que existen rutas previamente calculadas donde las capacidades residuales y ventanas de contactos permiten transmitir el paquete de análisis. Adicionalmente, el rango de veces que es calculado el algoritmo por cada nodo disminuye, y esto lo podemos comprobar si comparamos los valores de mediana de cada uno de los nodos con respecto al obtenido en la figura 4.4.

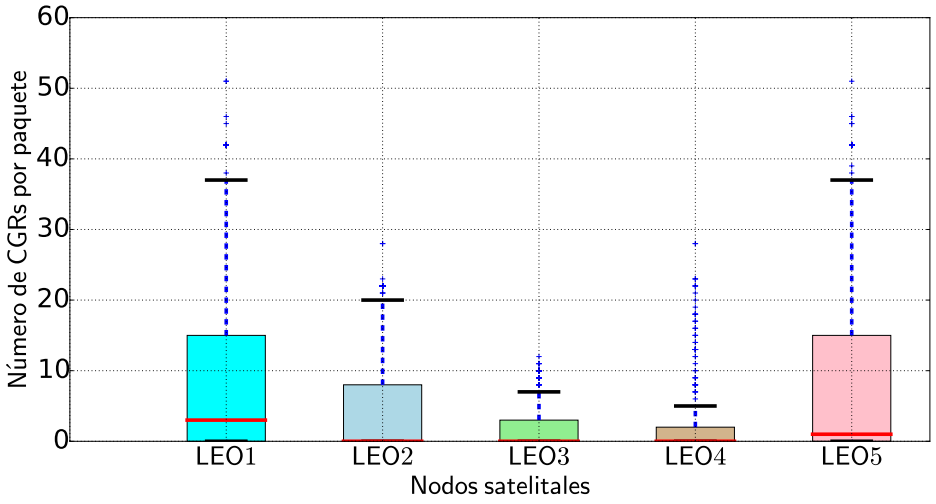

FIGURA 4.5: Se realizó una modificación de *CGR* para almacenar las rutas calculadas. Los valores de mediana 0 en los nodos *LEO2*, *LEO3* y *LEO4* representan que el 50 % de la veces *CGR* no necesitó realizar un cálculo de la ruta, debido a que existía una ruta previamente calculada. Estos datos fueron generados con los valores de la tabla 4.2.

4.3.2. Tasa de imágenes y paquetes transmitidos correctamente

Se implementaron funciones en *Python* para obtener y analizar cada una de las imágenes generadas en *OMNeT++* (ver secciones 3.5.2 y A.3). La figura 4.6 acentúa el hecho de

que todas las imágenes y paquetes fueron recibidos correctamente en el destino por el nodo de último salto configurado.

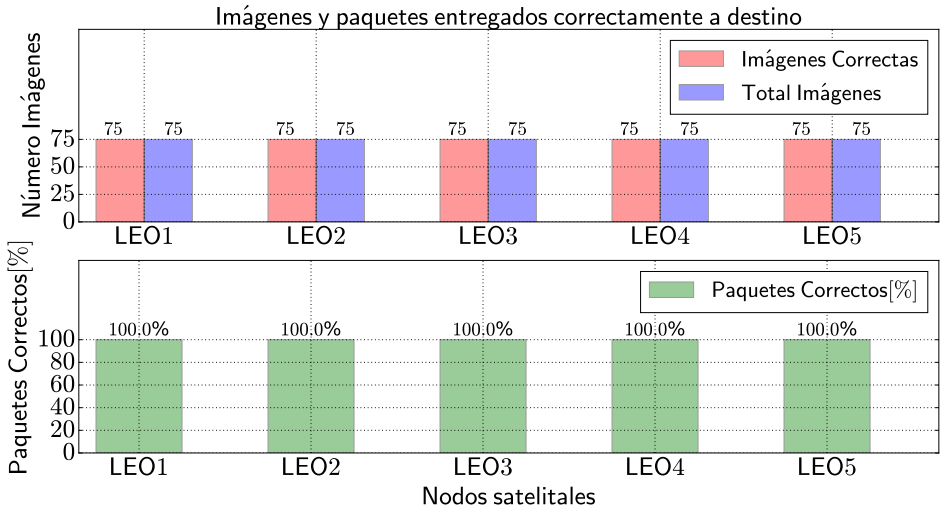

FIGURA 4.6: Tasa de imágenes entregadas correctamente con respecto a los valores de configuración de la tabla 4.2. El eje X registra cada uno de los nodos satelitales, mientras que el eje Y la cantidad imágenes y paquetes generados. Para este escenario todas las imágenes fueron entregadas correctamente en el destino.

4.3.3. Entrega de los paquetes por el nodo de último salto designado

Esta sección tiene como objetivo analizar dos puntos. En primer lugar, estudia el rendimiento de cada uno de los nodos, es decir, de la totalidad de paquetes generados qué porcentaje de éstos fueron transmitidos a las estaciones terrenas por el nodo de último salto configurado. Dado que se trabaja con 5 estaciones satelitales se espera que el 20 % de la información generada sea transmitida por cada nodo, con el propósito de que no existan estaciones satelitales sobre-utilizadas. Por otra parte, se evalúa si el nodo que capturó las imágenes es el encargado en la mayoría de las veces de entregar su propia información a las estaciones terrenas.

En este escenario un total de 257.250 paquetes fueron enviados correctamente, de los cuales el 22 % de la información fue trasmitida hacia las estaciones terrenas por el nodo *LEO2*, mientras que los otros nodos transmitieron de 17 % a un 21 % de los datos como se indica en la figura 4.7. Por otro lado, la parte derecha de la figura hace hincapié que *LEO1* fue empleado para entregar la mayoría de los paquetes generados por él mismo hacia las estaciones terrenas.

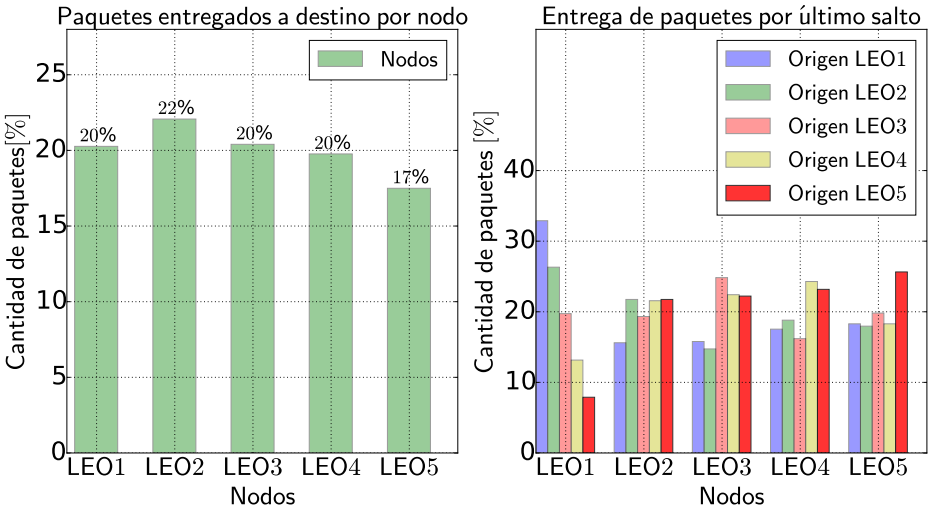

FIGURA 4.7: La figura de la derecha representa el porcentaje de paquetes entregados por cada uno de los nodos. Como ejemplo, del 100 % de paquetes entregados por *LEO1* a la estación terrena el 33 % corresponde a él mismo. La figura de la izquierda presenta qué porcentaje de los paquetes generados en toda la simulacion fueron entregados por el nodo de último salto configurado. Para este caso, *LEO2* fue empleado para entregar el 22 % de toda la información generada en este escenario. Estos datos fueron generados con los valores de la tabla 4.2.

4.3.3.1. Optimización del plan de contactos

El plan de contactos define las ventanas de comunicación entre dos nodos. Los valores de tiempo de conexión entre satélites, y éstos con las estaciones terrenas deben ser escogidos de tal manera que un nodo tenga una serie de contactos temporales[1] que le permitan descargar sus datos en alguna estación terrena considerando el caso ideal asumido en los escenarios.

La ubicación de los satélites en órbita y la configuración del plan de contactos influyen en que algunos nodos tengan una mayor cantidad de contactos temporales que otros. En la figura 4.8 podemos observar que el nodo *LEO5* para la primera descarga, tiene solamente una serie de contactos temporales con el nodo de último salto *LEO1*, mientras que el nodo *LEO2* que está contiguo a *LEO1* (nodo que descarga los datos en la estación terrena) tiene más oportunidades de enviar sus imágenes, ya que tiene 3 oportunidades para diferentes tiempos.

[1]Serie de contactos temporales, es la cantidad de contactos que tiene un nodo para enviar sus datos al destino.

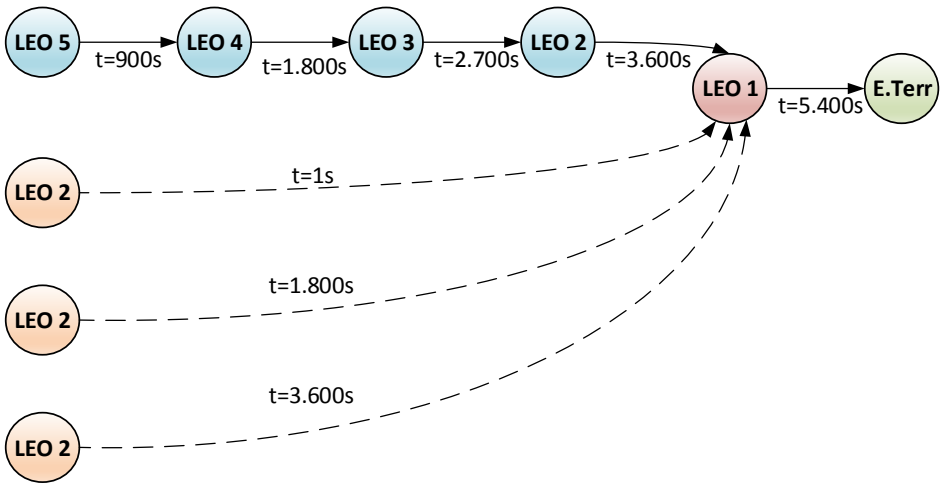

FIGURA 4.8: Cantidad de contactos temporales para los nodos *LEO2* y *LEO5* en la primera descarga en la estación terrena (5.400s). En este caso, *LEO5* tiene una serie de contactos temporales al nodo de último salto, mientras que *LEO2* por su vecindad con *LEO1* presenta tres contactos temporales. Los valores de tiempo representan los tiempos iniciales de la ventana de contacto entre dos nodos.

Para corregir esta peculiaridad, con *Python* se implementó un algoritmo que permite calcular la cantidad de contactos temporales de cada nodo por pasada, con el objetivo de eliminar contactos o enlaces satelitales que no influyen en el rendimiento de la red y que por cuestiones energéticas es mejor mantenerlos apagados. Las variables y funciones empleadas en este algoritmo son descritas en la tabla 4.4.

Algoritmo 3 Algoritmo que permite optimizar el plan de contactos.

Input: contactPlan, numVueltas y orbitalTime.
Output: newContactPlan.

```
1: lastHop=1
2: newContactPlan=[ ]
3: for i=1 to numVueltas do
4:    matrizVuelta=getVuelta(contactPlan, i , orbitalTime)
5:    minContact=getMinContact(matrizVuelta)
6:    for j=1 to numNod do
7:       newContacts=computeContacts(lastHop,j,matrizVuelta,minContact)
8:       append newContacts to newContactPlan
9:    end for
10:   lastHop=lastHop+1
11:   if lastHop > numNod then
12:      lastHop=1
13:   end if
14: end for
```

Variable	Descripción
orbitalTime	Período de tiempo, en el cual un satélite da una vuelta a la Tierra [minutos].
lastHop	Nodo que descarga las imágenes en la estación terrena.
contactPlan	Plan de contactos a optimizar.
newContactPlan	Plan de contactos optimizado.
numVueltas	Número de vueltas a la tierra
matrizVuelta	Matriz que contiene las ventanas de contacto de una vuelta a Tierra en particular.
getVuelta	Función que obtiene las ventanas de contacto de una vuelta a Tierra en particular.
getMinContact	Función que obtiene la cantidad mínima de contactos que tiene un nodo con la estación que descarga los datos en la estación terrena.
computeContact	Función que recalcula los contactos de los nodos con el nodo de último salto que descarga los datos en la estación terrena.
numNod	Cantidad de nodos satelitales.

Tabla 4.4: Descripción de las variables y funciones del algoritmo para optimizar el plan de contactos.

El algoritmo 3 genera una matriz, donde cada fila representa el número de vuelta por tierra y cada columna la cantidad de contactos temporales de cada nodo con el nodo que descarga las imágenes en la estación terrena (ver línea 4). Con los valores obtenidos de esta matriz se calcula un valor mínimo de contactos temporales para cada nodo, lo que permite eliminar contactos que no aportan positivamente en la red (ver línea 7). Por consiguiente, se eliminarían los dos primeros contactos entre *LEO2* y *LEO1* de la figura 4.8. Por lo tanto el nodo *LEO5* y *LEO2* tendrían las mismas oportunidades para el envío de las imágenes.

Mediante el algoritmo 3 se ejecutó una nueva simulación donde se obtuvieron los siguientes resultados. Por un lado, los paquetes generados en la simulación fueron distribuidos entre los nodos de último salto con valores cercanos al 20 % (ver figura 4.9). No obstante, al mejorar el plan de contactos permite que cada nodo entregue la mayoría de los paquetes que éste generó a la estaciones terrenas . Por otro lado, se consiguió disminuir el tamaño del plan de contactos de 497 a 317 ventanas (menor cantidad de enlaces encendidos), y por ende esto implica un menor cálculo de *CGR* en los nodos (ver figura 4.10). Estos resultados se traducen en un ahorro energético entre los enlaces, ya que éstos se mantienen apagados por mayores períodos de tiempo sin influir en la entrega correcta de las imágenes.

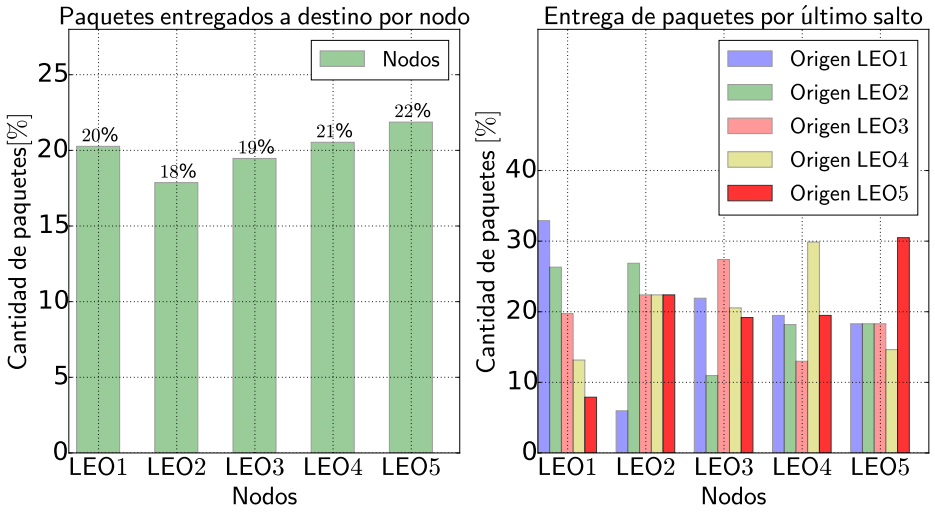

FIGURA 4.9: Al eliminar contactos que no influyen el rendimiento de la red, se consiguió de igual manera que casi el 20 % de la totalidad de la información generada en este escenario sea transmitida por cada nodo. Por otra parte, estos resultados influyeron para que cada nodo transmita la mayoría de los paquetes que generó a las estaciones terrenas tal como se describe en la figura de la derecha. Estos datos fueron generados con los valores de la tabla 4.2.

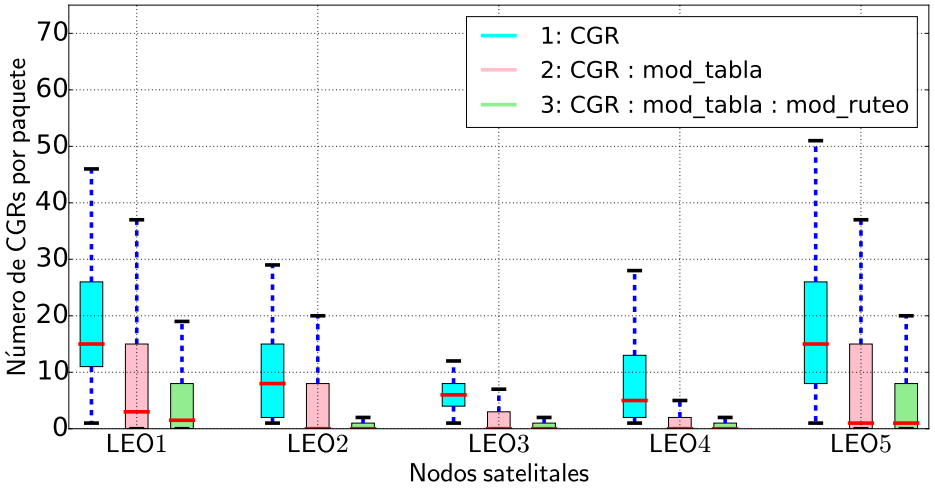

FIGURA 4.10: La modificación de la lógica que utiliza *CGR* para el enrutamiento, como también el plan de contactos permite disminuir el cálculo del algoritmo en los nodos. **1.** Algoritmo *CGR* normal; **2.** *CGR* con la variante de *Totsim* que permite almacenar las rutas en una tabla de enrutamiento para cálculos posteriores; **3.** *CGR* con la variante de *Totsim* para el enrutamiento y la mejora del plan de contactos mediante el algoritmo 3. En estos tres casos todas las imágenes fueron entregadas correctamente.

4.3.4. Distancia de red y tiempo de vida de los paquetes

En este apartado se evalúa la cantidad de saltos y el tiempo de vida que un paquete necesitó para ser enviado desde el nodo emisor hacia alguna estación terrena. Para el análisis de este punto, fue necesario calcular previamente la cantidad de saltos y el tiempo de vida de cada paquete generado en la simulación. En la figura 4.11 podemos observar que:

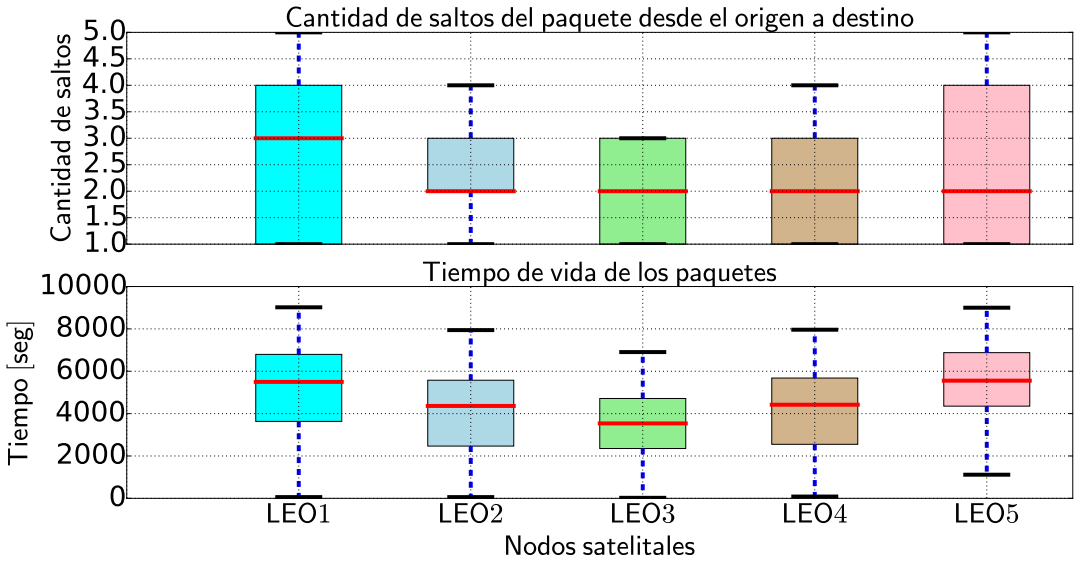

FIGURA 4.11: La cantidad de saltos para los nodos *LEO1* y *LEO5* es de 1 a 5 saltos, debido a que están localizados en los extremos de la constelación. El 50 % de la cantidad de datos se ubican por debajo de los dos saltos para todos los nodos con excepción de *LEO1*. Por otra parte, el rango de tiempos de vida de los paquetes está por debajo de los 10.800 segundos que fue el *TTL* configurado (ver tabla 4.2).

i. Los valores mínimos y máximos se encuentran entre 1 a 5 saltos para *LEO1* y *LEO5*, esto se debe a que estos nodos se localizan en los extremos de la constelación, y por lo tanto necesitan más nodos intermedios para descargar sus paquetes.

ii. El tiempo de vida de los paquetes en cada uno de los nodos varía desde un valor mínimo de 12 segundos a un máximo de 9.000 segundos. El 50 % de los datos se ubican por debajo de 5.800 segundos para los nodos *LEO5* y *LEO1*, 4.050 a 4.100 segundos para la segunda y cuarta estación satelital, y finalmente para *LEO3* por debajo de los 3.900 segundos. Por otra parte, todos estos valores están por debajo de los 10.800 segundos que fue el *TTL* configurado.

iii. Al igual que el análisis realizado para el número de saltos, los nodos ubicados en los extremos presentan un mayor rango de tiempos de vida en comparación a los que se localizan en la mitad del tren de satélites.

4.3.5. Trayectorias de los paquetes desde el nodo origen hacia la estación terrena

Tal como se describió en la sección 3.3.1, existe la posibilidad de que un nodo utilice otras estaciones para bajar sus imágenes. Por consiguiente, se implementaron algunas funciones que permiten graficar la trayectoria de un paquete desde el nodo origen hacia la estación terrena como se visualiza en la figura 4.12. En este caso, *CGR* para un paquete generado en el nodo *LEO5* calculará una ruta considerando una serie de contactos temporales en donde el nodo *LEO1* será utilizado como nodo de descarga de datos.

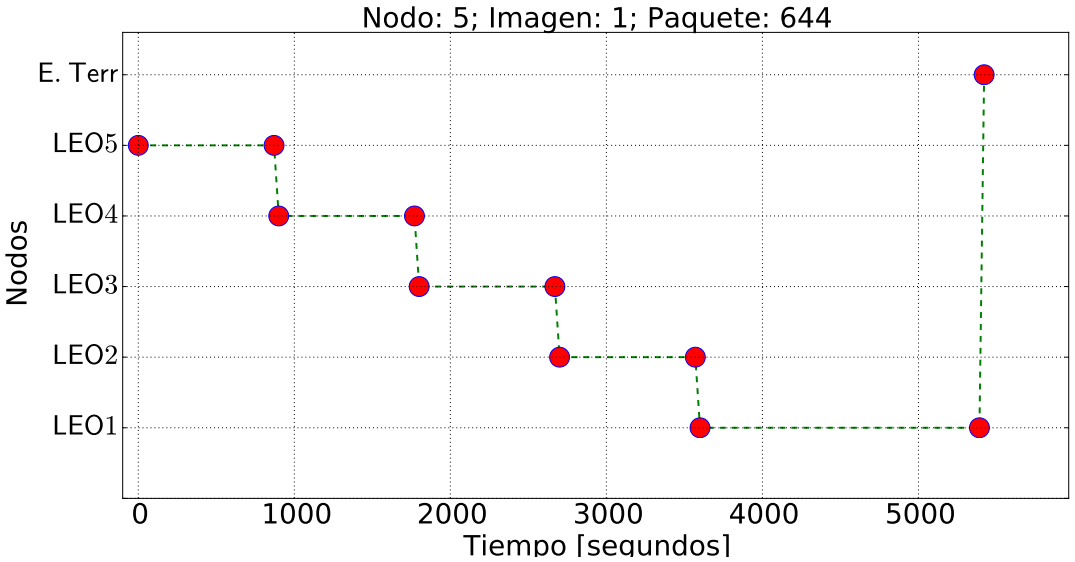

FIGURA 4.12: Recorrido del paquete 644 desde el nodo *LEO5* hacia la estación terrena. El plan de contactos configurado determina que el nodo que posee una oportunidad para descargar información en alguna estación terrena es el nodo *LEO1*, debido a ello *CGR* calcula una ruta a la estación terrena en donde el nodo de último salto es *LEO1*. Estos datos fueron generados con los valores de la tabla 4.2.

4.4. Escenario 2: Límite en la capacidad de almacenamiento en los nodos.

El objetivo de este escenario es verificar que el algoritmo *CGR* calcula las rutas en función a la capacidad residual de los contactos y el tiempo de vida de los paquetes, sin considerar el estado en el que se encuentran los nodos a los cuales se les va a enviar información. En otras palabras, *CGR* conoce únicamente el estado de ocupación de los enlaces desde un punto de vista local, y no sabe de qué manera los otros nodos están utilizando los recursos por donde va a cursar información. En este contexto, se ha limitado la capacidad de almacenamiento de las estaciones espaciales a 100 MB (ver tabla 4.5). Adicionalmente, se ha incrementado el tamaño de la imagen a 20 MB, mientras que los demás valores se mantienen con respecto al Escenario 1.

Con los valores de configuración de la tabla 4.5, cada nodo genera 1,46 GB de datos, y un tráfico de información total en los nodos de 7,32 GB. Cabe mencionar que se utilizó nuevamente el algoritmo 3 para la configuración del plan de contactos a cargar en los nodos espaciales.

Variable	**Valor configurado**
orbitalTime	90 minutos
datarateTerr	20 Mbps
datarateSat	2 Mbps
bufferSizeTerr	∞
bufferSizeSat*	**100 MB**
simulationTime	67hs 40min
timeSat	15 minutos
timeTerr	10 minutos
cantBundles	686
bundleSize	**29,9 KB**
bundleTTL	10.800 segundos
cantImagen	75
imagenSize	**20 MB**
timeGenImage	3.200 segundos

Tabla 4.5: Variables de simulación para el Escenario 2.
*Con respecto al escenario anterior, la capacidad de almacenamiento en las estaciones satelitales fue limitado a 100 MB.

4.4.1. Cálculo de Contact Graph Routing en los nodos

CGR fue calculado 1.079.129 veces en toda la simulación y éste presenta una distribución de valores similar con respecto al Escenario 1 (ver figura 4.13). Es decir, los nodos localizados al extremo de la constelación calculan un mayor número de veces este algoritmo. Asimismo, los nodos *LEO2*, *LEO3* y *LEO4* emplearon el 50 % de las veces una ruta ya calculada para la transmisión de los paquetes hacia los siguientes nodos debido a que el valor mínimo y los primeros dos cuartiles coinciden en un valor de 0.

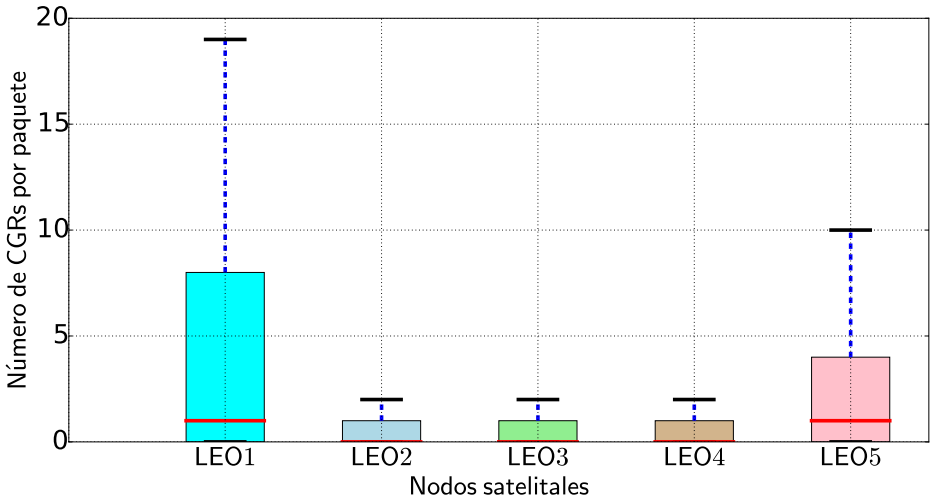

FIGURA 4.13: Diagrama de cajas que representa el cálculo de *CGR* cuando la capacidad de almacenamiento en los nodos es configurada a 100 MB (ver tabla 4.5). Valores de 0 indican que se utilizó una ruta ya calculada para el envío de los paquetes.

4.4.2. Tasa de imágenes y paquetes transmitidos correctamente a la estación terrena

En el Escenario 1 los nodos no superan su capacidad de almacenamiento y por lo tanto no rechazan paquetes entrantes. No obstante, en este escenario esta capacidad es limitada a 100 MB. En este contexto, se transmitieron solamente 26 imágenes completas en el caso del nodo *LEO1*, y un valor máximo de 43 imágenes para *LEO5* tal como se muestra en la figura 4.14. Esto se debe a que no existe un mecanismo que permita informar cuándo los nodos están congestionados. Cabe mencionar que este problema es complejo dado a la conectividad intermitente entre los nodos espaciales.

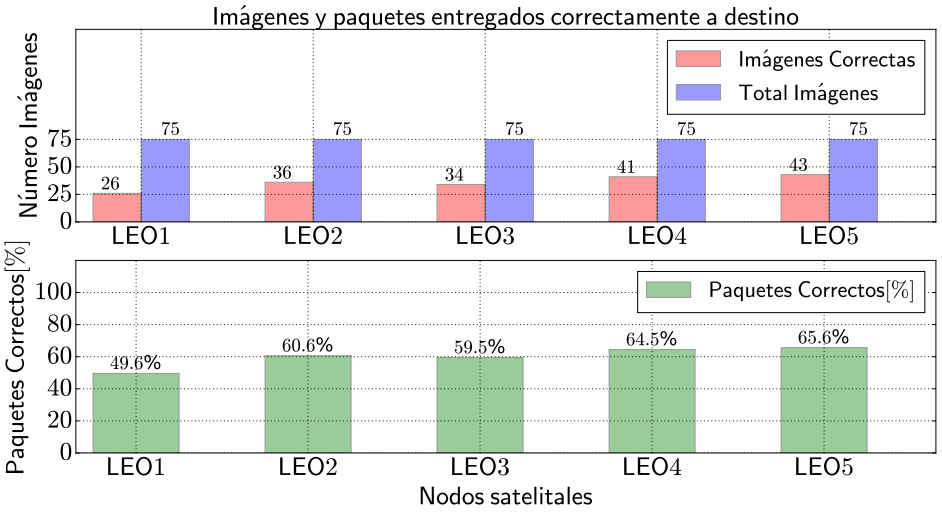

FIGURA 4.14: Al limitar la capacidad de almacenamiento a 100 MB (ver tabla 4.5), provocó que los nodos rechacen paquetes entrantes, cuando su capacidad de almacenamiento es superada, lo que se traduce en una mayor pérdida de información.

En la figura 4.15 se indica que el nodo que descartó la mayoría de los paquetes generados en la simulación a la entrada fue *LEO5* (24,5 %), y por otro lado si observamos la figura de la derecha el mayor porcentaje de éstos corresponden a *LEO2*. Además, *LEO5* no descartó paquetes provenientes de *LEO4*.

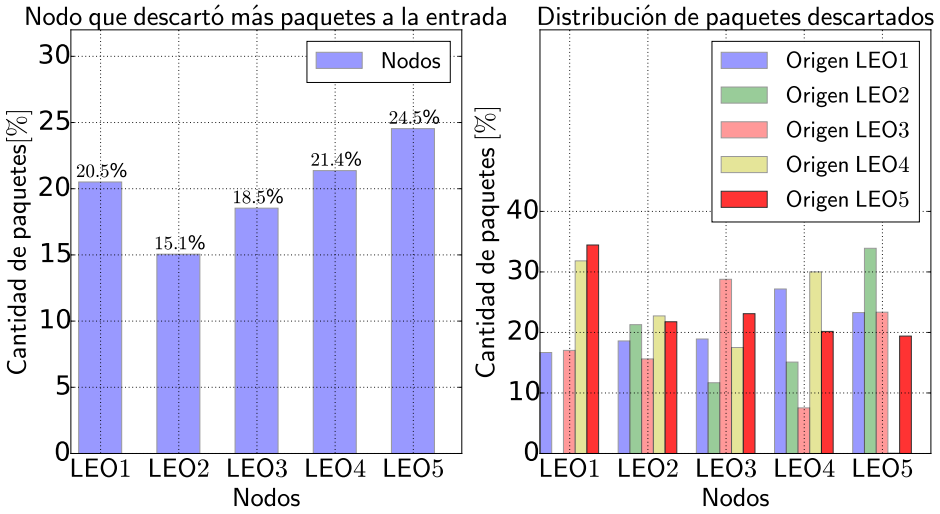

FIGURA 4.15: El nodo *LEO5* descartó la mayor cantidad de paquetes provenientes de los otros nodos. Adicionalmente, la mayoría de información descartada por este nodo corresponde a *LEO2* como se observa en la figura de la derecha. Estos datos fueron obtenidos para los valores de configuración de la tabla 4.5.

4.4.3. Entrega de los paquetes por el nodo de último salto designado

En la figura 4.16 se muestra que el tráfico es distribuido de forma equitativa entre los nodos. Sin embargo, este resultado se debe a que se ha limitado la capacidad de los nodos a un valor fijo de 100 MB. Por otra parte, los nodos *LEO1* y *LEO5* presentan una característica particular; el nodo *LEO1* no bajó ninguna imagen generada por *LEO5*, mientras que *LEO5* entregó solamente el 4 % de los paquetes de *LEO1*. Esta peculiaridad también es analizada en el siguiente apartado.

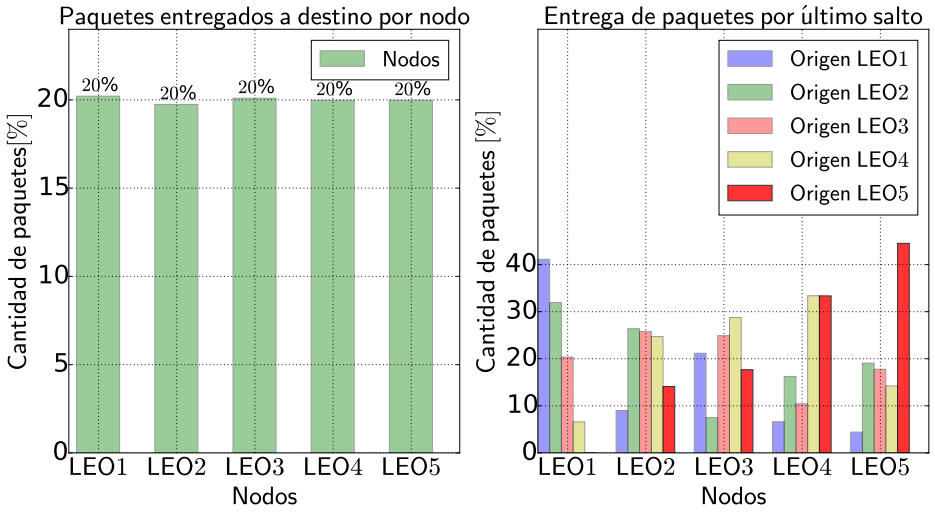

FIGURA 4.16: Un 20 % de los datos generados en la simulacion es distribuido entre cada nodo de último salto. Por otra parte, el nodo *LEO1* no descargó ningún paquete generado por *LEO5* como se observa en la figura de la derecha. Estos datos fueron obtenidos para los valores de configuración de la tabla 4.5.

4.4.4. Distancia de red y tiempo de vida de los paquetes

El análisis para este apartado es similar al realizado para la figura 4.11. No obstante, los valores mínimos de 0 saltos de la figura 4.17 indican que algunos paquetes fueron generados, pero *CGR* no pudo calcular una ruta. Cabe destacar que *LEO5* no presenta valores de 5 saltos (a diferencia de la figura 4.11), y esto se debe a que ningún paquete generado por *LEO5* fue descargado por el nodo de último salto *LEO1*.

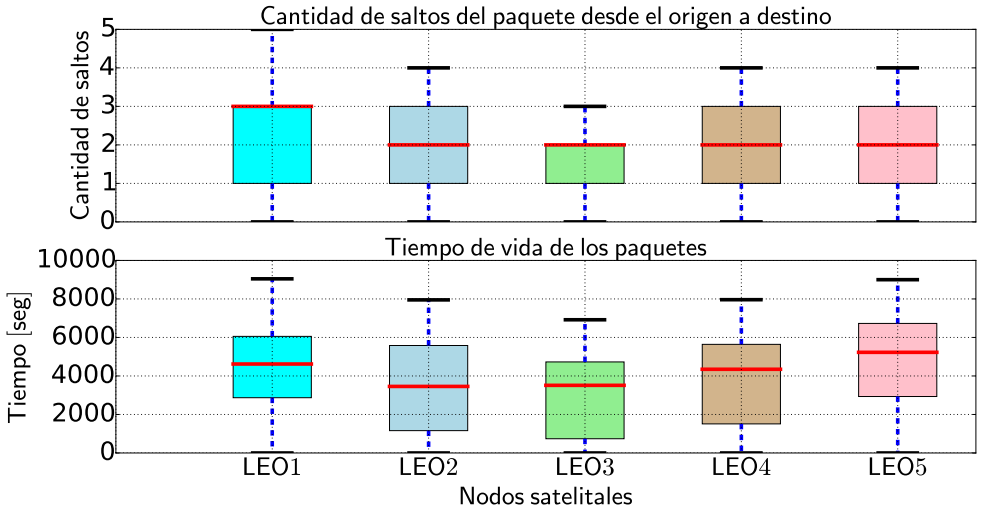

FIGURA 4.17: Distancia de red y tiempo de vida de los paquetes para los valores de configuración de la tabla 4.5. *LEO5* no presenta valores de 5 saltos, y esto se debe a que ningún paquetes generado por *LEO5* fue transmitido a la estación terrena por el nodo de último salto *LEO1*.

4.5. Escenario 3: Sistema al máximo de su capacidad teórica

Es importante definir la capacidad máxima de tráfico a cursar en toda la simulación, en donde las imágenes sean transmitidas con la esperanza de que estas lleguen completas a las estaciones terrenas. Por consiguiente, se ha tomado en cuenta las siguientes consideraciones:

i. Por cada período orbital, solamente un nodo del tren de satélites descarga la información generada por todas las estaciones espaciales en alguna estación terrena. Por otro lado, la tasa de transmisión del enlace de 20 Mbps y el tiempo de conexión entre el nodo de último salto designado y la estación terrena definen un volumen máximo de 1,39 GB para transmitir por cada oportunidad de descarga.

ii. Para un tiempo de simulación de 67 horas y 40 minutos, la constelación de satélites tendrá la oportunidad de descargar sus imágenes alrededor de 45 veces en las

estaciones terrenas[2]. Por consiguiente, en la simulación se generará un total de 62,84 GB de datos.

iii. El tamaño de cada imagen fue calculado en función de dos puntos. En primer lugar, los 62,55 GB de información calculados es distribuido entre las 5 estaciones satelitales, lo que da como resultado que cada nodo sea capaz de generar un total de 12,5 GB de datos en la simulación. En segundo lugar, esta cantidad se reparte entre las 75 imágenes que cada nodo es capaz de trasmitir. Por ende, el tamaño de imagen a utilizar en el presente escenario es de 171 MB. Finalmente, se empleó nuevamente el plan de contactos calculado con el algoritmo 3, el cual descartaba contactos que no influían en el rendimiento de la red.

Variable	**Valor configurado**
orbitalTime	90 minutos
datarateTerr	20 Mbps
datarateSat	2 Mbps
bufferSizeTerr	∞
bufferSizeSat	∞
simulationTime	67hs 40min
timeSat	15 minutos
timeTerr	10 minutos
cantBundles*	**686**
bundleSize	**256 KB**
bundleTTL	10.800 segundos
cantImagen	75
imagenSize	**171 MB**
timeGenImage	3.200 segundos

Tabla 4.6: Variables de configuración para el Escenario 3.
* Para incrementar el tamaño de la imagen se incrementó el tamaño del bundle debido a que se ha mantenido el número de bundles o paquetes por imagen.

4.5.1. Tasa de imágenes y paquetes transmitidos correctamente a la estación terrena

En este escenario se configuraron las variables de simulación en función de los puntos expuestos en el apartado anterior (ver tabla 4.6). Estos valores provocaron que una gran cantidad de imágenes llegarán incompletas tal como se visualiza en la figura 4.18. En este

[2]Es importante mencionar que si se considera una sola estación terrena para este periodo de tiempo en un caso real, los satélites tendrán una frecuencia de conexión con la estación terrena una menor cantidad de veces. Por ello, para tener más oportunidades de descarga por cada período orbital se han considerado más estaciones terrenas.

contexto, el nodo *LEO3* fue el más perjudicado, ya que solamente 7 imágenes fueron enviadas correctamente. Por otra parte, el porcentaje de paquetes correctos recibidos de *LEO5* es mayor a *LEO4*; no obstante, éstos corresponden a una menor cantidad de imágenes completas.

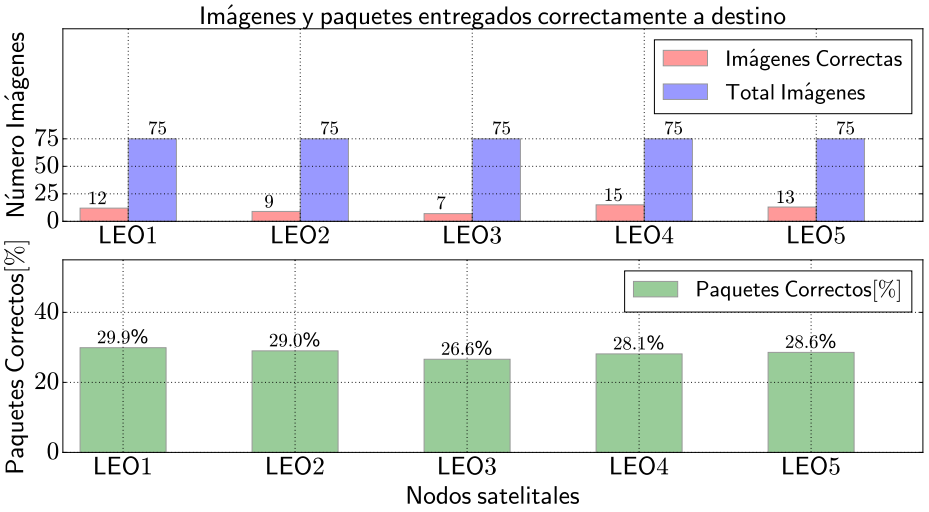

FIGURA 4.18: Al incrementar el tráfico a cursar en los nodos a su valor máximo provocó que una gran cantidad de imágenes y paquetes no lleguen correctamente (ver tabla 4.6).

Para solucionar estos inconvenientes es necesario sintonizar el sistema en función de dos objetivos. En primer lugar, que sean transmitidas el 100 % de las imágenes que son generadas a lo largo de la simulación. Por otro lado, que cada uno de los nodos que cumplen la función de nodo de último salto entreguen el 20 % de la información generada con el menor cálculo de *CGR*.

En el presente escenario se pueden variar dos parámetros para mejorar o empeorar el rendimiento del sistema. Por un lado, el tiempo de conexión entre satélites y por otro las tasas de transmisión en los enlaces satelitales. La variación de estos dos parámetros permite incrementar o disminuir el volumen de contactos a cursar entre los enlaces satelitales (ver sección 2.3.4.1).

4.5.1.1. Ventana de contacto entre satélites

El volumen de contactos que cada estación satelital es capaz de transferir al siguiente nodo está en función de la velocidad de transmisión del enlace y el tiempo de la ventana de contacto. Sin embargo, es importante mantener una relación entre los tiempos de

conexión entre satélites y el período orbital. Su incumplimiento provocaría que algunos nodos no sean capaces de enviar su información al último salto antes de la pasada por la estación terrena (ver figura 4.19). Por ejemplo, si el tiempo de conexión entre satélites se incrementa a 20 minutos y se mantiene que por cada período orbital de 90 minutos existirá la posibilidad de descargar las imágenes desde los nodos en órbita (caso ideal). Esto implica que el nodo *LEO5* que se encuentra en el extremo de la constelación no pueda enviar sus datos a *LEO1* a tiempo cuando este cumple la función de nodo de descarga de datos. Por consiguiente, sus datos serán enviados en la siguiente oportunidad de bajada.

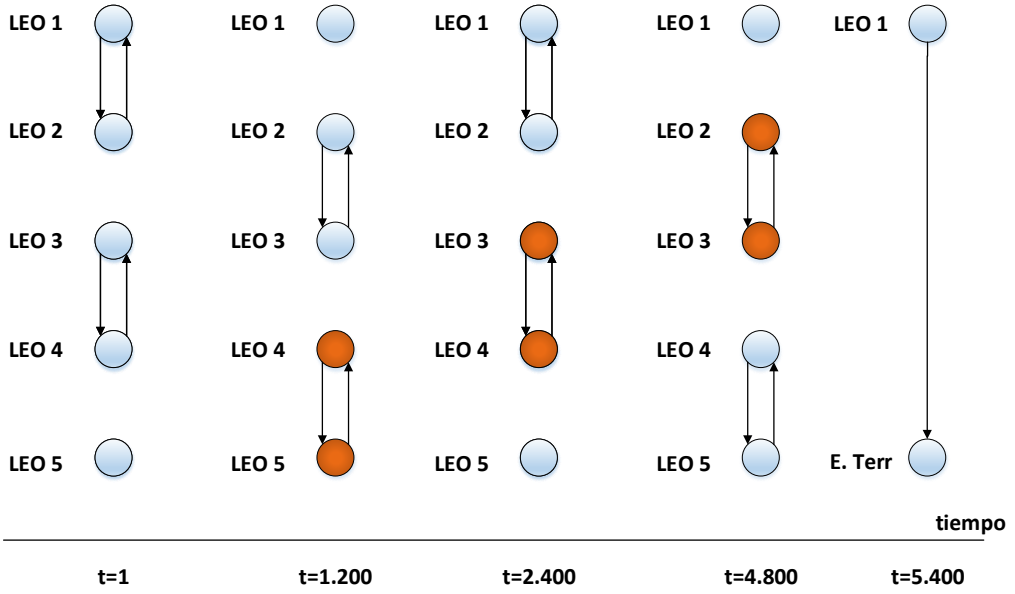

FIGURA 4.19: Configuración del plan de contactos para un tiempo de conexión entre satélites de 20 minutos. Con este valor de tiempo, los datos generados en el nodo *LEO5* no son capaces de llegar a tiempo al nodo *LEO1* que descarga los datos en la estación terrena.

4.5.1.2. Velocidad de transmisión de los enlaces

Para solucionar el inconveniente expuesto en el punto anterior, se decidió realizar 10 simulaciones variando la velocidad de transmisión entre los satélites desde un valor mínimo de 2 Mbps hasta un valor máximo de 20 Mbps (ver tabla 4.7). El objetivo es incrementar el volumen de contactos entre los enlaces satelitales. Es importante mencionar que el valor tope concuerda con la tasa de transmisión de los nodos espaciales con la estación terrena.

datarateSat [Mbps]	**Paquetes Correctos[%]**	**Nodo de último salto [%]**	**Cálculo CGR**
2	**28,4**	**16 a 24**	**672.548**
4	28,4	16 a 23	2.219.387
6	28,4	18 a 22	1.129.386
8	28,4	17 a 22	1.490.493
10	28,4	20	1.662.981
12	28,4	19 a 21	727.877
14	**28,4**	**16 a 25**	**2.790.543**
16	28,4	18 a 22	851.018
18	28,4	16 a 24	2.054.341
20	28,4	20	1.326.867

Tabla 4.7: La estación terrena recibió el mismo porcentaje de paquetes correctos como resultado en todas las simulaciones. Sin embargo, la forma como estos paquetes fueron distribuidos en los nodos de último salto presentó diferentes variaciones, al igual que el cálculo de *CGR*.

Tal como se indica en la tabla 4.7 fueron recibidos el mismo porcentaje de paquetes correctos en todas las simulaciones efectuadas. Sin embargo, la forma como estos paquetes fueron distribuidos en los nodos de último salto presentó diferentes variaciones. Por ejemplo, en la simulación realizada con una velocidad de transmisión de 14 Mbps, un nodo en particular entregó solamente el 16 % de la información, mientras que el nodo que fue mayormente utilizado por las otras estaciones como nodo de descarga de datos envió la cuarta parte de paquetes generados. Adicionalmente, en dicha simulación se calculó la mayor cantidad de veces *CGR*.

Debido a estos resultados, se ha decidido disminuir el tamaño de las imágenes dado que no se consiguió una respuesta del sistema favorable al variar la velocidad de transmisión en los enlaces entre satélites. El objetivo de estas nuevas simulaciones es determinar el tamaño de la imagen que permita enviar todas las imágenes completas (ver tabla 4.8). Se encontró que con un tamaño de imagen de 44,6 MB (26 % de la imagen original) se consiguen estos resultados. Sin embargo, el número de veces que es calculado *CGR* juega un factor importante a considerar, ya que existen varias simulaciones que indican el mismo tamaño de imagen óptimo, pero con mayores cómputos de *CGR* que otros. Dicho esto, el escenario que dió mejores resultados desde el punto de vista técnico fue la simulación realizada con una velocidad de transmisión de 16 Mbps. En este punto hay que hacer hincapié que si bien por cuestiones económicas colocar un enlace de 4 Mbps es de menor costo que un enlace de 16 Mbps; no obstante, desde el aspecto técnico analizado en este trabajo un enlace de 4 Mbps requiere mayores recursos de procesamiento debido

a un mayor cálculo de *CGR*.

datarateSat [Mbps]	**imagenSize [MB]**	**Nodo de último salto [%]**	**Cálculo de CGR**
2	37,7	18 a 22	587.423
4	**44,6**	**18 a 21**	**3.387.742**
6	42,9	18 a 22	511.337
8	44,6	18 a 21	2.867.348
10	17,2	18 a 22	1.225.933
12	44,6	18 a 21	727.954
14	44,6	18 a 21	3.467.499
16	**44,6**	**18 a 21**	**484.849**
18	44,6	18 a 21	3.310.832
20	30,9	18 a 22	708.612

TABLA 4.8: La segunda columna representa el nuevo tamaño de imagen para el cual todas las imágenes generadas en la simulación sean entregadas de forma correcta. El objetivo es enviar la imagen con el mayor tamaño posible y con el menor cómputo de *CGR* en la simulación.

4.5.2. Cálculo de Contact Graph Routing en los nodos

Se ejecutó una nueva simulación con los valores de configuración descritos en la tabla 4.9 en donde se modificó el tamaño de la imagen a 44,6 MB. Con estos nuevos valores se consiguió que *CGR* sea ejecutado solamente el 25 % de la veces en todos los nodos. Esto lo podemos verificar en la figura 4.20, ya que el valor mínimo y los tres cuartiles coinciden con un valor de 0 en todas las estaciones. Por otro lado, las imágenes fueron recibidas de forma correcta tal como se indica en la figura 4.21.

4.5.3. Entrega de los paquetes por el nodo de último salto designado

Los paquetes generados en toda la simulación fueron distribuidos casi de forma homogénea entre todos los nodos de último salto configurados, ya que los valores fluctúan entre el 18 % al 21 % del 20 % que es el valor ideal para que todos los nodos tengan el mismo índice de utilización a lo largo de la simulación como se observa en la figura 4.22.

Variable	Valor configurado
orbitalTime	90 minutos
datarateTerr	20 Mbps
datarateSat	**16 Mbps**
bufferSizeTerr	∞
bufferSizeSat	∞
simulationTime	67hs 40min
timeSat	15 minutos
timeTerr	10 minutos
cantBundles	686
bundleSize	**67 KB**
bundleTTL	10.800 segundos
cantImagen	75
imagenSize	**44,6 MB**
timeGenImage	3.200 segundos

Tabla 4.9: Variables de configuración para el Escenario 3 donde se modificó el tamaño de imagen de 171 MB a 44,6 MB.

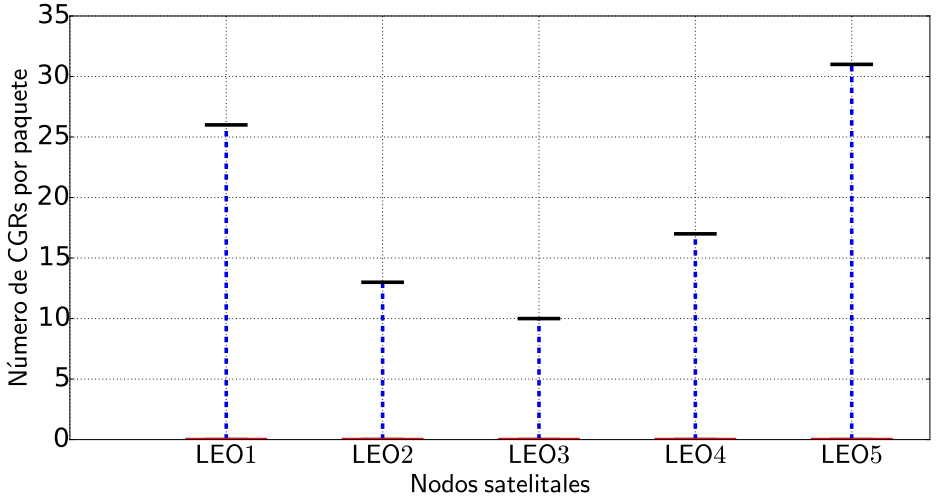

Figura 4.20: Con los valores de configuración de la tabla 4.9, *CGR* es ejecutado solamente el 25 % de las veces, ya que la mayoría de las veces se utiliza una ruta previamente calculada para enviar el paquete de análisis. El valor mínimo y los tres cuartiles convergen en un valor de 0 para todos los nodos.

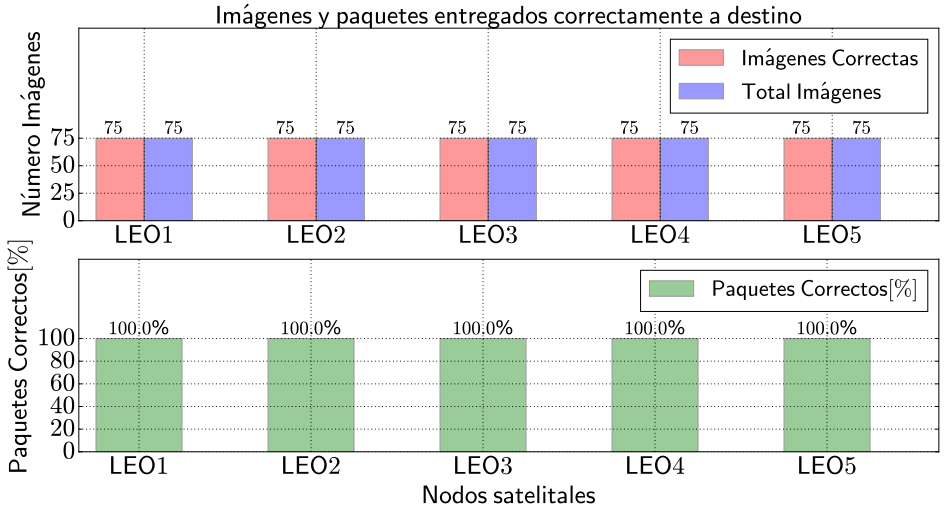

FIGURA 4.21: Al modificar el imagenSize=44,6 MB y el datarateSat=16 Mbps se consiguió que todas las imágenes lleguen de forma correcta en la estación terrena y con el menor computo de *CGR* posible (ver tabla 4.9).

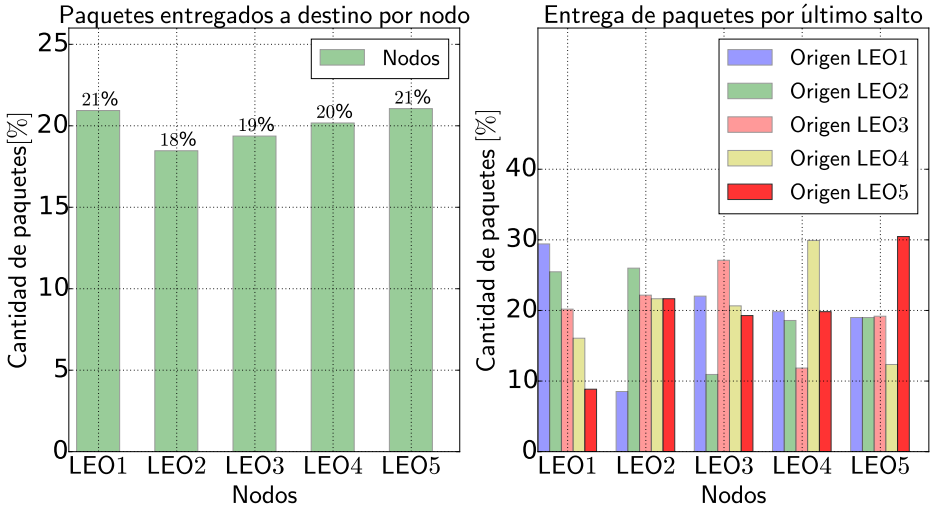

FIGURA 4.22: Entrega de los paquetes por el nodo de último salto designado para los valores de configuración de la tabla 4.9. Cada nodo entrega del 18 % al 21 % de la información generada en toda la simulacion que son valores cercanos al 20 % que es el valor ideal para una utilización equitativa entre los nodos.

4.6. Escenario 4: Pérdida intermitente de contactos entre enlaces.

En este apartado se ha decidido dar una probabilidad de pérdida de contactos entre los satélites y estos con las estaciones terrenas. Adicionalmente, en todos los escenarios se estableció que la probabilidad de pérdida de contactos entre los nodos y las estaciones terrenas es menor que la probabilidad de pérdida entre satélites tal como se visualiza en la tabla 4.10. Por otro lado, se han empleado los valores de configuración obtenidos del escenario anterior con el objetivo de evaluar cómo estos cambios afectan en el rendimiento de la red (ver tabla 4.11).

Escenario	**probTerr***	**probSat****
Escenario 3	0	0
Escenario 4.1	0,05	0,15
Escenario 4.2	0,10	0,20
Escenario 4.3	0,15	0,25

Tabla 4.10: Probabilidad de pérdida de contactos entre nodos.
***Probabilidad de perder el contacto entre la estación satelital y la estación terrena.
****Probabilidad de perder contactos entre las estaciones satelitales.

Variable	**Valor configurado**
orbitalTime	90 minutos
datarateTerr	20 Mbps
datarateSat	**16 Mbps**
bufferSizeTerr	∞
bufferSizeSat	∞
simulationTime	67hs 40min
timeSat	15 minutos
timeTerr	10 minutos
cantBundles	686
bundleSize	**67 KB**
bundleTTL	10.800 segundos
cantImagen	75
imagenSize	44,6 MB
timeGenImage	3.200 segundos

Tabla 4.11: Variables de configuración para el Escenario 4.

Adicionalmente, se empleó el plan de contactos completo (ver tabla 4.12), es decir no se utilizó el plan de contactos modificado con el algoritmo 3, el cual eliminaba contactos innecesarios que no influían en el rendimiento de la red.

Por cada escenario se ejecutaron 10 simulaciones, y de éstas, mediante *Python*, se obtuvieron los siguientes datos: por un lado, el último nodo que fueron encaminados los

Enlace	Cantidad de contactos
Nodos ↔ Estaciones terrenas *	9
LEO1 ↔ LEO2	113
LEO2 ↔ LEO3	113
LEO3 ↔ LEO4	113
LEO4 ↔ LEO5	113

Tabla 4.12: Ventanas de contacto entre nodos.
***** Cada nodo tiene 9 oportunidades de contacto para enviar la información generada en todas las estaciones satelitales a las estaciones terrenas. Estos valores fueron calculados para un caso ideal en donde por cada vuelta por tierra cada nodo tendrá la posibilidad de tener conexiones con alguna estación terrena.

paquetes que no pudieron ser transmitidos debido a la pérdida de contactos entre enlaces; por otro lado, la cantidad de contactos perdidos entre enlaces, y éstos con su estación terrena receptora. Finalmente, el porcentaje de paquetes entregados correctamente por el nodo que baja las imágenes.

Toda esta información obtenida es analizada como un conjunto de datos mediante diagramas de caja. En la figura 4.23 se visualiza que conforme la probabilidad de pérdida de contactos es incrementada en cada escenario, influye para que una mayor cantidad de paquetes no puedan ser entregados. Cabe destacar que en el *Escenario 4.1 (probTerr=0,05 y probSat=0,15)* el valor de la mediana en todos los nodos converge casi al mismo valor. Sin embargo, en los otros escenarios que tienen mayor probabilidad de pérdidas de contactos esto no sucede, y lo que se observa es que los nodos ubicados en los extremos son los que almacenan la mayor cantidad de paquetes que no pudieron ser enviados al siguiente salto. Esto se debe a dos razones:

i. Los nodos *LEO1* y *LEO5* tienen una estación espacial vecina a la cual ellos pueden enviar información. Por lo tanto, si pierden un contacto con su vecino respectivo, estas estaciones no tendrán otra alternativa para transmitir sus datos. No obstante, esto no sucede con las estaciones ubicadas al centro del tren de satélites que tienen dos vecinos por donde enviar sus imágenes.

ii. Sin embargo, es importante mencionar que la lógica de *CGR* no permite que los paquetes sean enviados de vuelta para atrás cuando se encuentran con un corte en el camino. Esto se debe a que el algoritmo guarda en la lista de nodos excluidos a los nodos por donde recibió la información con el objetivo de evitar bucles de enrutamiento (ver sección 2.3.5.1).

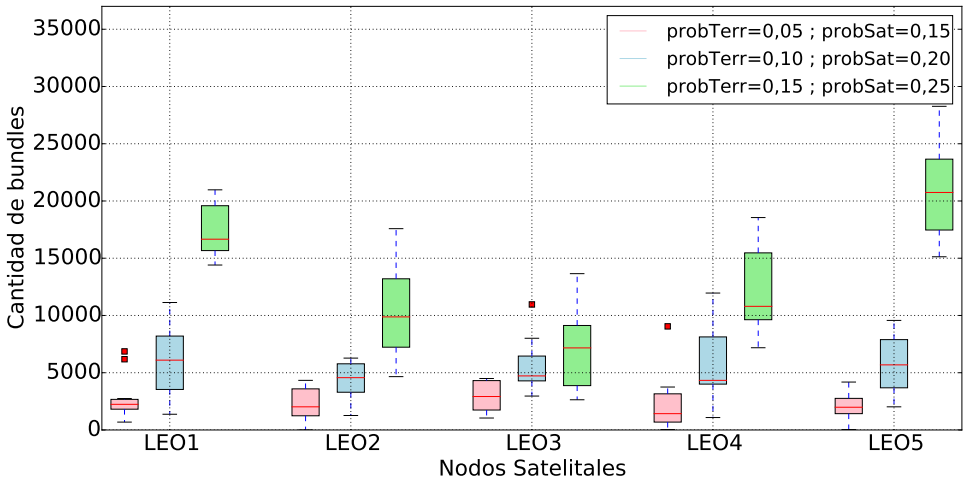

FIGURA 4.23: Esta gráfica representa la cantidad de paquetes que fueron almacenados en los nodos, ya que no fueron entregados al destino dado a la pérdida de contactos entre enlaces. Claramente al incrementar la probabilidad de pérdida entre contactos influye a que una mayor cantidad de paquetes no sea entregada al siguiente nodo vecino.

Por otra parte, se realizó un nuevo conjunto de 10 simulaciones en función del plan de contactos calculado con el algoritmo 3, el cual eliminaba algunos contactos entre enlaces que no influían en el rendimiento de la red. Como resultado, la figura 4.24 muestra que la distribución de los datos fue similar. Es decir, al aumentar la probabilidad de pérdida entre contactos, los nodos de los extremos son los que almacenan la mayor cantidad de paquetes que no fueron enviados.

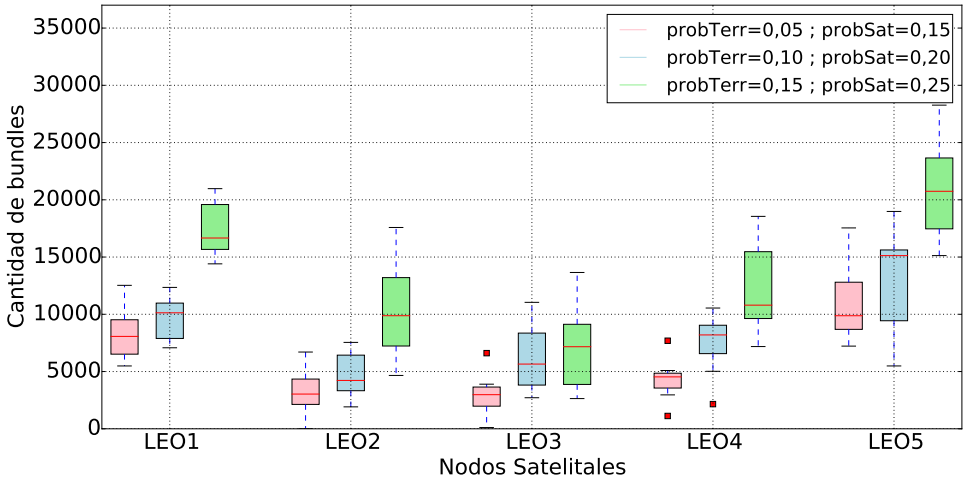

FIGURA 4.24: Al emplear el plan de contactos modificado con el algoritmo 3, se obtuvo una distribución similar de valores con respecto a la figura 4.23 en donde los nodos *LEO1* y *LEO5* almacenan la mayor cantidad de bundles que no fueron enviados.

4.7. Escenario 5: Plan de contactos calculado con parámetros orbitales

En este escenario se ha empleado un plan de contactos calculado en función de los parámetros orbitales indicados en la tabla 4.13. Dicho plan de contactos está optimizado con programación lineal, en donde no sólo se consideran las oportunidades de contacto entre nodos sino también el tráfico a cursar (imágenes) [46]. Por otro lado, a diferencia de los escenarios anteriores, se ha considerado en este apartado un tren de 4 satélites donde cada nodo alterna el enlace de descarga con una estación terrena. Donde el rango de comunicación para los enlaces entre satélites es de 1.000 km mientras que el enlace con la estación terrena es de 2.000 km.

Intervalo de inicio de la topología	1-Enero-2015, 0hs 0min 0s
Intervalo de finalización de la topología	1-Enero-2015, 2015, 21hs 43min 18sec
Bstar Coefficient (/ER)	90.039
Inclination (deg)	98º
RAAN (deg)	0º
Eccentricity	9.152
Argument of Perigee (deg)	0º, 5º, 10º and 15º
Mean Anomaly (deg)	0º
Mean Motion (rev/day)	15,07561758

Tabla 4.13: Parámetros orbitales empleados para generar el plan de contactos de un tren de cuatro estaciones satelitales. [46]

En la figura 4.25 se indica la topología del plan de contactos calculado y que va a ser cargado posteriormente en cada uno de los nodos [46]. Como se observa en dicha figura, para un tiempo aproximado de 24 horas cada nodo logra tener una buena oportunidad de bajada de datos en la estación terrena. Además, el plan de contactos está diseñado para que entre pasada y pasada exista posibilidad de intercambio en órbita de datos. En particular, las ventanas de contacto entre enlaces satelitales son más grandes cuando las estaciones *LEO1* y *LEO3* cumplen la función de nodo de descarga de datos en la estación terrena.

Por otra parte, la tabla 4.14 describe las variables de configuración utilizadas para generar la simulación en *OMNeT++*, en donde cada nodo generará 24 imágenes ya que el tiempo de simulación es menor al de los escenarios anteriores.

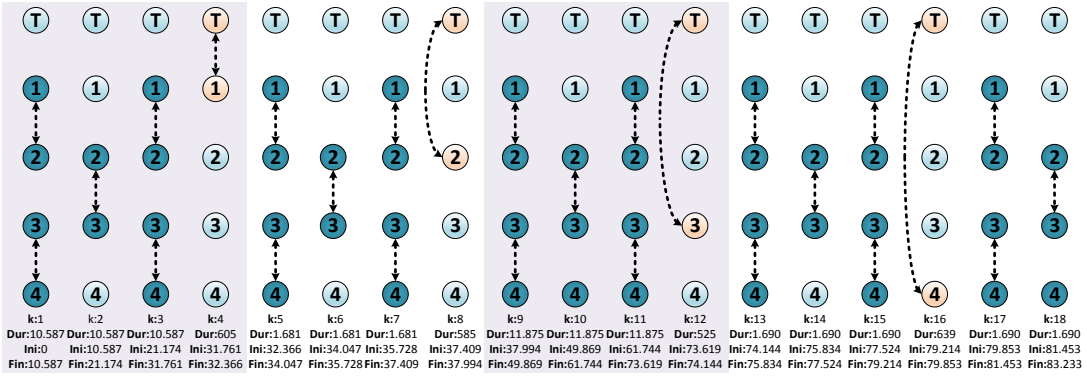

FIGURA 4.25: Topología del plan de contactos calculado con los valores de la tabla 4.13. El índice k representa las oportunidades de comunicación entre los nodos, además los tres siguientes valores indican la duración y los tiempos de inicio y fin de cada una de las ventanas de contacto. Como se indica en la presente gráfica para un tiempo aproximado de 24 hs todos los nodos logran tener una buena oportunidad de descarga de datos en la estación terrena. Adicionalmente, el plan de contactos está diseñado para que entre pasada y pasada exista posibilidad de intercambio de datos en órbita. [46]

Variable	**Valor configurado**
datarateTerr	20 Mbps
datarateSat	16 Mbps
bufferSizeTerr	∞
bufferSizeSat	∞
cantImagen	24
imagenSize	10 MB
cantBundles	686
bundleSize	14,9 KB
timeGenImage	3.200 segundos

TABLA 4.14: Variables de configuración empleadas en el Escenario 5.

4.7.1. Cálculo de Contact Graph Routing en los nodos

En este apartado analizaremos el número de veces que fue calculado el algoritmo de enrutamiento al emplear el plan de contactos generado en función de los parámetros orbitales de la tabla 4.13. Por un lado, el algoritmo fue calculado 720.492 veces cuando no se emplea una tabla de enrutamiento. No obstante, al utilizar la variante de *CGR* que permite reutilizar las rutas ya calculadas se consiguió reducir el cálculo en un 85,02 % (107.872 veces). Cabe recalcar que estos resultados son mejores a los conseguidos en el Escenario 1 donde el cómputo se disminuyó en un 63,71 %. La figura 4.26 muestra el cálculo del algoritmo en cada una de las estaciones espaciales para las dos variantes de *CGR* mencionadas anteriormente. En particular, los nodos *LEO2* y *LEO3* utilizan el

75 % de las veces una ruta ya calculada para el envío de los paquetes, ya que el valor mínimo y los tres cuartiles coincide con el valor de 0.

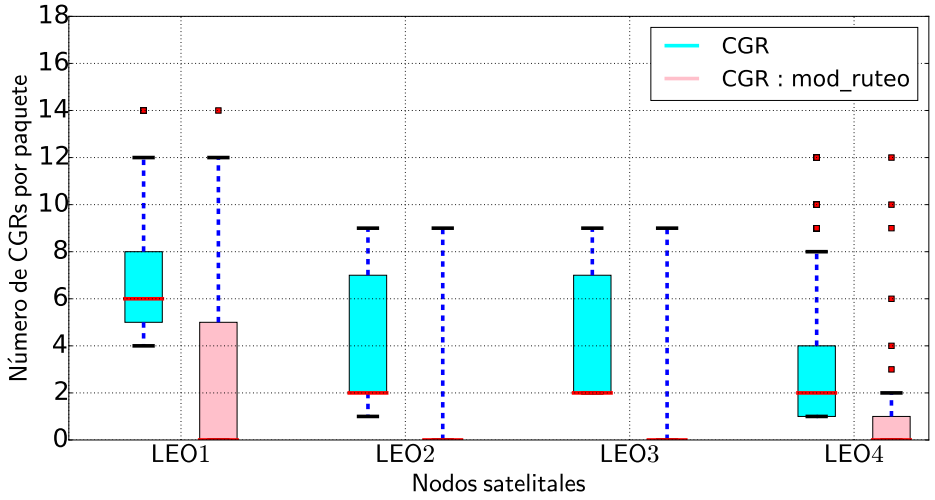

FIGURA 4.26: Esta figura representa el número de veces que fue calculado *CGR* al utilizar el plan de contactos generado con la tabla 4.13. Al reutilizar las rutas para el envío de nuevos paquetes se consigue reducir el cálculo del algoritmo en cada uno de los nodos como se observa en las diagramas de caja de color rosado que representa la variante de *CGR* que utiliza almacenamiento de rutas.

4.7.2. Tasa de imágenes y paquetes transmitidos correctamente

Tal como se indica en la figura 4.27 casi todas las imágenes se transmitieron correctamente a la estación terrena a excepción del nodo *LEO1*. De este nodo llegaron completas 23 de las 24 imágenes generadas. Además mediante las funciones implementadas en *Python* (ver sección A.3) [45] se verificó que no se transmitió un paquete de los 686 totales que componen la imagen número 11 de dicho nodo.

4.7.3. Entrega de los paquetes por el nodo de último salto designado

En este apartado se analiza qué porcentaje de la información generada en la simulación es entregada por cada nodo. Dado que este escenario consta de un tren de cuatro satélites se espera que el 25 % de datos generados por las estaciones sea transmitida a la estación terrena por cada nodo. Sin embargo, esto no sucede ya que cuando el nodo *LEO1* y *LEO3* descargan los datos en las estación terrena los tiempos de las ventanas de contactos entre enlaces son más grandes con respecto a los otros nodos (ver figura 4.25). Como resultado

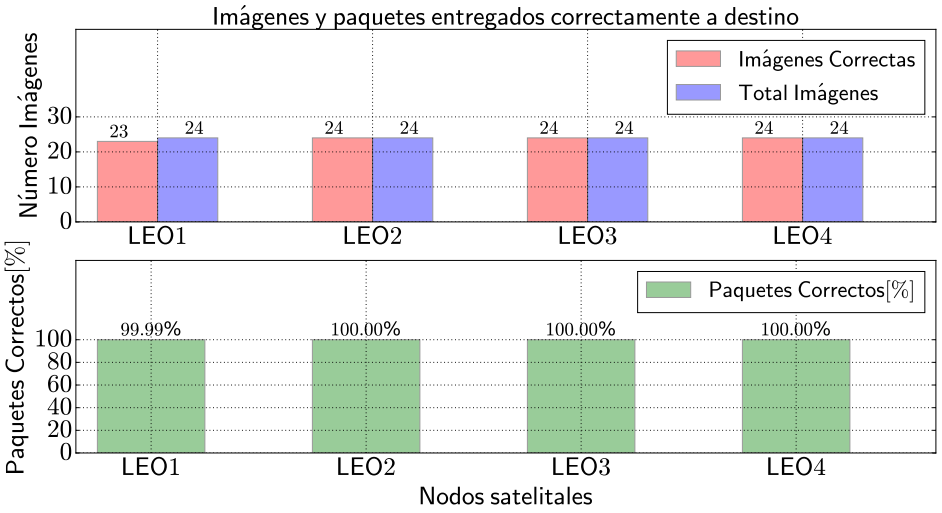

FIGURA 4.27: Con las variables de configuración de la tabla 4.14 se consiguió que casi todas las imágenes sean transmitidas correctamente. En particular el nodo *LEO1* no transmitió un paquete de los 686 totales de la imagen 11, por ello este nodo sólo pudo transmitir 23 imágenes completas a la estación terrena.

existe un mayor tráfico de datos hacia los nodos de último salto *LEO1* y *LEO3* tal como se indica en la figura 4.28. Por otra parte, en la figura de la derecha se observa que los nodos *LEO1* y *LEO4* transmitieron la mayor cantidad de información de *LEO1*, mientras que *LEO2* y *LEO3* descargaron a la estación terrena en su mayoría paquetes de *LEO4*.

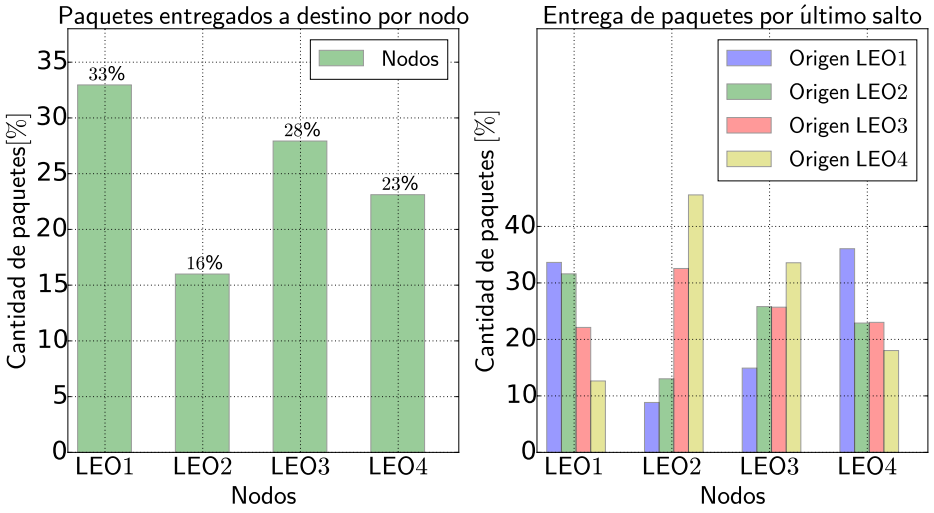

FIGURA 4.28: Entrega de los paquetes por el nodo de último salto designado. Debido a que las ventanas de contactos entre enlaces satelitales son mayores cuando los nodos *LEO1* y *LEO3* descargan la información en la estación terrena. Esto provoca a que dichos nodos reciban la mayor cantidad de información que debe ser descargada. Por otra parte, en la figura de la derecha se observa que los nodos *LEO1* y *LEO4* transmitieron la mayor cantidad de información de *LEO1*.

4.7.4. Distancia de red y tiempo de vida de los paquetes

Al igual que en los escenarios anteriores, los paquetes generados en los nodos que se ubican en los extremos de la constelación necesitan un mayor cantidad de saltos para ser transmitidos hacia la estación terrena (ver figura 4.29). En particular, el nodo *LEO1* exhibe un valor mínimo de 0 saltos que hace referencia al paquete que no fue transmitido de la imagen 11. Finalmente, en la gráfica inferior se indica los tiempos de vida de los paquetes los cuales están por debajo de los 40.000 segundos debido a que el plan de contactos utilizado emplea ventanas de contacto entre satélites mayores a la de los escenarios anteriormente analizados.

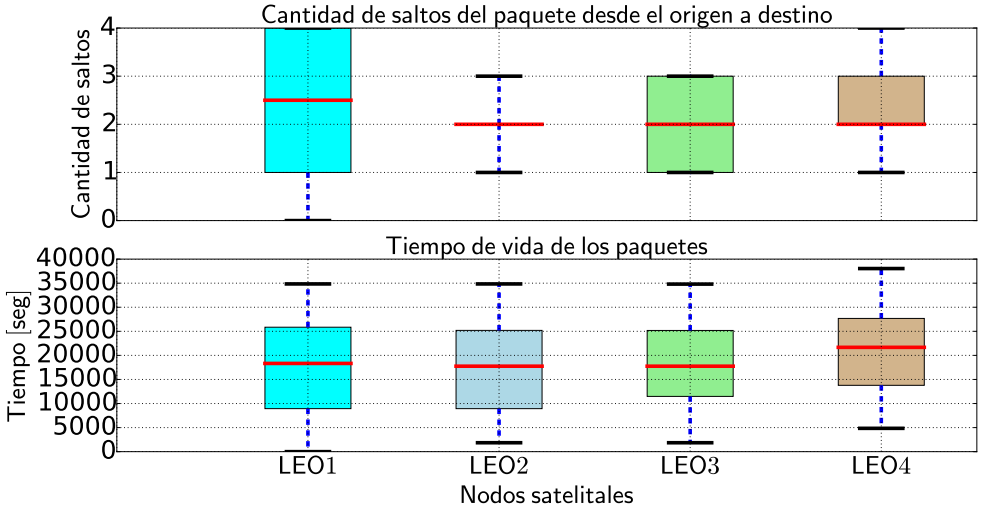

FIGURA 4.29: Los paquetes generados por *LEO1* y *LEO4* requieren una mayor cantidad de saltos para ser transmitidos en la estación terrena. La gráfica localizada en la parte inferior indica que el tiempo de vida de los paquetes esta por debajo de los 40.000 segundos ya que las ventanas de contactos entre satélites son mayores a la de los escenarios anteriormente analizados.

Capítulo 5

Conclusiones

5.1. Conclusiones

En el presente trabajo se evaluó el rendimiento del protocolo *Contact Graph Routing* en comunicaciones por satélite. Para ello se analizaron las modificaciones que deben realizar los esquemas tradicionales de *TCP/IP* para ofrecer sus servicios en un entorno espacial. Por otro lado se expuso la arquitectura *DTN* como esquema genérico para entornos de red desafiantes. Se explicó en detalle la implementación del algoritmo *CGR* en *OMNeT++*, y las diferentes pautas a considerar para generar un plan de contactos óptimo. Finalmente se plantearon cinco escenarios que permitieron analizar las características más importantes de *CGR* en dichos escenarios. Estas conclusiones se sintetizan en los siguientes puntos:

i. Algunos algoritmos de enrutamiento han sido formulados para la arquitectura *DTN*. No obstante, la aleatoriedad con respecto a los contactos no permite que sean aplicados de manera correcta en un entorno espacial donde el volumen de tráfico y las oportunidades de contactos son programadas de antemano (ver sección 2.2.2).

ii. *CGR* tiene un conocimiento completo de la red en base a un plan de contactos de eventos, el cual contiene todas las oportunidades de contactos de los nodos presentes en la red (ver figura 2.4). Esta información permite el cómputo de las rutas en función de dos algoritmos (ver algoritmos 1 y 2). El primero calcula un listado de nodos vecinos de un salto capaces de enviar el paquete a destino,

mientras que el segundo ordena este conjunto de nodos en función de una métrica con el objetivo de obtener el mejor vecino para ese paquete.

iii. *CGR* por cada paquete calcula la ruta completa al destino, en otras palabras no maneja una tabla de enrutamiento (ver figura 2.5). Caini and Firrincieli [40] hacen hincapié en que el tamaño de un bundle es mucho mayor a un paquete *IP*, y por lo tanto esta característica de *CGR* no afecta en mayor medida el rendimiento de la red, ya que para enviar la información se requiere una pequeña cantidad de bundles. Sin embargo, se empleó la variante de *CGR* programada de *Totsim* que permite almacenar rutas que ya fueron previamente calculadas, las cuales son recalculadas cuando el contacto no tiene suficiente capacidad residual ó las ventanas de contacto han expirado. En la sección 4.3.1 se demostró que el cálculo de *CGR* se redujo en un 63,71 % al emplear un esquema donde se almacenan las rutas previamente calculadas (ver figuras 4.4 y 4.5), ya que no necesita recálculo porque no hubo modificaciones.

iv. La configuración del plan de contactos influye en el desempeño de la red. Para ello en la sección 4.3.3 se ejecutó una modificación al plan de contactos con el propósito de optimizar la cantidad de contactos temporales que cada nodo tiene con el nodo de último salto designado. Esta acción permite una reducción del tamaño del plan de contactos en un 36,2 % (ahorro energético). Por consiguiente, los enlaces entre satélites permanecen desconectados por mayores periodos de tiempo sin influir en el desempeño de la red, ya que no se registraron pérdidas de paquetes (ver figura 4.10).

v. En el primer escenario se asumió que la capacidad de almacenamiento en los nodos es lo suficientemente grande como para que los paquetes entrantes no sean descartados cuando la capacidad de almacenamiento es superada. No obstante, en el Escenario 2 se demostró que al limitar la capacidad de almacenamiento de los nodos a 100 MB implicó que una gran cantidad de paquetes fuesen eliminados, como se ilustró en la figura 4.14. Esto se debe a que *CGR* no tiene un conocimiento del estado en el que se encuentran los nodos a los cuales va a enviar información y además no existe un mecanismo que advierta a los demás nodos cuándo éstos están congestionados. En otras palabras, *CGR* conoce únicamente el estado de ocupación de los enlaces desde un punto de vista local, y no sabe de qué manera los otros nodos están utilizando los recursos que va a emplear para enviar información.

vi. El objetivo del Escenario 3 fue encontrar un valor máximo de tráfico, el cual puede ser cursado por los nodos sin que éstos sean eliminados, y que además permita que cada uno de los nodos envíe el 20 % de la información generada en la simulación. En un principio, se exigió al sistema a su máxima capacidad teórica, es decir se calculó el tamaño de las imágenes (171,5 MB) en función del volumen de contactos máximo que puede transmitir el nodo de último salto a las estaciones terrenas por cada pasada (Caso Ideal). Como resultado, se observó que una gran cantidad de imágenes llegaron incompletas a la estación terrena (ver figura 4.18), por lo cual mediante algunas simulaciones se encontró que el sistema presenta el mejor desempeño cuando trabaja al 26 % de su capacidad teórica (ver sección 4.5.1.2). Para este valor todas las imágenes (44,6 MB) fueron transmitidas correctamente por el nodo de último salto.

vii. En el cuarto escenario se analizaron los inconvenientes que tiene *CGR* cuando existen pérdidas entre los contactos (ver sección 4.6). Lo que se observó es que conforme la probabilidad de pérdida de contactos entre enlaces aumenta, los nodos ubicados en los extremos almacenan la mayor cantidad de paquetes que no pudieron ser entregados a las estaciones terrenas (ver figuras 4.23 y 4.24). Esto se debe a que estos nodos tienen un vecino al cual pueden ellos enviar información, mientras que las estaciones localizadas en el centro de la constelación poseen dos vecinos a los cuales podrían enviar sus imágenes. Cabe recalcar que la lógica de *CGR* no permite que los paquetes sean enviados de vuelta para atrás cuando se encuentran con un corte en el camino. Dado que el algoritmo guarda en la lista de nodos excluidos al nodo por donde recibió la información con el objetivo de evitar bucles de enrutamiento entre este nodo y el nodo de análisis.

viii. En el último escenario se consideró un tren de cuatro satélites más una estación terrena. Las decisiones de enrutamiento se toman en función de un plan de contactos calculado con parámetros orbitales reales (ver tabla 4.13), y además éste es optimizado mediante programación lineal, ya que se consideran para su cálculo no sólo las ventanas de contacto entre nodos, sino también el tráfico a cursar [46]. Esto se debe a que en los otros escenarios se asumió un caso ideal en donde por cada período orbital los nodos espaciales tendrían la posibilidad de descargar la información en las estaciones terrenas. En este escenario observamos que el emplear la variante de *CGR* de *Totsim* que almacena las rutas ya calculadas, permitió reducir

el cálculo del algoritmo en un 85,02 % (ver figura 4.26) a diferencia del Escenario 1 donde este cálculo se disminuyó en un 63,71 %. De igual manera, la asimetría que exhibe la topología del tren de satélites provoca que los nodos localizados a los extremos requieran una mayor cantidad de saltos para transmitir sus datos en la estación terrena. Finalmente, la topología de los contactos (ver figura 4.25) influye para que las ventanas de contacto entre los satélites tengan un mayor tamaño cuando los nodos *LEO1* y *LEO3* descargan la información. Esto implica un mayor tráfico de imágenes hacia estos nodos cuando estos descargan los datos tal como se indicó en la figura 4.28.

El trabajo realizado permitió comprender la importancia y el impacto de los esquemas de enrutamiento en redes tolerantes a demoras, y con especial énfasis en las comunicaciones por satélite. El aspecto más importante a destacar del proyecto, es que el rendimiento de *CGR* radica en su mayoría en el diseño del plan de contactos y la programación del tráfico a cursar en los nodos, ya que una mala configuración influye a que algunos nodos se sobrecarguen con mas información que otros, y en algunos casos con gran pérdida de datos.

Apéndice A

Código del programa

En este apartado se describen las funciones más importantes empleadas en el trabajo. En un principio se explica el algoritmo encargado de generar los planes de contactos a cargar en los nodos. Posteriormente, se discute sobre la función programada para importar los datos generados en *OMNeT++*. Finalmente, se mencionan las funciones que extraen y calculan cada uno de los parámetros de las imágenes [45].

A.1. Plan de Contactos

La tabla A.1 indica las variables empleadas para configurar el plan de contactos. Los parámetros de entrada para esta función son la cantidad de nodos satelitales, el número que identifica a la estación terrena, el período orbital de los satélites y las ventanas de contacto entre las estaciones satelitales y de éstas con la estación terrena. Como resultado de esta función, se genera un matriz que contiene el plan de contactos de todas las oportunidades de comunicación de los nodos para un tiempo de simulación determinado.

El algoritmo 4 [45] calcula el plan de contactos a cargar en cada uno de los nodos. Este algoritmo realiza el cálculo de los contactos en función de la topología presentada en las figuras 3.3 y 3.4. De este seudocódigo podemos observar que: las dos primeras líneas calculan la configuración de los nodos que descargan las imágenes en la estación terrena y estos valores son almacenados en una matriz (ver líneas 1 y 2). La estructura condicional que figura en la línea 11 tiene como objetivo crear el primer contacto y añadirlo al plan

Variable	Valor Inicial	Descripción
orbitalTime		Período de tiempo, en el cual un satélite da una vuelta a la tierra.
simulationTime		Tiempo de simulación.
timeSat		Ventana de contacto entre las estaciones satelitales.
timeTerr		Ventana de contacto entre los satelitales y la estación terrena.
nodSat		Cantidad de nodos satelitales.
idTerr		Identificador de la estación terrena.
id	0	Esta variable identifica cada oportunidad de contacto entre dos nodos en el plan de contactos.
contactPlan	[ ]	Matriz que contiene el plan de contactos
contacto	[ ]	Vector que contiene la ventana de comunicación entre dos nodos.
contTiempo	0	Esta variable acumula los valores de tiempo cada vez que se crea un contacto.
lastHop	[ ]	Matriz que contiene los nodos y los tiempos en que se descargan los datos en la estación terrena.
bajadaTierra		Función que configura los nodos que descargan la información en la estación terrena.
crearContact		Función que crea cada ventana de contacto.

Tabla A.1: Descripción de las variables y funciones empleadas para generar el plan de contactos [45].

de contactos. Por otro lado, la línea 15 verifica si ya se alcanzó el tiempo para agregar un contacto que descargue la información en la estación terrena, si este no es el caso se continúa creando contactos entre los nodos satelitales (ver línea 19). Finalmente, las líneas 17 y 25 tienen como propósito sumar los tiempos en función de si se agregó un contacto entre satélites o un contacto con la estación terrena.

A.2. Lectura de los archivos generados en OMNeT++

La tabla A.2 describe las variables y funciones empleadas para leer los archivos generados en *OMNeT++*. La variable *archivosOmnet* contiene el directorio de los archivos generados en la simulación. Cabe recordar que por cada nodo presente en la simulación se crea un archivo, el cual registra los paquetes que pasan a través de ese nodo. Dado que estos archivos no nos permiten conocer el nodo en donde se originó el paquete y su recorrido a través de los nodos, por esta razón se empleó la variable *archivosNodos*. Esta variable contiene el directorio de unos nuevos archivos, en donde cada uno contiene la trayectoria de cada paquete generado por cada nodo.

Algoritmo 4 Algoritmo que calcula el plan de contactos.
Las variables y funciones empleadas son descritas en la tabla A.1 [45].

Input: orbitalTime, simulationTime, timeSat, nodSat, idTerr y timeTerr.
Output: contactPlan

```
 1: numVueltas = simulationTime / orbitalTime
 2: lastHop = bajadasTierra(orbitalTime, nodSat , numVueltas)
 3: while contTiempo ≤ simulationTime
 4:    contSat = 1
 5:    for j = 1 to 2
 6:       sat1 = contSat
 7:       for i=1 to floor(nodSat/2)
 8:          sat2 = sat1+1
 9:          if sat2 ≤ nodSat then
10:             id = id+1
11:             if contTiempo == 0 then
12:                contacto = crearContact(id, sat1, sat2, timeSat)
13:                contacto append to contactPlan
14:             else
15:                if (contTiempo + timeSat) ≥ getTime(lastHop) then
16:                   contacto = crearContact(id, lastHop, idTerr, timeTerr)
17:                   contTiempo = getTime(lastHop) + timeTerr
18:                end if
19:                contacto = crearContact(id, sat1, sat2, timeSat)
20:                contacto append to contactPlan
21:             end if
22:             sat1 = sat1 + 2
23:          end if
24:       end for
25:       contTiempo = contTiempo + timeSat
26:       contSat = contSat + 1
27:    end for
28: end while
```

El algoritmo 5 [45] describe el proceso de lectura de los archivos creados en la simulación. La primera parte de este algoritmo (ver líneas 1 a 11) tienen como propósito procesar los archivos de *OMNeT++*, en dónde por cada línea se obtiene el identificador del nodo que creó el bundle de análisis (ver línea 4). Posteriormente, se almacena el paquete en función del nodo al que pertenece tal como se indica en la línea 8. Cabe mencionar, que el objetivo de las líneas 5 y 19 es verificar si existen los archivos donde se va a almacenar la información; en caso contrario estos son creados. Antes de empezar la segunda parte del algoritmo es importante mencionar que se ordenaron los archivos en función del identificador del paquete y los tiempos (ver línea 12). El objetivo es facilitar el análisis, ya que de esta manera los paquetes están ordenados de forma cronológica lo que permite calcular más rápido los parámetros como el tiempo de entrega de las imágenes ó los

Variable	Valor Inicial	Descripción
archivosOmnet	{ }	Un conjunto que contiene el directorio de los archivos generados en *OMNeT++*.
archivosNodos	{ }	Un conjunto que contiene el directorio de los archivos en donde se almacena el trayecto de los paquetes generados en cada nodo.
archivosImagenes	{ }	Un conjunto que contiene el directorio de los archivos en donde se almacena los paquetes por cada imagen generada.
ordenarArchivos		Esta método ordena los archivos en función del identificador de cada bundle y el tiempo.
getNodo		Cada línea de los archivos contiene una parte del recorrido del paquete a través de los nodos. Esta función obtiene el nodo que originó el paquete de análisis.
getImagen		Cada línea contiene una parte del recorrido del paquete a través de los nodos. Esta función obtiene la imagen a la cual pertenece el paquete de análisis.

Tabla A.2: Descripción de las variables y funciones empleadas leer los archivos generados en *OMNeT++* [45].

nodos de último salto.

En la segunda parte del algoritmo 5 (ver líneas 14 a 25) se procede a dividir los archivos con respecto a las imágenes; en donde cada imagen se almacena en un archivo individual. De esta manera se simplificó el análisis de los datos generados, ya que no es necesario procesar cada vez un archivo grande donde están todas las imágenes mezcladas.

A.3. Análisis de las imágenes

En el presente apartado se explica el procedimiento empleado para obtener todos los datos que contiene cada una de las imágenes. Las variables y funciones utilizadas son descritas en la tabla A.3.

Para comenzar a analizar los datos es necesario obtener todos los paquetes presentes en cada imagen, y que éstos a su vez se encuentren estructurados de una forma correcta. En este contexto, el algoritmo 6 [45] crea como salida un diccionario[1], en donde el elemento *clave()* almacena todos los identificadores de los paquetes, mientras que el elemento

[1]Un diccionario es un conjunto de pares *clave-valor* donde la clave es inmutable y el valor es cualquier cosa.

Algoritmo 5 Algoritmo que procesa los archivos generados de *OMNeT++* [45].

Input: Una lista llamada archivosOmnet que contiene el directorio de los archivos generados en *OMNeT++*.

Output: Un conjunto de archivos, en donde cada imagen y sus respectivos paquetes se almacena en un archivo individual.

```
1:  foreach file ∈ archivosOmnet
2:    f = open(file)
3:    for linea in f
4:      nodoSource = getNodo(linea)
5:      if archivosNodo.get(nodoSource) == false then
6:        archivosNodo[nodoSource] = crearArchivo(nodoSource)
7:      end if
8:      linea append to archivosNodo[nodoSource]
9:    end for
10:   f.close()
11: end for
12: ordenarArchivos(archivosNodo)
13: archivosNodo.close()
14: foreach file ∈ archivosNodo
15:   f = open(file)
16:   for linea in f
17:     nodoSource = getNodo(linea)
18:     idImagen = getImagen(linea)
19:     if archivosImagenes.get(nodoSource, idImagen) == false then
20:       archivosImagenes[idImagen] = crearArchivo(nodoSource,IdImagen)
21:     end if
22:     linea append to archivosImagenes[idImagen]
23:   end for
24:   f.close()
25: end for
26: archivosImagenes.close()
```

valor() alberga una matriz, la cual contiene el trayecto de cada paquete. Esto facilita posteriormente el análisis de cada uno de los paquetes, ya que para la búsqueda de un paquete en particular se lo realiza mediante su clave. Por otra parte, este seudocódigo necesita como parámetro de ingreso el directorio del archivo que alberga la imagen (ver línea 1), el cual fue procesado previamente por el algoritmo 5.

En particular, la línea 4 verifica si existe ya una matriz con *clave* ïdBundleçreada, si es que no es el caso se crea un nuevo elemento con dicha *clave*. El siguiente paso del algoritmo 6 [45] es separar cada una de las líneas del archivo texto y guardarlas como valores numéricos en la matriz según la *clave* del diccionario (ver línea 7).

Una vez que se tienen todos los paquetes estructurados de una forma apropiada, el siguiente paso es analizar cada uno de los parámetros de la imagen tal como se muestra

Variable	Valor Inicial	Descripción
bundles	{ }	Un diccionario, en donde el elemento *clave()* almacena todos los identificadores de los paquetes, mientras que el elemento *valor()* alberga una matriz la cual contiene todo el trayecto de los paquetes.
idNodo		Valor numérico que identifica al nodo.
idImagen		Valor numérico que identifica a la imagen.
idBundle		Valor numérico que identifica el paquete.
vectorIdBundles		Vector que almacena todos los identificadores de los paquetes de la imagen de análisis.
hopCount		Función que calcula el número de saltos de un paquete.
bundleAge		Función que calcula el tiempo de vida de los paquetes.
getLastHop		Función que obtiene el último nodo al cual llegaron los paquetes.

Tabla A.3: Descripción de las variables y funciones empleadas leer los archivos generados en *OMNeT++* y analizar las imágenes [45].

Algoritmo 6 Algoritmo **getBundles(idNodo, idImagen)**.
Este algoritmo obtiene todos bundles que se encuentran en la imagen [45].

Input: archivosImagenes, idNodo, idImagen
Output: Un diccionario llamado **bundles** la cual contiene como *clave()* el identificador del bundle y como *valor()* una matriz que contiene todo el trayecto del bundle.

```
1: file = open(archivosImagenes.get(idNodo, idImagen))
2: for linea in archivoImagen
3:    idBundle = getIdBundle(linea)
4:    if bundles.get(idBundle) == false then
5:       bundles[idBundle] = [ ]
6:    end if
7:    bundles[idBundle] = getDatos(linea)
8: end for
9: file.close()
```

en el algoritmo 7 [45]. En primer lugar, se analiza el último nodo al cual llegaron las paquetes (ver línea 8). Este valor nos permite obtener los nodos que fueron utilizados como nodos de descarga de datos en la estación terrena; en caso contrario nos ofrece la posibilidad de saber el último nodo al cual llegaron los paquetes que no fueron recibidos correctamente. Otros parámetros calculados son el tiempo de entrega de la imagen y la cantidad de paquetes entregados correctamente. Cabe mencionar que el número de saltos y el tiempo de vida de los paquetes se almacenan en vectores con el propósito de analizarlos después mediante herramientas estadísticas como diagramas de caja (ver líneas 6 y 7).

Algoritmo 7 Algoritmo **analizarImagen(idNodo, idImagen)**.
Este algoritmo analiza todos los parámetros presentes en una imagen en particular [45].

Input: idNodo, idImagen
Output: resultadoimagen

```
1: bundles = getBundles(idNodo, idImagen)
2: vectorIdBundles=getIdsBundles(bundles)
3: foreach idBundle ∈ vectorIdBundles
4:    hopCount = hopCount(bundles.get(idBundle))
5:    bundleAge = bundleAge(bundles.get(idBundle))
6:    hopCount append to vectorHopCount
7:    bundleAge append to vectorBundleAge
8:    lastHop = getlastHop(bundles.get(idBundle))
9: end for
10: tiempoInicial = getCreationImagen(bundles)
11: tiempoFinal = getFinImagen(bundles)
12: cantCorrectos= getBundlesOK(lastHop)
```

Finalmente, el algoritmo 8 [45] se encarga de analizar todas las imágenes generadas por los nodos; por consiguiente se invoca en reiteradas ocasiones al algoritmo 7 (ver línea 4). El rol que cumple este algoritmo es almacenar todos los resultados de las imágenes en un solo archivo de texto (ver figura 3.10).

El archivo de texto resultante permite analizar los datos mediante gráficas, ya sea mediante diagramas de caja o gráficas de barra como se mostró en detalle en el capítulo 4.

Algoritmo 8 Algoritmo que analiza todas las imágenes generadas en *OMNeT++* [45].

Input: nodSat, cantImagenes
Output: Un archivo llamado archivosResultados que contiene el resultado de todas las imágenes analizadas en la simulación.

```
1: file=open(archivoResultados)
2: for idNodo = 1 to nodSat
3:    for idImagen = 1 to cantImagenes
4:       resultadoImagen=analizarImagen(idNodo,idImagen)
5:       resultadoImagen append to file
6:    end for
7: end for
8: file.close()
```

Bibliografía

[1] J. A. Bava and A. J. Sanz. Microondas y recepcion satelital. *Comunicaciones*, 1995. URL http://sedici.unlp.edu.ar/handle/10915/36183.

[2] C. Caini, P. Cornice, R. Firrincieli, and D. Lacamera. A DTN approach to satellite communications. *Selected Areas in Communications, IEEE Journal on*, 26(5):820 – 827, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4530738.

[3] Scott Burleigh. Contact graph routing. *Internet Draft*, 2010. URL https://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-00.

[4] Ronald Charca Choque. Satélite artificial. *Revista de Información, Tecnología y Sociedad*, page 64, 2011. URL http://www.revistasbolivianas.org.bo/scielo.php?pid=S1997-40442011000100016&script=sci_abstract.

[5] Jahel O Ramos Mallea. Satélites de comunicación. *Revista de Información, Tecnología y Sociedad*, page 28, 2011. URL http://www.revistasbolivianas.org.bo/scielo.php?pid=S1997-40442011000100007&script=sci_arttext&tlng=es.

[6] Juan Ramírez Marín. Comunicacion via satelite. URL http://201.147.98.14/camara/content/download/193821/464902/file/comunicacion%20via%20satelite.pdf.

[7] ARTHUR C CLARKE. Sir arthur c. clarke—visionario de la era espacial. URL http://dialnet.unirioja.es/servlet/articulo?codigo=2582472&orden=263308&info=link.

[8] Domingo Lara Rodríguez. *Sistemas Inalámbricos de comunicación personal*. Marcombo, 2001.

[9] David J Whalen. Communications satellites: making the global village possible. *NASA History Office Publications*, 1997.

[10] Carlo Caini, Haitham Cruickshank, Stephen Farrell, and Mario Marchese. Delay- and disruption-tolerant networking (dtn): an alternative solution for future satellite networking applications. *Proceedings of the IEEE*, 99(11):1980–1997, 2011. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5959951`.

[11] Yurong Hu and Victor OK Li. Satellite-based internet: a tutorial. *IEEE Communications Magazine*, 39(3):154–162, 2001. URL `http://hub.hku.hk/handle/10722/44853`.

[12] John Border, Jim Griner, Gabriel Montenegro, Zach Shelby, and Markku Kojo. Performance enhancing proxies intended to mitigate link-related degradations. *Communications*, 2001. URL `http://tools.ietf.org/html/rfc3135.txt`.

[13] H Cruickshank. Technical report on performance enhancing proxies (peps) for the european etsi broadband satellite multimedia (bsm) working group. etsi report tr 102 676 (september 2009), 2001.

[14] Michele Luglio, M Yahya Sanadidi, Mario Gerla, and James Stepanek. On-board satellite"split tcp"proxy. *Selected Areas in Communications, IEEE Journal on*, 22(2):363–370, 2004. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1264081`.

[15] Swastik Kopparty, Srikanth V Krishnamurthy, Michalis Faloutsos, and Satish K Tripathi. Split tcp for mobile ad hoc networks. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 138–142. IEEE, 2002. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1188057`.

[16] Ronnie D Caytiles, Do-Huy Wang, Deock-Gil Oh, and Byungjoo Park. An enhanced packet buffering transmission (epbt) architecture design for performance enhancing proxies (peps). *International Journal of Software Engineering and Its Applications*, 6(4):155–164, 2012. URL `http://www.earticle.net/Article.aspx?sn=208409`.

[17] Venkatesh Obanaik, Lillykutty Jacob, and Akkihebbal L Ananda. Secure performance enhancing proxy: To ensure end-to-end security and enhance tcp performance over ipv6 wireless networks. *Computer Networks*, 50(13):2225–2238, 2006. URL `http://www.sciencedirect.com/science/article/pii/S1389128605002719`.

[18] Hugo Adrián Francisconi. Ipsec en ambientes ipv4 e ipv6. *Versión 1.0) ISBN*, pages 987–43, 2005. URL `http://redes-linux.com/manuales/seguridad/IPsec_IPv4_IPv6.pdf`.

[19] Alex McMahon and Stephen Farrell. Delay-and disruption-tolerant networking. *IEEE Internet Computing*, pages 82–87, 2009. URL `http://www.computer.org/csdl/mags/ic/2009/06/mic2009060082-abs.html`.

[20] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. Technical report, RFC 4838, April, 2007. URL `http://www.rfc-editor.org/rfc/rfc4838.txt`.

[21] Scott Burleigh, Vinton Cerf, Robert Durst, Kevin Fall, Adrian Hooke, Keith Scott, and Howard Weiss. The interplanetary internet: a communications infrastructure for mars exploration. *Acta astronautica*, 53(4):365–373, 2003. URL `http://www.sciencedirect.com/science/article/pii/S0094576503001541`.

[22] K. L. Scott and S. Burleigh. Bundle protocol specification. RFC 5050, November. 2007. URL `http://tools.ietf.org/html/rfc5050`.

[23] Michael Demmer, Jörg Ott, and Simon Perreault. Delay tolerant networking tcp convergence layer protocol. *RFC 7242, June*, 2014. URL `https://tools.ietf.org/html/rfc7242`.

[24] H Kruse and S Ostermann. Udp convergence layers for the dtn bundle and ltp protocols. *IETF Draft*, 2008. URL `https://tools.ietf.org/html/draft-irtf-dtnrg-udp-clayer-00`.

[25] Scott Burleigh. Delay-tolerant networking ltp convergence layer (ltpcl) adapter. *Communications*, 2013. URL `http://tools.ietf.org/html/burleigh-dtnrg-ltpcl-01`.

[26] Manikantan Ramadas, Scott Burleigh, et al. Licklider transmission protocol-specification. *Communications*, 2008. URL `http://tools.ietf.org/html/rfc5326.txt`.

[27] C Caini. Delay-tolerant networks (dtns) for satellite communications. *Advances In Delay-Tolerant Networks (DTNs): Architecture and Enhanced Performance*, page 25, 2014.

[28] Sushant Jain, Kevin Fall, and Rabin Patra. *Routing in a delay tolerant network*, volume 34. ACM, 2004. URL `http://dl.acm.org/citation.cfm?id=1015484`.

[29] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. Replication routing in dtns: a resource allocation approach. *IEEE/ACM Transactions on Networking (TON)*, 18(2):596–609, 2010. URL `http://dl.acm.org/citation.cfm?id=1816282`.

[30] Michael Demmer and Kevin Fall. Dtlsr: delay tolerant routing for developing regions. In *Proceedings of the 2007 workshop on Networked systems for developing regions*, page 5. ACM, 2007. URL `http://dl.acm.org/citation.cfm?id=1326579`.

[31] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000. URL `ftp://jflap.org/.snapshot/nightly.6/tr/2000/2000-06.ps`.

[32] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003. URL `http://link.springer.com/chapter/10.1007/978-3-540-27767-5_24`.

[33] Anders Lindgren, Elwyn Davies, Samo Grasic, and Avri Doria. Probabilistic routing protocol for intermittently connected networks. *Communications*, 2012. URL `http://tools.ietf.org/html/rfc6693.txt`.

[34] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005. URL `http://dl.acm.org/citation.cfm?id=1080143`.

[35] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM*, volume 6, pages 1–11, 2006. URL `http://briangallagher.net/pubs/burgess-infocom-2006.pdf`.

[36] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. Dtn routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review*, 37(4):373–384, 2007. URL `http://dl.acm.org/citation.cfm?id=1282422`.

[37] Jay Wyatt, Scott Burleigh, Ross Jones, Leigh Torgerson, and Steve Wissler. Disruption tolerant networking flight validation experiment on nasa's epoxi mission. In *Advances in Satellite and Space Communications, 2009. SPACOMM 2009. First International Conference on*, pages 187–196. IEEE, 2009. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5194608`.

[38] S. Burleigh. Contact graph routing. *Internet Draft*, 2011. URL `https://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-01`.

[39] E. Birrane, S. Burleigh, and N. Kasch. Analysis of the contact graph routing algorithm: Bounding interplanetary paths. *Acta Astronautica*, 75:108 – 119, 2012. URL `http://www.sciencedirect.com/science/article/pii/S0094576512000288`.

[40] Carlo Caini and Rosario Firrincieli. Application of contact graph routing to leo satellite dtn communications. pages 3301–3305, 2012. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6363686`.

[41] Adelina García, Nilda Pérez Otero, Marcelo Pérez Ibarra, and Cecilia María Lasserre. Simulación de clusters: integración de inet a clusim. In *XVII Congreso Argentino de Ciencias de la Computación*, 2011. URL `http://sedici.unlp.edu.ar/handle/10915/18691`.

[42] S Wang, KZ Liu, and FP Hu. Simulation of wireless sensor networks localization with omnet. In *Mobile Technology, Applications and Systems, 2005 2nd International Conference on*, pages 1–6. IEEE, 2005. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1656681`.

[43] András Varga et al. The omnet++ discrete event simulation system. volume 9, page 65. sn, 2001. URL `https://labo4g.enstb.fr/twiki/pub/Simulator/SimulatorReferences/esm2001-meth48.pdf`.

[44] Johanna Patricia Fonte Arequipa and Fabiola Elizabeth Mora Mulla. *Implementación de protocolos de capar de enlace de datos en los simuladores Omnet++ Y Ns-2*. PhD thesis, QUITO/EPN/2008, 2008. URL `http://bibdigital.epn.edu.ec/handle/15000/4238`.

[45] Grupo de redes complejas y comunicación de datos. Software para la simulación de redes satelitales. 2015. URL `http://cnet.fi.uba.ar/simulaciones_redes_satelitales/`.

[46] J. Fraire, P. Madoery, J. M. Finochietto, and I. Edward Birrane. Congestion modeling and management techniques for predictable disruption tolerant networks. *in 40th Annual IEEE Conference on Local Computer Networks (LCN 2015), Clearwater Beach, USA, Oct. 2015.*