Trabajo Profesional de Ingeniería en Informática

ANTop - Satelital

Tutor: Dr. Ing. José Ignacio Alvarez Hamelin

Co-Tutor: Dr. Ing. Pablo Gustavo Madoery

Alumnos

Gastón Frenkel *(Padrón 107718)*
gfrenkel@fi.uba.ar

Valentina Adelsflügel *(Padrón 108201)*
vadelsflugel@fi.uba.ar

Facultad de Ingeniería
Universidad de Buenos Aires

25 de marzo de 2026

Agradecimientos

En primer lugar, agradecemos a nuestras familias y mascotas por apoyarnos en los momentos difíciles y en los buenos también.

A nuestros amigos, con quienes compartimos incontables horas de clase y memorables momentos, tanto dentro como fuera de la Facultad. Sin ellos, este recorrido no hubiese sido lo mismo.

A nuestros tutores, el Dr. Ing. José Ignacio Álvarez Hamelin y el Dr. Ing. Pablo Madoery por el acompañamiento y la oportunidad de realizar este trabajo, permitiéndonos finalizar nuestros estudios.

A la Facultad de Ingeniería de la Universidad de Buenos Aires y sus docentes por habernos formado en esta profesión con los más altos estándares.

Índice

1. Introducción

Este trabajo consiste en la implementación del protocolo Adjacent Network Topology (ANTop) en un entorno de simulación de redes satelitales, comparando su desempeño con el protocolo Contact Graph Routing (CGR) para determinar su eficiencia y viabilidad en este tipo de redes.

En particular, el trabajo implicó resolver cinco problemáticas principales:

1. Desarrollar una implementación en C++ del protocolo ANTop integrándolo con la biblioteca H3.
2. Configurar y ejecutar simulaciones del protocolo ANTop utilizando simuladores basados en OMNET++, modelando escenarios que reflejen condiciones típicas de redes satelitales.
3. Realizar una comparación de desempeño entre ANTop y el protocolo CGR, utilizando Python para el análisis de los resultados obtenidos en las simulaciones.
4. Evaluar los resultados obtenidos en términos de parámetros clave, como latencia, tasa de entrega de paquetes y tiempo de procesamiento, con el fin de determinar las fortalezas y limitaciones de ANTop en comparación con CGR.
5. Documentar las conclusiones derivadas del análisis comparativo, identificando posibles áreas de mejora y sugerencias para trabajos futuros en el ámbito de las comunicaciones satelitales.

En lo académico, se aplicaron conocimientos adquiridos en materias como Algoritmos y Programación (I, II y III), Redes, Programación Concurrente, Simulación y Sistemas Distribuidos I. Además, existió un desafío adicional relacionado a la gestión del proyecto, la división de tareas y la estimación de tiempos, para lo cual se aplicaron contenidos relacionados a las materias Gestión del Desarrollo de Sistemas Informáticos y Taller de Programación II.

2. Estado del arte

A diferencia de las redes terrestres, las redes satelitales presentan desafíos únicos debido a su topología dinámica, la variabilidad en la conectividad y las altas latencias de propagación de señales producto del movimiento de los satélites, lo cual genera la necesidad de desarrollar e implementar algoritmos de enrutamiento específicos que se adapten a estos cambios.

En base a esto, existe una tendencia hacia una arquitectura de protocolos basada en capas jerárquicas la cual integra elementos espaciales, aéreos y terrestres. A su vez, el componente satelital de dicha arquitectura puede ser considerado una única capa (single-layer satellite networks) o bien puede subdividirse en dos o tres capas, con cada satelite perteneciendo a una capa u otra en función de la distancia desde su órbita a la Tierra (multi-layer satellite networks) [3]. Algunos ejemplos de algoritmos incluyen LZDR (Localized Zone Distributed Routing), DRA (Datagram Routing Algorithm), LCRA (Low-complexity Probabilistic Routing Algorithm), MLSR (Multilayer Satellite Routing), SGRP (Satellite Grouping and Routing Protocol), entre muchos otros [14].

Otro protocolo relevante es Contact Graph Routing (CGR), el cual se utiliza en Delay-Tolerant Networks (DTN) para gestionar el enrutamiento en entornos donde la conectividad es intermitente y las latencias son elevadas [2]. Esto lo logra calculando aproximaciones de rutas óptimas basándose en la predicción de contactos futuros entre nodos. Aunque CGR ha demostrado ser efectivo en la mitigación de algunos de los problemas asociados con las redes satelitales, su necesidad de poseer un conocimiento preciso de la dinámica de la red limitan su escalabilidad a escenarios complejos.

En este contexto, la implementación y simulación del protocolo ANTop en redes satelitales busca ofrecer una solución eficiente, adaptable y escalable, capaz de enfrentar los desafíos mencionados.

Este trabajo se desarrolla en el marco del proyecto de investigación ANTop, donde se han publicado los siguientes trabajos:

- José Ignacio Alvarez-Hamelin, Aline Carneiro Viana, Marcelo Dias de Amorim. DHT-based Functionalities Using Hypercubes. IFIP 19th World Computer Congress, Aug 2006, Santiago, Chile. pp.157-176.
- A. Marcu. Desarrollo y simulación de un protoclo para redes ad-hoc. University of Buenos Ares, Jul 2007, Buenos Aires, Argentina.
- Matías Campolo. Estudio y análisis del funcionamiento de ANTop sobre IPv6. University of Buenos Aires, Apr 2012, Buenos Aires, Argentina.
- Pablo D. Torrado. Fragmentación y Mezcla de redes ad-hoc utilizando el protocolo ANTop sobre IPv6. University of Buenos Aires, Mar 2018, Buenos Aires, Argentina.

3. ANTop Satelital

3.1. Introducción al protocolo original

En redes Delay/Disruption Tolerant Networks (DTN), los mecanismos clásicos de ruteo basados en conectividad permanente resultan inadecuados debido a la naturaleza intermitente de los enlaces, las largas latencias y la movilidad de los nodos. En este contexto, resulta necesario emplear protocolos capaces de tomar decisiones de ruteo sin depender de una topología estática ni de información global consistente.

El protocolo ANTop (Adjacent Network Topologies) [9], a diferencia de otros enfoques basados en el intercambio continuo de información de estado o en la replicación probabilística de mensajes, codifica la topología de la red directamente en las direcciones de los nodos. Esto permite que cada nodo tome decisiones de ruteo utilizando únicamente información local, sin necesidad de mantener tablas globales o conocimiento completo de la red. Está diseñado para redes ad-hoc descentralizadas en donde todos los nodos tienen igual jerarquía. Dado que las direcciones cambian dinámicamente, los nodos cuentan además con identificadores universales que permiten su identificación persistente.

Para lograr este comportamiento, ANTop impone una estructura lógica sobre la red. La idea central del protocolo es modelar la topología como una proyección sobre un hipercubo [1]. En este modelo, cada nodo posee una dirección representada como una secuencia de bits, la cual refleja su posición relativa dentro de dicha estructura.

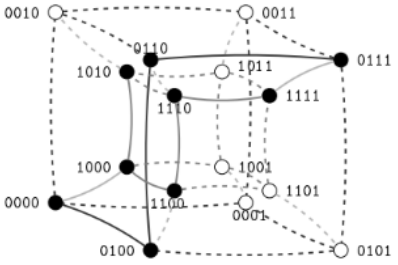

Figura 1: Hipercubo de dimensión $n = 4$.

Aplicado a redes, el caso ideal sería aquel en el que cada nodo (por ejemplo, un satélite) se corresponde con un vértice del hipercubo. Sin embargo, esto no es factible en escenarios reales, por lo que se trabaja con hipercubos incompletos, donde no todos los vértices tienen un nodo asignado. Cuando un nuevo nodo desea incorporarse a la red, emite una solicitud en modo broadcast dirigida a los nodos dentro de su radio de cobertura. Cada nodo existente, además de su dirección relativa, mantiene asociada una máscara que define el subconjunto del espacio de direcciones que administra. A partir de este conjunto, el nodo puede ceder una dirección disponible al nuevo nodo, permitiendo así su incorporación al hipercubo [9].

La distancia topológica entre dos nodos, expresada en términos de número de saltos (hops), se calcula mediante la operación XOR entre sus direcciones. Esta operación no solo permite cuantificar la distancia, sino también identificar en qué posiciones difieren ambas direcciones. En un hipercubo completo, esto habilita un esquema de encaminamiento greedy, donde en cada salto se selecciona un vecino cuya dirección reduzca dicha diferencia, garantizando así un camino de longitud mínima hacia el destino.

No obstante, en el caso de hipercubos incompletos, no siempre existe un vecino que permita reducir la distancia en cada paso y, como consecuencia, no es posible garantizar la existencia de caminos óptimos. Aun así, el protocolo intenta aproximarse progresivamente al destino siguiendo este criterio.

Esta limitación motiva la incorporación de mecanismos adicionales que permitan continuar el enrutamiento cuando el progreso se ve interrumpido. En la siguiente sección se describe el enfoque de ruteo reactivo utilizado para abordar estos casos.

3.2. Ruteo reactivo

El ruteo o enrutamiento es el proceso mediante el cual se selecciona una secuencia de nodos intermedios para transportar un paquete desde un origen hasta un destino. En términos generales, los algoritmos de ruteo pueden clasificarse en dos grandes categorías: proactivos y reactivos.

En los esquemas proactivos, las rutas hacia todos los destinos se calculan y mantienen actualizadas de manera continua. Este enfoque permite disponer de caminos óptimos en todo momento, pero requiere la propagación constante de información de estado ante cualquier cambio en la topología de la red. En entornos altamente dinámicos como lo son las redes satelitales, este mantenimiento puede implicar un costo significativo en términos de señalización y consistencia.

Por el contrario, los esquemas reactivos calculan las rutas únicamente cuando son necesarias. Las decisiones se toman en base al estado local disponible al momento de rutear, sin requerir conocimiento global completo ni tablas previamente calculadas. Si bien esto puede conducir ocasionalmente a caminos subóptimos, reduce la necesidad de sincronización global y resulta particularmente adecuado para redes con cambios frecuentes de conectividad.

En el contexto de redes satelitales LEO, donde la topología varía continuamente debido al movimiento orbital y no siempre se dispone de un hipercubo completo de conectividad, mantener rutas proactivamente actualizadas resulta costoso y potencialmente ineficiente, razón por la cual se adoptó un esquema de ruteo reactivo.

La implementación desarrollada en este trabajo se basa en el algoritmo ANTop propuesto por Marcu [9], el cual realiza la selección de next hop de manera dinámica en cada nodo. No obstante, como se detallará en la siguiente subsección, se introducen modificaciones con el objetivo de mejorar la exploración de rutas alternativas a un camino inválido.

3.3. Tabla de ruteo

Cada nodo debe mantener un estado local mínimo que le permita seleccionar dinámicamente el próximo salto hacia un destino determinado. Debido a que ANTop opera bajo un esquema de ruteo reactivo, las entradas de la tabla no son precomputadas globalmente, sino que se construyen dinámicamente a partir de las decisiones de ruteo realizadas durante la operación del sistema.

En este contexto, la tabla de ruteo implementada en este trabajo se encuentra basada en la estructura y el algoritmo propuesto por Marcu [9], manteniendo la semántica principal de selección de vecino por prioridad hacia el destino. No obstante, se introducen extensiones orientadas a mejorar el manejo de bucles y la coherencia del reencaminamiento bajo escenarios dinámicos.

3.3.1. Estructura

Cada nodo mantiene un estado local de ruteo que almacena la información necesaria para tomar decisiones de reenvío de paquetes. Este estado permite reutilizar conocimiento derivado de decisiones de ruteo previas, evitando recomputaciones innecesarias durante la operación del protocolo.

En la implementación propuesta, el estado de ruteo del nodo está compuesto por dos estructuras principales:

1. Tabla de ruteo

   Dado un destino, permite acceder a información acerca del próximo salto (*nexthop*), la distancia hacia el destino o aquellos vecinos que han sido explorados durante un proceso de búsqueda anterior. Indexada por identificador de destino, la misma almacena:

   - Next hop: vecino actualmente seleccionado para rutear hacia el destino.
   - Distance: distancia estimada hacia el destino.
   - Neighbors: conjunto de vecinos, ordenados de forma ascendente según distancia al destino.
   - visited bitmap: estructura de bits que registra qué vecinos ya han sido explorados con anterioridad. A medida que nuevos vecinos son explorados, sus correspondientes bits son marcados como visitados.

2. Tabla por pares origen-destino

   Indexada por el par origen-destino, almacena información utilizada para la detección y corrección de bucles en el algoritmo.

   - Distance: menor número de hops con el que el nodo recibió un paquete de origen *source* y destino *destination*. Permite identificar bucles sin guardar información específica sobre cada paquete recibido.

Destination	Next Hop	Distance	Neighbors	visited bitmap
J	H	-	H,G,F,D	1000
B	D	3	D,G,F,H	0000

Cuadro 1: Tabla de ruteo luego de que un nodo haya recibido y enviado un paquete de origen B y destino J.

Source	Destination	Distance	Loop Epoch
B	H	3	0

Cuadro 2: Tabla por pares origen-destino luego que un nodo haya recibido un paquete que pasó por tres nodos anteriores.

- Loop epoch: contador que permite mantener coherencia en los procesos de resolución de bucles. La incorporación de esta estructura constituye una extensión respecto a la implementación original.

Ejemplos de la tabla de ruteo y de la tabla por pares origen-destino pueden ser observados en la Tabla 1 y la Tabla 2, respectivamente.

3.3.2. Detección de bucles

Se considera que un paquete entró en un bucle cuando este se encuentra en un ciclo repetitivo de nodos del cual no puede salir a menos que se dé un cambio de topología. En esquemas reactivos basados en decisiones locales, la posibilidad de ciclos temporales es inherente, particularmente en redes de poca conectividad entre nodos.

En la implementación original de ANTop, la detección de un loop se resuelve devolviendo el paquete al nodo emisor inmediato [9]. Este mecanismo garantiza la ruptura inmediata del ciclo, pero no fuerza una reexploración local de alternativas, delegando esta decisión en el nodo previo. Si bien esta estrategia asegura progresión, introduce una limitación exploratoria: el nodo que detecta el loop no reconsidera su conjunto de vecinos para el destino en cuestión, lo que puede impedir el descubrimiento de caminos alternativos válidos.

En consecuencia, en la implementación propuesta, cuando un nodo detecta un loop, en lugar de retornar automáticamente el paquete al *sender* (nodo del cual se recibe dicho paquete), inicia un proceso de búsqueda de un nuevo vecino candidato. Dicho proceso de búsqueda se puede resumir en:

1. Obtener el conjunto de vecinos (neighbors) ordenados ascendentemente según distancia al destino. Este conjunto puede ser obtenido de la tabla indexada por destino en caso de que un paquete ya haya sido enviado con anterioridad al mencionado destino.
2. Excluir aquellos vecinos ya explorados utilizando el visited bitmap.
3. Evitar, en primera instancia, seleccionar nuevamente al sender. Esto se realiza con el

	Nodo X	Paquete 1	Paquete 2
Distancia al origen	?	5	5
Loop epoch	0	0	0

Cuadro 3: Estados de la tabla de pares del nodo X y de los paquetes antes de ser procesados.

objetivo de explorar todas las rutas posibles antes de retornar el paquete.

4. Seleccionar el vecino no explorado cuya distancia al destino sea la más baja entre todos los vecinos.

Si bien este mecanismo de explorar nuevos vecinos al momento de detectar bucles permite descubrir nuevas rutas que la implementación original ignoraría, el mismo introduce un fenómeno a mitigar en paquetes con mismo par origen-destino enviados en un plazo de tiempo acotado. A modo de ejemplo, un mensaje, en caso de tener un peso considerable, podría ser fragmentado en varios paquetes los cuales son enviados desde un mismo origen y hacia un mismo destino en una ventana de tiempo extremadamente acotada. Si se llegase a generar un bucle para dichos paquetes, será un mismo nodo el cual detecte dicho bucle para todos esos paquetes. El primer bucle detectado provocaría una nueva búsqueda de una ruta, lo cual es válido. Sin embargo, debido a que se detectaría un bucle para todos los paquetes siguientes debido a que todos comparten origen y destino, cada uno de ellos dispararía una búsqueda independiente, lo cual provocaría que todos los paquetes sean ruteados a través de un vecino distinto, hasta que no haya vecinos disponibles.

Es en este contexto que se introduce un mecanismo de versionado denominado loop epoch. Cada par origen-destino mantiene un contador entero que representa la versión actual del proceso de resolución de bucles, mientras que cada paquete transporta su propio valor de epoch. Ante la detección de un bucle, el nodo compara el epoch del paquete con el almacenado localmente. Si el epoch local es mayor, esto quiere decir que el bucle ya fue resuelto con anterioridad y el paquete es ruteado con la decisión vigente. Si el epoch local es menor o coincide con el del paquete, se dispara un nuevo proceso de exploración.

El mecanismo de loop epoch no actúa únicamente como un contador local de reexploraciones, sino como un esquema de versionado distribuido asociado al par origen-destino. Cada resolución de bucle incrementa monotonamente la versión local, la cual es propagada implícitamente a través de los paquetes. La comparación entre el loop epoch de un paquete y el almacenado localmente en un nodo implementa una política de convergencia basada en el máximo valor observado, garantizando consistencia eventual sin requerir sincronización explícita entre nodos.

Si se vuelve a considerar el ejemplo anterior en donde un mensaje se fragmente en varios paquetes, se tendría, en un inicio, variables como las de la Tabla 3.

Debido a que el Nodo X no tiene información acerca de su distancia al origen de los paquetes, actualiza su tabla con la información de la cantidad de saltos que les llevó a los paquetes llegar desde el origen hasta el Nodo X. De esta forma, queda la Tabla 4. Cabe destacar que una vez

	Nodo X	Paquete 1	Paquete 2
Distancia al origen	5	6	6
Loop epoch	0	0	0

Cuadro 4: Estado del nodo X luego de procesar y rutear ambos paquetes.

	Nodo X	Paquete 1	Paquete 2
Distancia al origen	5	8	8
Loop epoch	0	0	0

Cuadro 5: Estado del nodo X y de los paquetes antes de ser nuevamente procesados.

que los paquetes son ruteados por el Nodo X, la distancia al origen de los mismos aumenta en uno debido a que dicho ruteo debe involucrar un nuevo salto.

Si ahora los paquetes entrasen en un bucle de tres nodos, los mismos volverían al Nodo X con una distancia al origen (número de hops) de 8, tal como se puede apreciar en la Tabla 5.

Debido a que el número de hops de los paquetes es mayor a la distancia del Nodo X al origen, se detecta que los paquetes entraron en un bucle. Asumiendo que el primer paquete en llegar es el paquete 1, se dispararía el proceso de resolución de bucles y, debido a que el loop epoch del nodo y el del paquete son iguales (0), el Nodo X realizaría una reexploración de rutas. Como resultado, los loop epoch del nodo y del paquete aumentarían en uno, como se puede apreciar en la Tabla 6, mientras que la distancia al origen del paquete 1 es corregida por el valor original.

Al igual que con el paquete 1, el Nodo X también detectaría en un bucle al paquete 2 debido a que su número de hops es mayor a la distancia del Nodo X al origen. Sin embargo, el loop epoch del nodo es mayor al del paquete, por lo que se presume que dicho bucle ya fue resuelto con anterioridad y, en vez de realizar una reexploración, simplemente se utiliza el next hop actualmente vigente, no sin antes actualizar el loop epoch del paquete 2. De esta forma, las variables quedarían tal como en la Tabla 7.

Si ambos paquetes ingresaran posteriormente en un nuevo bucle detectado por un nodo distinto Y, dicho nodo ejecutaría el mismo procedimiento de comparación de versiones. En caso de observar un loop epoch transportado por el paquete mayor al almacenado localmente, el nodo actualizaría su estado adoptando dicha versión, aun cuando no haya participado en la resolución original del bucle. De esta forma, la información sobre la resolución no permanece

	Nodo X	Paquete 1	Paquete 2
Distancia al origen	5	6	8
Loop epoch	1	1	0

Cuadro 6: Estados del nodo X y de los paquetes luego de que el primero de estos sea nuevamente procesado y ruteado.

	Nodo X	Paquete 1	Paquete 2
Distancia al origen	5	6	6
Loop epoch	1	1	1

Cuadro 7: Estados del nodo X y de los paquetes luego de que sean nuevamente procesados y ruteados.

confinada al nodo que la generó, sino que se propaga implícitamente a través del tráfico de datos, garantizando que los distintos nodos converjan progresivamente hacia la versión más reciente observada para el par origen-destino.

Es de esta forma que el sistema implementa una política de convergencia, sin necesidad de implementar mecanismos explícitos de sincronización o coordinación global. Mientras que el algoritmo asegura una única reexploración por versión de epoch, todas las detecciones de bucle posteriores convergen hacia la misma decisión. De esta forma, el mecanismo de loop epoch actúa como un esquema ligero de coherencia distribuida proporcionando consistencia eventual en decisiones de reexploración.

3.3.3. Algoritmo

Con el objetivo de facilitar la reproducibilidad del mecanismo propuesto y permitir su implementación independiente, a continuación se presenta el pseudocódigo correspondiente a los algoritmos principales involucrados en el ruteo reactivo de ANTop.

Algorithm 1 `findNewNeighbor`

Require: `pkt`, `sender`, `dst`

```
1: if ttlExpired() then
2:     clearTables()
3: end if
4: nextHop ← findNextCandidate(dst, sender)
5: if nextHop = None then
6:     nextHop ← self
7: end if
8: routingTable[dst].nextHop ← nextHop
9: return nextHop
```

El Algoritmo 1 implementa la función `findNewNeighbor`, que devuelve el mejor vecino para rutear al paquete `pkt` con destino `dst`, en función del estado actual de las tablas del nodo. Los pasos 1 a 3 limpian las tablas del nodo cuando se ha excedido el TTL de las mismas. Los pasos 4 a 7 se encargan de la búsqueda del mejor próximo salto para rutear al paquete `pkt` hacia `dst`. En caso de que todos los vecinos ya hayan sido explorados, se devuelve el propio nodo. Por último, el paso 8 actualiza el estado de la tabla de pares para el destino `dst` con la nueva información de ruteo.

Algorithm 2 `findNextCandidate`

Require: `dst`, `sender`

```
1: for neighbor ∈ neighbors do
2:     if routingTable[dst].visitedBitmap[neighbor] = false ∧ neighbor ≠ sender then
3:         routingTable[dst].visitedBitmap[neighbor] = true
4:         return neighbor
5:     end if
6: end for
7: if routingTable[dst].visitedBitmap[sender] = false then
8:     routingTable[dst].visitedBitmap[sender] = true
9:     return sender
10: end if
11: return None
```

La segunda función, `findNextCandidate`, es implementada en el Algoritmo 2. Esta devuelve el próximo mejor vecino para rutear un paquete hacia su destino. Los pasos 1 a 6 eligen el primer vecino no visitado ya que estos se encuentran ordenados de forma ascendente según distancia al destino. Cabe destacar que se decide ignorar al sender, independientemente de su distancia al destino, con el objetivo de evitar devolver el paquete por donde llegó y fomentando la exploración de nuevas rutas de forma local. Por su lado, los pasos 7 a 11 actúan como alternativas en caso de no existir vecino disponible, priorizando devolver al paquete por donde llegó, o devolviendo `None` en caso de que todos los vecinos ya hayan sido explorados.

El algoritmo 3 implementa la función `findNextHop`, la cual, al igual que `findNewNeighbor`, devuelve el próximo mejor salto para acerca al paquete `pkt` hacia su destino `dst`. La diferencia entre ambas funciones radica en que `findNewNeighbor` fuerza la exploración de un nuevo vecino, mientras que `findNextHop` hace uso de las entradas de la tabla de ruteo y la tabla de pares para evaluar si un camino actualmente vigente sigue siendo válido; en caso de serlo, se sigue apostando por dicho camino en vez de forzar una posible exploración innecesaria.

Al igual que `findNewNeighbor`, los pasos 1 a 3 limpian las tablas del nodo cuando se ha excedido el TTL de las mismas. Los pasos 4 a 6 son los encargados de detectar y corregir bucles. Para ello, se busca si anteriormente ha pasado un paquete con mismo origen y destino pero con una distancia al origen menor, en cuyo caso se procede a validar y/o corregir dicho bucle. El paso 7 actualiza la información de distancia en la tabla de pares para el par origen-destino. En los pasos 8 a 12 se agrega la entrada inversa si no existe o si es peor que la actualmente utilizada. Es decir, se almacena la ruta para ir hacia el nodo origen `src`. Finalmente, los pasos 13 a 16 devuelven el próximo salto para el destino `dst`, en caso de existir dicha entrada en la tabla de ruteo, o bien se invoca a `findNewNeighbor` para explorar un vecino.

Cabe destacar que el sistema de detección de bucles mediante la comparación de distancias al origen de paquetes anteriores puede provocar falsos positivos en escenarios donde un cambio de

Algorithm 3 `findNextHop`

Require: `pkt`, `src`, `sender`, `dst`

```
1: if ttlExpired() then
2:     clearTables()
3: end if
4: if (src, dst) ∈ pairTable ∧ pairTable[(src,dst)].distance < pkt.hopCount then
5:     return handleLoop(pkt,src,sender,dst)
6: end if
7: pairTable[(src,dst)].distance ← pkt.hopCount
8: if dst ∉ routingTable ∨ routingTable[src].distance > pkt.hopCount then
9:     routingTable[src].nextHop ← sender
10:    routingTable[src].distance ← pkt.hopCount
11:    routingTable[src].visitedBitmap[sender] = true
12: end if
13: if dst ∈ routingTable then
14:    return routingTable[dst].nextHop
15: end if
16: return findNewNeighbor(pkt, sender, dst)
```

topología o una falla de un nodo haya provocado una ligera desviación en la ruta originalmente utilizada. Es por esta razón que se introdujo un margen de error de forma tal de que aquellos paquetes cuya distancia al origen sea mayor a la de la tabla de pares, pero que no superen el umbral establecido, no sean detectados como en bucle. En caso de que verdaderamente sea un bucle de pocos nodos, este eventualmente sería capturado luego de algunas iteraciones extra. Sin embargo, en las simulaciones realizadas se notó que la necesidad de que algunos paquetes permanezcan más tiempo en bucle antes de que este sea corregido no fue suficientemente contrarrestada por la reducción en falsos positivos. Por lo tanto, se decidió mantener el método de detección de bucles original sin umbral de detección.

Algorithm 4 `handleLoop`

Require: `pkt`, Source `src`, `sender`, `dst`

```
1: if pkt.loopEpoch < pairTable[(src,dst)].loopEpoch then
2:     pkt.loopEpoch ← pairTable[(src,dst)].loopEpoch
3:     return routingTable[dst].nextHop
4: end if
5: pkt.loopEpoch ← pkt.loopEpoch + 1
6: pairTable[(src,dst)].loopEpoch ← pairTable[(src,dst)].loopEpoch + 1
7: if pkt.loopEpoch > pairTable[(src,dst)].loopEpoch then
8:     pairTable[(src,dst)].loopEpoch ← pkt.loopEpoch
9: end if
10: return findNewNeighbor(pkt, sender, dst)
```

Por último, se puede apreciar la implementación de la función `handleLoop` en el Algoritmo 4. Los pasos 1 a 4 detectan y corrigen un bucle que ya fue solucionado anteriormente para otro paquete, actualizando el loop epoch del paquete actual, y devolviendo el vigente próximo salto. A su vez, los pasos 5 a 10 se encargan de corregir un bucle nuevo, incrementando el loop epoch del paquete en uno con el fin de señalizar una nueva versión en el control de bucles, y actualizando el loop epoch del nodo para el par origen-destino. Esta última actualización se realiza en función de cuál es el loop epoch más grande: si el actual del nodo, o el del paquete a rutear. Una explicación más en detalle del algoritmo puede ser estudiada en la sección 3.3.2.

3.4. Desafíos en constelaciones de Satélites

3.4.1. Representacion terrestre

Para aplicar ANTop en constelaciones satelitales es necesario definir cómo se representan las posiciones geográficas dentro del esquema de direccionamiento del protocolo. Para ello, resulta indispensable contar con una representación adecuada de la superficie terrestre sobre la cual se distribuyen los satélites y se definen las regiones de cobertura.

En trabajos anteriores, la superficie de la Tierra ha sido modelada mediante una grilla de celdas cuadradas, utilizada como mecanismo de mapeo espacial. Si bien este enfoque simplifica la implementación y el análisis, presenta limitaciones inherentes debido a que la Tierra es, en primera aproximación, una superficie esférica y no plana.

La utilización de una grilla de celdas cuadradas introduce deformaciones geométricas que afectan tanto la representación de distancias como las relaciones de vecindad entre regiones. Estas distorsiones se vuelven particularmente significativas a medida que aumenta la extensión del área considerada, lo que puede impactar negativamente en la precisión del modelo y, en consecuencia, en el desempeño del algoritmo de ruteo.

3.4.2. Solución propuesta

Con el objetivo de mitigar las ya mencionadas limitaciones, se opta por utilizar una discretización basada en celdas hexagonales en lugar de cuadradas. En este contexto, se emplea la biblioteca H3 [7], la cual provee un mapeo jerárquico de la superficie terrestre en celdas hexagonales.

El desafío que se plantea consiste en generar, a partir de una celda origen en H3, un conjunto de direcciones en formato ANTop que cubran la superficie discretizada, de manera tal que se conserven las relaciones topológicas de vecindad entre los hexágonos. Esto implica traducir la estructura geométrica provista por H3 a un esquema de direccionamiento compatible con el modelo de hipercubo de ANTop, sin perder consistencia en las distancias ni en las conexiones entre celdas adyacentes.

3.5. Biblioteca H3

3.5.1. Descripción general

H3 [5] [7] es un sistema de indexación geoespacial jerárquico creado por Uber para representar la superficie de la Tierra mediante una teselación (recubrimiento completo de una superficie con figuras geométricas) global de celdas hexagonales. Un sistema de indexación geoespacial es un método estructurado para asignar identificadores a regiones de la superficie terrestre de forma que sea fácil localizar y comparar ubicaciones. La jerarquía significa que existen múltiples niveles de resolución: cada celda en un nivel puede subdividirse en varias celdas más pequeñas en niveles más altos de precisión, lo que permite analizar datos desde escalas globales hasta locales sin perder consistencia en la referencia espacial.

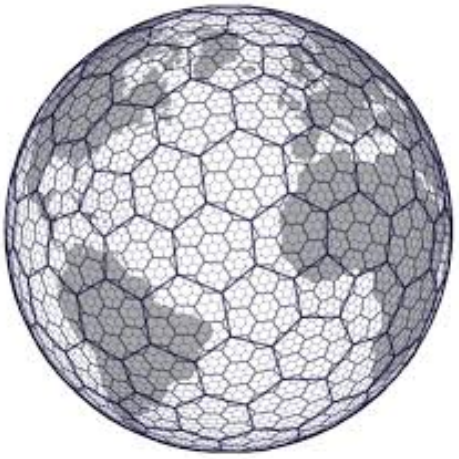

Figura 2: Teselación hexagonal de H3 sobre la superficie terrestre, ilustrando la uniformidad de las celdas y la presencia de pentágonos necesarios para cerrar la superficie.

Para construir esta teselación, H3 usa un icosaedro como estructura intermedia. Un icosaedro es un poliedro regular de veinte caras, donde cada cara es un triángulo equilátero. Este sólido se "inscribe" dentro de la esfera terrestre, lo que significa que sus vértices tocan la superficie de la Tierra. Luego, se proyecta una rejilla de hexágonos sobre cada una de las caras triangulares del icosaedro y dicha rejilla se "mapea" de vuelta a la esfera mediante una proyección gnomónica [11] policéntrica (se utilizan varios centros en vez de uno solo), un tipo de transformación matemática que permite llevar formas planas a la superficie curva de la Tierra con distorsión controlada.

La orientación del icosaedro se basa en la proyección Dymaxion [11] (también llamada proyección de Fuller, por Buckminster Fuller), que es una forma de desplegar la superficie esférica sobre un poliedro de manera que minimiza la distorsión de áreas y mantiene las masas terrestres lo más intactas posibles cuando se proyecta en dos dimensiones.

Debido a una limitación geométrica fundamental, ningún sistema puede cubrir completamente una esfera usando solo hexágonos regulares; por ello, en cada nivel de resolución, H3 incluye exactamente doce celdas pentagonales además de los hexágonos, ubicadas en los vértices del icosaedro. Estos pentágonos (celdas con cinco lados) permiten cerrar los huecos que quedarían si se intentara cubrir toda la superficie con solo hexágonos.

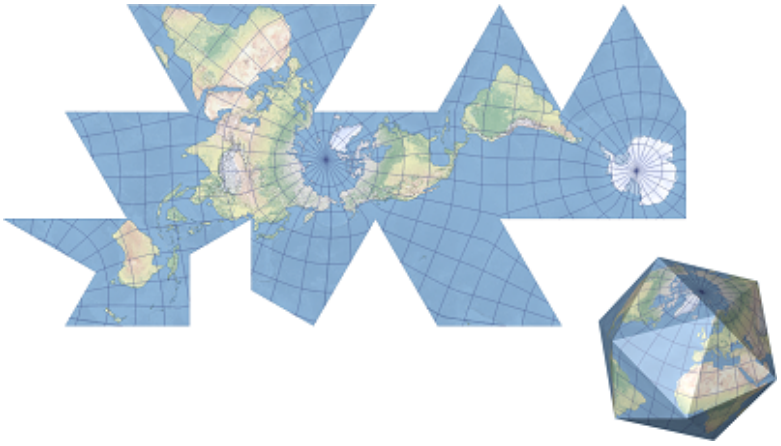

Figura 3: Mapa Dymaxion desplegado y doblado para formar un icosaedro.

La biblioteca emplea un sistema de resoluciones que va de 0 a 15, donde cada número representa un nivel de granularidad diferente en la teselación hexagonal. En la resolución 0, la superficie de la Tierra está dividida en 122 celdas grandes (denominadas *base cells*), y a medida que aumenta el número de resolución, cada celda se subdivide en celdas más pequeñas que cubren la misma área con mayor detalle. Por ejemplo, un hexágono de resolución 0 se subdivide en aproximadamente siete hexágonos de resolución 1, y cada uno de ellos a su vez en otros siete de resolución 2, y así sucesivamente, lo cual permite una división jerárquica y exponencial del espacio. Esto permite adaptar el nivel de detalle del análisis espacial a las necesidades del problema: resoluciones bajas para visión global, y resoluciones altas para análisis muy locales con celdas de área muy reducida.

3.5.2. Sistema de indexación

H3 asigna índices discretos de 64 bits (`H3Index`) a celdas hexagonales y pentagonales que cubren toda la superficie terrestre. Cada `H3Index` codifica tanto el tipo de objeto como la resolución de la celda y una secuencia de dígitos que permite reconstruir su posición dentro de la jerarquía global del sistema. Esta representación compacta y uniforme habilita operaciones eficientes de comparación, navegación jerárquica y análisis espacial en múltiples escalas.

La indexación en H3 se basa en funciones que realizan la conversión entre coordenadas geográficas (latitud y longitud) y estos índices discretos:

- Conversión de coordenadas geográficas a índice H3 — `latLngToCell`: esta función ubica

una coordenada geográfica en la celda H3 que la contiene para una resolución dada. Internamente, el proceso consiste en proyectar las coordenadas geográficas sobre una de las caras del icosaedro subyacente, transformarlas a un sistema cartográfico bidimensional asociado a grillas hexagonales (Hex2d), luego convertir esas coordenadas a un sistema discreto de coordenadas IJK y finalmente codificar toda esta información en un H3Index.

- Conversión de índice H3 a coordenadas geográficas — `cellToLatLng`: A partir de un `H3Index` se obtiene el centro geográfico de la celda correspondiente. El índice se decodifica para identificar la cara del icosaedro y las coordenadas IJK normalizadas en esa cara; estas se proyectan de vuelta a coordenadas Hex2d y, mediante una proyección inversa, a latitud y longitud sobre la esfera terrestre.

Para realizar estas conversiones y otras operaciones espaciales, H3 utiliza una serie de sistemas de coordenadas intermedios que facilitan la representación y manipulación de posiciones dentro de la rejilla hexagonal.

3.5.3. Sistema de coordenadas

En primer lugar, existe IJK: un sistema de coordenadas discretas planas en el que las tres componentes espacialmente correlacionadas (i, j y k) describen posiciones sobre una rejilla hexagonal con ejes separados 120° entre sí. H3 normaliza estas coordenadas para evitar redundancias, lo que permite una representación única dentro de cada cara del icosaedro.

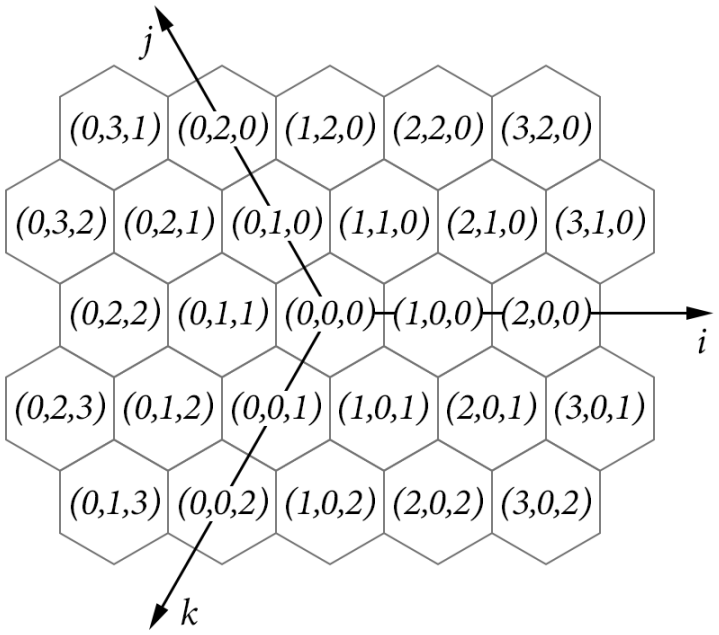

Figura 4: Sistema de coordenadas IJK en H3.

También existen otros tres métodos de indexación: FaceIJK, Hex2d y Local IJ, que combina-

dos al anterior, permiten que H3 ofrezca una indexación espacial robusta, jerárquica y eficiente, integrando la geometría de la esfera terrestre con estructuras discretas manejables computacionalmente. Mediante las funciones de conversión mencionadas, es posible mapear puntos geográficos a índices discretos para su almacenamiento, comparación y análisis, y reconstruir posiciones geográficas a partir de índices cuando sea necesario.

3.5.4. Construcción de direcciones ANTop

Se utiliza la indexación de H3 como base para asignar direcciones ANTop a las celdas que discretizan la superficie terrestre

Las direcciones son números binarios que representan una serie de movimientos *ijk* que permiten ir desde un origen hasta una celda en cuestión. Tomando como ejemplo la figura 4, se elige como origen a la celda ubicada en $(0, 0, 0)$. La dirección para esa celda serían todos ceros, dado que no hay que moverse en ninguna dirección para llegar a sí misma.

Si en cambio se quisiese construir la dirección para la celda $(2, 1, 0)$, para llegar desde el origen con el camino más corto, hay que realizar tres movimientos: dos en i hacia la derecha y uno en j hacia arriba, lo cual se puede descomponer en: $(1, 0, 0) + (1, 0, 0) + (0, 1, 0)$.

Dicha descomposición es la que se codifica en la dirección, donde cada posición es un bit, quedando entonces: 100 100 010. A su vez, cada movimiento puede ser traducido en un número entero del uno al seis. En este caso, 4, 4 y 2, respectivamente.

Debido a que la cantidad de celdas a las cuales se les debe asignar direcciones es extensa, incluso en resolución 1, las direcciones son comprimidas para que ocupen la menor cantidad de bits posibles. Dicha compresión consiste en almacenar hasta dos movimientos en un byte, completando con ceros los bits restantes. Retomando el ejemplo de la celda $(2, 1, 0)$, su dirección es codificada tal que:

1. Se codifica el primer $(1, 0, 0)$ en los últimos tres bits del primer byte de la dirección.
2. Se codifica el segundo $(1, 0, 0)$ en los siguientes tres bits del primer byte de la dirección.
3. Debido a que restan solamente dos bits en el primer byte, es necesario asignar un nuevo byte para continuar codificando.
4. Se codifica $(0, 1, 0)$ en los últimos tres bits del segundo byte de la dirección.
5. Como ya no se deben codificar más movimientos, la dirección queda codificada como la secuencia de los bytes, con aquellos bits inutilizados en 0: 00100100 00000010.

Este mismo ejemplo puede verse representado paso a paso en la Figura 5, al igual que una implementación del proceso en el Algoritmo 5. El paso 1 transforma la coordenada de entrada en su correspondiente dígito direccional, mientras que los pasos 2 a 4 validan que dicho valor obtenido corresponda a una dirección válida (valor entre 1 y 6, ya que un hexágono tiene como máximo 6 desplazamientos posibles).

Una vez validado el movimiento, el algoritmo determina cómo debe almacenarse dentro del arreglo de bytes `data`. Para ello se evalúa la paridad de la variable `size`, la cual indica la cantidad total de movimientos ya almacenados. Si `size` es impar, significa que el último byte aún puede contener un movimiento adicional. De lo contrario, es necesario crear un nuevo byte. A la hora de agregar un movimiento a un byte ya existente (paso 9), resulta necesario realizar una operación de shift a la izquierda, con el objetivo de que el nuevo movimiento quede alineado con la segunda posición de almacenamiento dentro del byte.

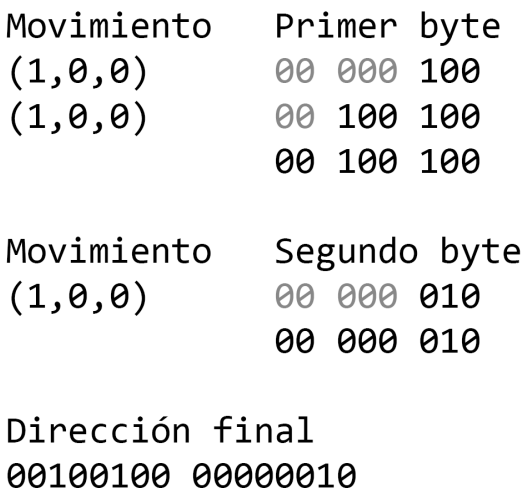

Figura 5: Ejemplo de algoritmo de compresión de direcciones ANTop

Algorithm 5 `Address.push(coord)`

Require: CoordIJK `coord`

1: `direction` ← `unitIjkToDigit(coord)`
2: **if** `direction` $\notin \{1, \ldots, 6\}$ **then**
3: **return**
4: **end if**
5: **if** `isEven(size)` **then**
6: `data[len]` ← `direction`
7: `len` ← `len` + 1
8: **else**
9: `data[len - 1]` ← `data[len - 1]` + (`direction` $\ll 3$)
10: **end if**
11: `size` ← `size + 1`

3.5.5. Definición del origen y partición del espacio

Inicialmente, se consideró la utilización de un único sistema de coordenadas local basado en el esquema *ijk* de H3, extendido progresivamente para cubrir la totalidad de las celdas de

la resolución seleccionada. Sin embargo, debido a la naturaleza geométrica del sistema H3 y a su proyección sobre la superficie esférica de la Tierra, este enfoque introduce distorsiones acumulativas a medida que se incrementa la distancia respecto del origen del hipercubo. En particular, la presencia de celdas pentagonales genera discontinuidades en la orientación de los ejes locales, lo que provoca rotaciones implícitas del sistema de coordenadas *ijk* y afecta la consistencia de los cálculos de distancia entre celdas alejadas del origen.

Para mitigar estos efectos, se optó por dividir el espacio discretizado en múltiples hipercubos independientes, cada uno con su propio sistema de coordenadas local. Cada hipercubo se define tomando como origen una celda pentagonal. Esta elección se fundamenta en varias propiedades relevantes de los pentágonos dentro del sistema H3. En primer lugar, cada pentágono se encuentra centrado respecto de su base cell, ya que los pentágonos coinciden con los vértices del icosaedro subyacente y su base cell es, a su vez, un pentágono. En segundo lugar, para cualquier resolución dada, existe una cantidad finita y determinística de pentágonos, específicamente doce. Finalmente, estos pentágonos se encuentran distribuidos de manera aproximadamente uniforme sobre la superficie terrestre, lo que permite una partición equilibrada del espacio.

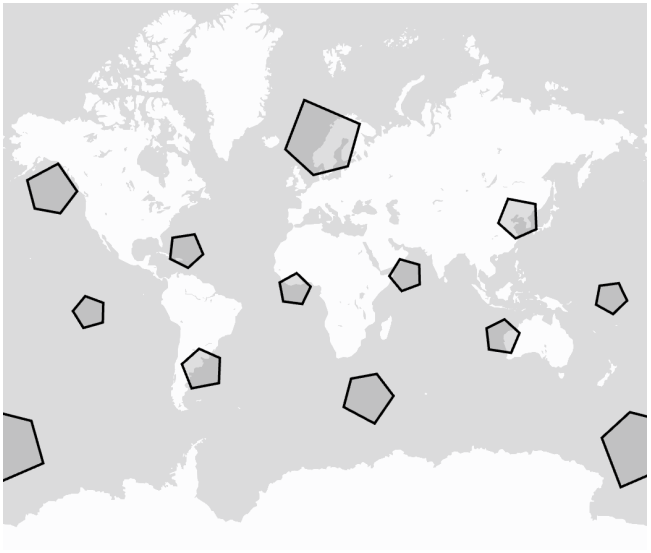

Figura 6: Los doce pentágonos de H3 en resolución cero.

Que la celda origen se encuentre centrada respecto de su base cell resulta especialmente relevante para garantizar la coherencia entre sistemas de coordenadas. En el esquema de coordenadas *ijk* de H3, una celda que se encuentra centrada en su base cell presenta, respecto de sí misma, un vector de coordenadas *ijk* igual a $(0, 0, 0)$. Esta propiedad permite una correspondencia directa con el sistema de coordenadas global adoptado en el algoritmo de creación de direcciones.

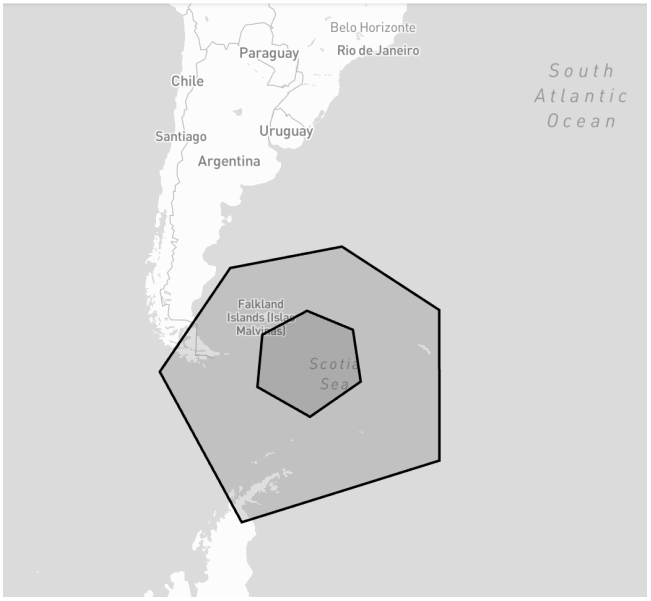

Figura 7: Ejemplo de posible origen en H3: hexágono ubicado en el centro de su base cell.

En contraste, si se seleccionara como origen una celda que no estuviese centrada respecto de su base cell, el valor de salida del sistema *ijk* para dicha celda podría ser distinto de $(0, 0, 0)$, por ejemplo $(1, 0, 0)$. Esta discrepancia introduce un desplazamiento artificial entre el sistema de coordenadas local de H3 y el sistema global del algoritmo, lo que complica la interpretación y el uso de las coordenadas, y puede derivar en distorsiones adicionales en los cálculos de distancia y enrutamiento.

Mediante este enfoque, se limita el alcance espacial de cada sistema de coordenadas local, reduciendo la influencia de las distorsiones geométricas asociadas a la rotación de los ejes *ijk*. Como resultado, los cálculos de distancia y las operaciones de enrutamiento se realizan dentro de dominios donde la geometría local se mantiene más estable, mejorando la consistencia y robustez general de la simulación.

3.5.6. Dimensiones

La Figura 4 ilustra el sistema de coordenadas locales *ijk* utilizado por H3. Este sistema define tres ejes sobre la grilla hexagonal, permitiendo expresar la posición de cada celda como un desplazamiento discreto respecto de un origen. Sin embargo, debido a que cada hexágono posee seis vecinos pero el sistema *ijk* sólo dispone de tres ejes, no todas las direcciones de vecindad coinciden con un eje del sistema. Como consecuencia, existen desplazamientos físicamente adyacentes que requieren más de un desplazamiento en el sistema *ijk*.

Este fenómeno puede ser observado en la Figura 8, donde se destacan dos celdas. La celda resaltada en verde se encuentra a un salto físico de distancia del origen, mientras que la celda celeste se encuentra a dos saltos físicos. No obstante, sus direcciones en el sistema *ijk* son $(1, 0, 1)$

y $(2, 1, 0)$ respectivamente. En términos de desplazamientos en el sistema *ijk*, estas celdas se encuentran a distancia dos y tres del origen, lo que evidencia que la utilización de un único sistema de coordenadas no refleja fielmente la conectividad hexagonal subyacente.

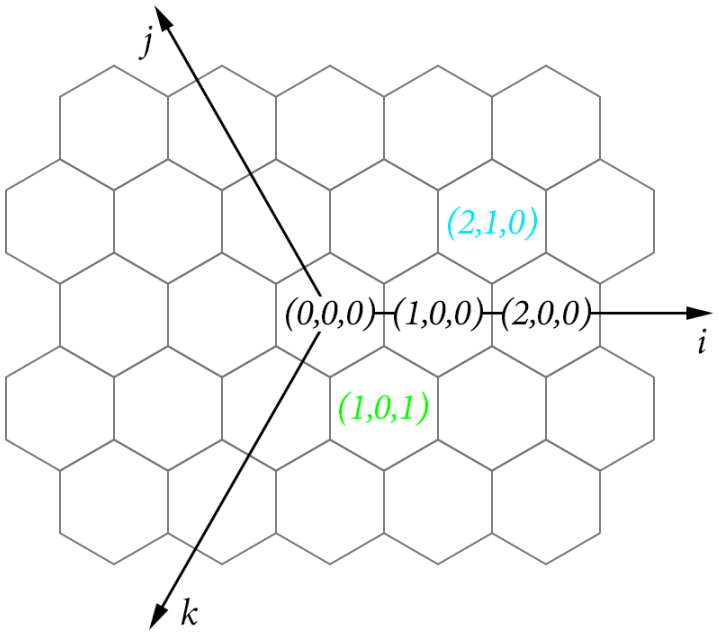

Figura 8: Sistema de coordenadas locales *ijk* de H3. Se resaltan dos celdas: una adyacente al origen (verde) y otra a dos saltos físicos (celeste).

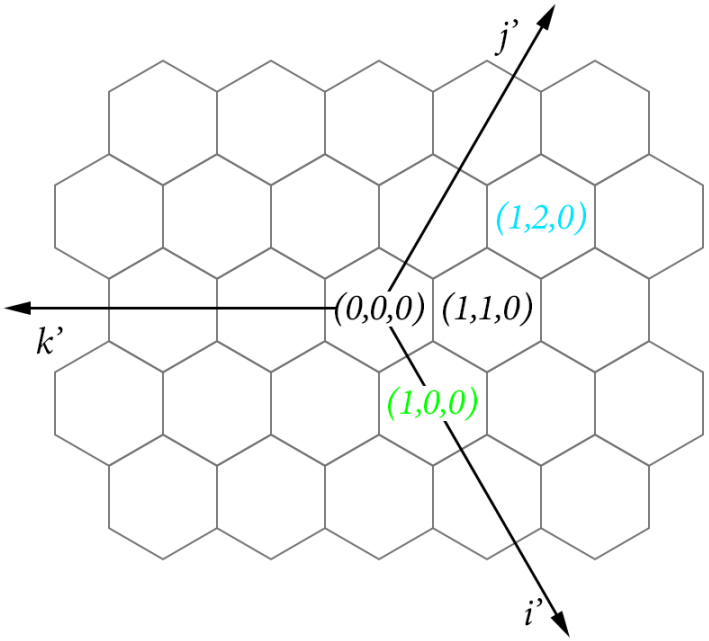

Figura 9: Sistema de coordenadas *ijk* rotado 60° (dimensión prima). Las mismas celdas de la Figura 8 son representadas con direcciones distintas.

Para reducir esta discrepancia geométrica, se introduce una segunda dimensión de direccionamiento construida a partir de una rotación de 60° en sentido horario de los ejes del sistema *ijk*. La Figura 9 muestra este sistema rotado, denominado dimensión prima. En esta nueva orientación, las mismas celdas destacadas adoptan direcciones distintas: la celda verde pasa a representarse como $(1, 0, 0)$, posibilitando llegar desde el origen a esta en un solo desplazamiento, mientras que la celda celeste adopta la dirección $(1, 2, 0)$.

Al igual que el sistema *ijk* original, la dimensión prima dispone únicamente de tres ejes, por lo que continúan existiendo direcciones de vecindad que no coinciden con un eje del sistema. Es por esta razón que nuevamente resulta imposible alcanzar la celda celeste desde el origen en solo dos desplazamientos. No obstante, ambos sistemas definen, en su conjunto, seis direcciones independientes de desplazamiento, equivalentes a las seis direcciones de adyacencia de la grilla hexagonal.

En la dimensión principal (Figura 8), es posible avanzar desde el origen hasta la celda intermedia de dirección $(1, 0, 0)$. Sin embargo, continuar el desplazamiento dentro de esta dimensión requeriría dos saltos adicionales para alcanzar a la celda celeste (dirección $(2, 1, 0)$). No obstante, si se considera la dimensión prima (Figura 9), dicha celda intermedia (ahora dirección $(1, 1, 0)$) está a distancia uno de la celda celeste (dirección $(1, 2, 0)$). Este cambio de dimensión permite alcanzar el destino en un único salto adicional.

Cada una de estas dimensiones define un hipercubo de direccionamiento independiente dentro del esquema de ANTop. En consecuencia, las direcciones pertenecientes a distintas dimensiones

no son directamente comparables entre sí, ya que cada hipercubo constituye una estructura algebraica autónoma que modela la misma topología física desde una orientación distinta.

Cuando una celda dispone de direcciones en múltiples hipercubos, como ocurre en el esquema propuesto, un nodo puede utilizar cualquiera de estas representaciones para realizar un salto hacia un vecino. Desde el punto de vista físico, el desplazamiento es único ya que lo que varía es el sistema de coordenadas empleado para representarlo.

En síntesis, la introducción de una dimensión rotada no busca redefinir la estructura de la grilla hexagonal, sino proporcionar una representación complementaria al sistema *ijk*. La combinación de ambos sistemas permite aproximar de forma más precisa la conectividad real entre celdas y reducir la longitud de los caminos utilizados por el protocolo de ruteo.

3.5.7. Algoritmo

El algoritmo de creación de direcciones se basa en un recorrido expansivo del grafo de celdas H3, comenzando desde una celda origen y propagándose hacia sus vecinos inmediatos utilizando una estrategia de búsqueda en anchura (BFS). El objetivo es asignar a cada celda una o más direcciones de forma consistente, garantizando unicidad, vecindad lógica y compatibilidad con la topología hexagonal subyacente.

Para cada hipercubo, a la celda origen se le asignan inicialmente las direcciones base correspondientes a las dimensiones principal y prima. Estas direcciones constituyen el punto de partida del proceso de propagación y se almacenan tanto en la estructura interna de la celda como en una tabla global que mantiene la correspondencia entre direcciones y celdas.

A continuación, se inicia el recorrido BFS. Para cada celda visitada, se obtienen sus celdas vecinas utilizando las primitivas provistas por la API de H3 (`gridDisk`) [4]. Para cada vecino válido que aún no haya sido procesado, se calculan sus coordenadas relativas respecto de la celda origen en el sistema de coordenadas locales *ijk*. Este cálculo se realiza mediante la función `cellToLocalIjk`, permitiendo expresar la posición del vecino como un desplazamiento discreto sobre la grilla hexagonal.

A partir de este desplazamiento se generan dos variantes: la original y una versión rotada 60° en sentido horario. Cada una de estas variantes se utiliza para extender la dirección correspondiente a una de las dimensiones, generando así dos direcciones candidatas para la celda vecina. Estas direcciones se obtienen copiando la dirección base del hipercubo correspondiente y agregando el desplazamiento *ijk* calculado.

Previo a la asignación, cada dirección candidata es validada mediante un proceso de verificación que asegura dos propiedades fundamentales: (i) que la dirección no haya sido asignada previamente a otra celda, y (ii) que cualquier dirección que difiera en un único bit corresponda efectivamente a una celda vecina en la topología H3. Esta validación garantiza que la distancia lógica entre direcciones refleje correctamente la vecindad espacial entre celdas.

Si ambas direcciones candidatas resultan válidas, se asignan a la celda vecina, la cual se

incorpora a la cola del BFS, permitiendo que el proceso continúe de manera expansiva hasta cubrir todas las celdas requeridas.

Una vez finalizada la asignación de direcciones base, el algoritmo realiza una segunda pasada destinada a la creación de direcciones suplementarias. Esta etapa tiene como objetivo compensar la pérdida de vecindad que se produce en los hexágonos ubicados en los bordes de la proyección. Debido a los límites del dominio, algunas celdas no disponen del conjunto completo de vecinos definido por la topología H3, lo que implica que queden con menos vecinos que las celdas interiores.

Para mitigar este efecto, se consideran pares de celdas que no son vecinas directas en la topología H3, pero que pueden alcanzarse mediante combinaciones válidas de desplazamientos *ijk*. En particular, se analizan aquellas configuraciones donde la falta de vecindad se debe únicamente a la condición de borde. Para cada una de estas celdas, el algoritmo intenta extender las direcciones existentes respetando las restricciones de direccionalidad y unicidad previamente definidas. Cuando se identifica una dirección válida, esta se asigna a la celda correspondiente, permitiendo así restaurar la conectividad que se pierde en los límites de la proyección.

Finalmente, con todas las direcciones asignadas, se construye explícitamente el grafo de vecindad entre celdas. Para cada celda, se almacenan únicamente aquellos vecinos cuya distancia topológica es exactamente uno.

Este procedimiento asegura que cada celda obtenga un conjunto de direcciones que preserva la estructura hexagonal de H3, mantiene relaciones de vecindad consistentes y habilita un enrutamiento eficiente y determinista sobre la topología resultante.

4. DTNSim

DTNSim es un simulador orientado al estudio de distintos aspectos de las redes tolerantes a demoras/interrupciones (Delay/Disruption Tolerant Networks, DTN), tales como el enrutamiento, el reenvío, la planificación, o la programación, entre otros. El simulador está implementado sobre el framework OMNeT++ versión 6.x y puede utilizarse de manera conveniente mediante el entorno Qtenv, así como modificarse o extenderse utilizando el IDE basado en Eclipse que provee OMNeT++.

DTNSim modela la dinámica de la red a través de planes de contacto (contact plans), los cuales describen explícitamente las oportunidades de comunicación entre nodos en intervalos de tiempo determinados. De este modo, la movilidad de los nodos no se representa mediante desplazamientos continuos en el espacio, sino mediante la disponibilidad temporal de enlaces, lo cual resulta especialmente adecuado para redes satelitales, interplanetarias o altamente dinámicas.

A su vez, permite ejecutar escenarios configurables mediante archivos *ini*, recolectando estadísticas tanto escalares como vectoriales, que luego pueden ser utilizadas para el análisis de desempeño de los protocolos evaluados. Entre los protocolos soportados se incluye CGR.

4.1. Arquitectura

DTNSim se organiza en una arquitectura modular que modela explícitamente los distintos componentes funcionales de una red de tipo DTN. La estructura general del simulador se define mediante archivos NED, que describen tanto los módulos como sus interconexiones y parámetros.

4.1.1. Nivel superior

En el nivel más alto de abstracción, el simulador modela una red DTN completa, denominada *Dtnsim*, compuesta por un conjunto de nodos autónomos y un módulo central de soporte. Esta estructura permite separar claramente la lógica distribuida de la red de aquellas funcionalidades transversales a toda la simulación. Dicha red incluye:

- Un módulo *Central*, que actúa como entidad de coordinación global.
- Un conjunto de módulos *Node*, donde cada instancia representa un nodo DTN independiente.

El módulo *Central* cumple funciones auxiliares al modelo, tales como la recolección de métricas globales, la supervisión del estado de la simulación, la gestión de eventos especiales o la inyección de fallos. Su interacción con los nodos es de naturaleza lógica y no representa enlaces de comunicación físicos dentro de la red DTN simulada.

Esta separación permite que el comportamiento de la red emerja principalmente de la interacción entre los nodos, mientras que el módulo central actúa como un observador y coordinador externo, sin introducir dependencias artificiales en el funcionamiento distribuido del sistema.

4.1.2. Módulo Node

Cada nodo de la red DTN se modela mediante el módulo compuesto *Node*, el cual representa una pila funcional completa de un nodo participante en una red tolerante a retardos. Internamente, el nodo se organiza en submódulos que reflejan una separación por responsabilidades, alineada con la arquitectura lógica del simulador más que con capas OSI tradicionales.

Los principales submódulos que componen un nodo son:

- **App**: módulo de aplicación, responsable de generar y consumir datos a nivel de bundle. Representa el origen y destino lógico del tráfico en la red DTN.
- **Dtn**: núcleo del nodo, encargado de la lógica propia de las redes DTN. Este módulo gestiona el almacenamiento temporal de bundles, la toma de decisiones de enrutamiento y el reenvío de datos en función del estado de la red y de la información disponible.
- **Com**: módulo de comunicación, que abstrae el proceso de transmisión entre nodos mediante una capa de contacto lógica. Este módulo modela la disponibilidad temporal de enlaces y su capacidad de transmisión, incorporando retardos y restricciones de tasa, sin representar explícitamente detalles físicos ni de enlace tradicionales.
- **Graphics**: módulo opcional destinado a la visualización del estado interno del nodo durante la simulación.
- **Fault**: módulo opcional utilizado para la simulación de fallos en el nodo.

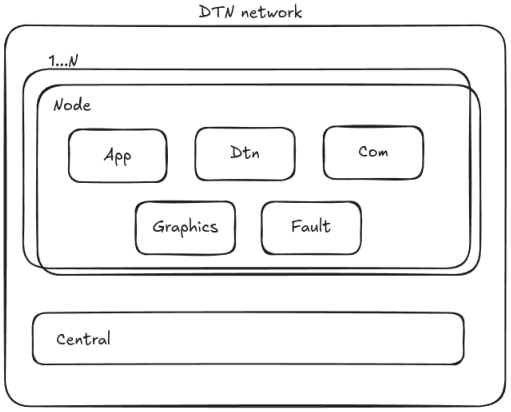

Figura 10: Arquitectura de módulos original de DTNsim.

Desde el punto de vista del flujo de información, los bundles generados por el módulo *App* son entregados al módulo *Dtn*, que decide su almacenamiento, encaminamiento o reenvío. Cuando un bundle debe ser transmitido a otro nodo, el módulo *Dtn* delega dicha operación en el módulo *Com*, el cual se encarga de modelar la transmisión respetando las restricciones temporales y de capacidad del contacto.

Esta organización permite desacoplar claramente la lógica de enrutamiento y toma de decisiones del modelado de la transmisión, facilitando la experimentación con distintos algoritmos de routing sin modificar los mecanismos de comunicación subyacentes.

4.2. Contactless DTN

El módulo *Dtn* original se encuentra diseñado en torno al concepto de *contact plans*, los cuales describen de forma explícita los intervalos temporales durante los cuales existen oportunidades de comunicación entre pares de nodos. Este enfoque resulta adecuado para escenarios donde la conectividad es conocida de antemano o puede ser precomputada, como ocurre típicamente en redes DTN estáticas o con movilidad determinista.

Sin embargo, el funcionamiento del protocolo ANTop presenta requerimientos distintos, ya que se basa en la utilización del sistema H3, el cual depende directamente de posiciones geográficas reales para determinar la vecindad entre nodos. En este contexto, la conectividad entre nodos no se encuentra definida de manera explícita mediante planes de contacto, sino que emerge dinámicamente como consecuencia del movimiento de los satélites y de su proximidad espacial.

Con el objetivo de superar estas limitaciones, se desarrolló un nuevo módulo denominado *ContactlessDtn*. Este módulo desacopla el funcionamiento del módulo *Dtn* de los *contact plans* tradicionales y permite que la conectividad entre nodos sea determinada dinámicamente a partir de sus posiciones geográficas.

De esta manera, cada nodo mantiene una representación explícita de su ubicación en el espacio, la cual puede variar a lo largo del tiempo siguiendo modelos de movilidad predefinidos, y es a partir de estas posiciones que se evalúa de forma continua la conectividad entre nodos.

4.3. Movilidad

Dado que el protocolo ANTop depende directamente de la ubicación geográfica de los satélites para determinar la ocupación de celdas H3 y la conectividad entre nodos, la elección de un modelo de movilidad resulta un aspecto central del diseño del simulador.

El modelo de movilidad adoptado es el de SGP4 (*Simplified General Perturbations 4*) [13], el cual es ampliamente utilizado para la propagación de órbitas satelitales en escenarios de baja órbita terrestre (LEO). SGP4 permite calcular la posición y velocidad de un satélite como función del tiempo a partir de un conjunto reducido de parámetros orbitales, ofreciendo una representación suficientemente precisa del movimiento orbital para aplicaciones de simulación.

4.3.1. OS^3

OS^3 (*The Open Source Satellite Simulator*) [10] es un simulador orientado a redes satelitales que extiende las capacidades de OMNeT++ mediante la integración de modelos de movilidad orbital y componentes de comunicación.

Dentro de OS^3, los modelos de movilidad se implementan como módulos independientes que actualizan la posición de los nodos en función del tiempo de simulación. Estos módulos se basan en distintos modelos dinámicos, entre ellos SGP4, y se integran de manera transparente con el resto del simulador, permitiendo que los protocolos de red accedan a la información de posición sin acoplarse directamente al modelo orbital subyacente.

Los modelos de movilidad de OS^3 requieren ser configurados a partir de archivos *Two-Line Element* (TLE), los cuales constituyen el formato estándar para describir órbitas satelitales reales, y permiten reproducir constelaciones existentes con alto grado de fidelidad.

No obstante, la utilización de archivos TLE introduce una limitación desde el punto de vista de la flexibilidad experimental ya que restringe la simulación a configuraciones orbitales previamente definidas y dificulta la generación de escenarios ficticios y parametrizables, necesarios para evaluar de forma sistemática el impacto de distintos parámetros de densidad, cobertura y distribución espacial sobre el comportamiento del protocolo ANTop.

Es en función de estas consideraciones que se decidió adoptar una alternativa que permitiera definir constelaciones satelitales de manera completamente parametrizable, sin depender de archivos TLE.

4.3.2. Atlas

Con el objetivo de superar las restricciones mencionadas, se optó por utilizar *Atlas* [12], un simulador desarrollado sobre OS^3 que permite definir constelaciones satelitales de manera completamente paramétrica. Este entorno introduce un módulo específico, denominado *NORAD*, que actúa como generador de órbitas sintéticas y permite instanciar constelaciones completas a partir de un conjunto reducido de parámetros.

Entre los parámetros soportados se incluyen el intervalo de actualización (*updateInterval*), la cantidad total de satélites (*numOfSats*), el número de planos orbitales (*planes*), la cantidad de satélites por plano (*satPerPlane*), la altitud orbital (*altitude*) y la inclinación respecto al Ecuador (*inclination*). Este enfoque elimina la necesidad de utilizar archivos TLE y permite construir escenarios altamente configurables, facilitando la exploración de distintas topologías y densidades de nodos.

4.3.3. Phase Offset

El *phase offset* es un parámetro que define el desfase temporal entre satélites pertenecientes a planos orbitales consecutivos. En particular, indica el instante relativo en el que los satélites de distintos planos cruzan el ecuador, y toma valores normalizados entre cero y uno [8].

Un valor de *phase offset* igual a cero implica que el satélite n de un plano orbital cruza el ecuador simultáneamente con el satélite n del plano siguiente. En el extremo opuesto, un valor igual a uno indica que el satélite n de un plano cruza el ecuador al mismo tiempo que el satélite $n + 1$ del plano adyacente. Valores intermedios permiten distribuir los satélites de manera progresiva a lo largo de la órbita, mejorando la uniformidad espacial de la constelación.

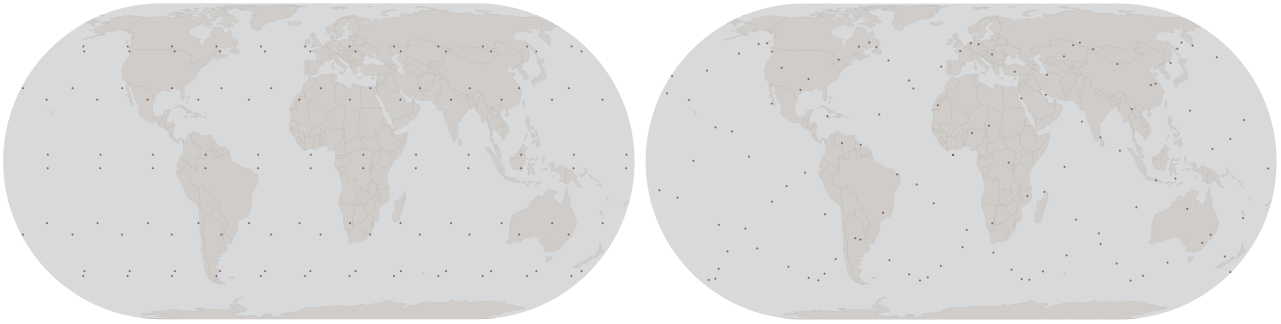

Figura 11: Posicionamiento inicial de los satélites en una constelación Walker: (izquierda) sin phase offset, (derecha) con phase offset distinto de cero.

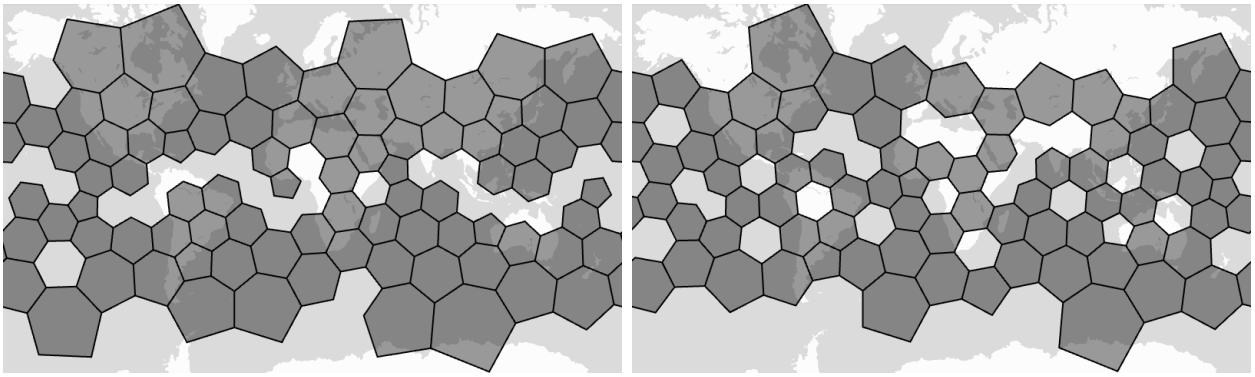

Figura 12: Cobertura de celdas H3 correspondiente a la constelación Walker: (izquierda) sin phase offset, (derecha) con phase offset distinto de cero.

La implementación original de *Atlas* no incluye soporte para *phase offset*, el cual es necesario para modelar correctamente constelaciones del tipo Walker. Por esta razón, se extendió la implementación existente para incorporar dicho parámetro, permitiendo la generación de constelaciones Walker completas y realistas.

En el contexto de las simulaciones con ANTop, esta propiedad es fundamental para asegurar que la mayor cantidad posible de celdas H3 se encuentren ocupadas por al menos un nodo durante la simulación. Una ocupación más uniforme del espacio contribuye a mejorar la conectividad de la topología resultante y a reducir los casos en los que el algoritmo de ruteo se ve limitado por la ausencia de nodos en determinadas regiones.

4.4. Modelo de tolerancia a fallas

Con el objetivo de evaluar la robustez del sistema frente a desconexiones de nodos, se implementó un modelo de tolerancia a fallas de tipo estocástico, en el cual cada nodo de la red alterna entre estados operativos y estados de falla. Este enfoque permite representar de manera realista el comportamiento intermitente de nodos en redes satelitales, donde las fallas pueden

producirse de forma impredecible y recuperarse automáticamente tras cierto período de tiempo.

En el contexto de las simulaciones con ANTop, esta propiedad es fundamental para asegurar que la mayor cantidad posible de celdas H3 se encuentren ocupadas por al menos un nodo. Una ocupación más uniforme del espacio discretizado contribuye a mejorar la conectividad de la topología resultante y a reducir los casos en los que el algoritmo de ruteo se ve limitado por la ausencia de nodos en determinadas regiones.

Como se observa en la Figura 11, la introducción de un phase offset distinto de cero produce una distribución más uniforme de los satélites a lo largo de los planos orbitales. Este efecto se refleja directamente en la ocupación de celdas H3, donde la Figura 12 evidencia una cobertura global más homogénea, principalmente alrededor de la línea del ecuador.

4.4.1. Modelo de estados

Cada nodo puede encontrarse en uno de los siguientes estados:

- **Estado operativo**: el nodo funciona normalmente y puede transmitir, recibir y reenviar bundles.
- **Estado de falla**: el nodo se considera inoperativo y sus funcionalidades de comunicación se encuentran deshabilitadas.

Las transiciones entre estos estados se modelan mediante dos variables aleatorias:

- **Tiempo hasta la falla** (Time To Failure, TTF).
- **Tiempo de reparación** (Time To Repair, TTR).

Ambas variables se modelan mediante distribuciones exponenciales, caracterizadas por sus valores medios:

$$\mathbb{E}[TTF] = \text{MTTF} \tag{1}$$
$$\mathbb{E}[TTR] = \text{MTTR} \tag{2}$$

La elección de la distribución exponencial se debe a su propiedad de falta de memoria, lo cual implica que la probabilidad de que ocurra una falla es independiente del tiempo transcurrido desde la última transición.

4.4.2. Modelo matemático

Sean las variables aleatorias T_f y T_r correspondientes al tiempo hasta la falla y al tiempo de reparación, respectivamente. Estas se definen como:

$$T_f \sim \text{Exponencial}(\lambda_f) \tag{3}$$
$$T_r \sim \text{Exponencial}(\lambda_r) \tag{4}$$

donde los parámetros de las distribuciones están dados por:

$$\lambda_f = \frac{1}{\text{MTTF}} \tag{5}$$
$$\lambda_r = \frac{1}{\text{MTTR}} \tag{6}$$

De esta forma, las funciones de densidad de probabilidad quedan definidas como:

$$f_{T_f}(t) = \lambda_f e^{-\lambda_f t}, \quad t \geq 0 \tag{7}$$
$$f_{T_r}(t) = \lambda_r e^{-\lambda_r t}, \quad t \geq 0 \tag{8}$$

4.4.3. Porcentaje esperado de tiempo en falla

En régimen estacionario, el porcentaje esperado de tiempo que un nodo permanece en estado de falla se obtiene a partir de la relación entre los tiempos medios de operación y reparación. Dado un ciclo completo compuesto por un período operativo seguido de un período de falla, la fracción de tiempo en falla P_f está dada por:

$$P_f = \frac{\text{MTTR}}{\text{MTTF} + \text{MTTR}} \tag{9}$$

Esta expresión permite configurar el sistema para simular distintos niveles de fallas promedio, ajustando adecuadamente los valores de MTTF y MTTR.

4.4.4. Implementación basada en eventos

La implementación se realizó utilizando mensajes (*self-messages*) provistos por el simulador OMNeT++. Cada nodo crea un único mensaje de control de fallas durante su fase de inicialización, el cual se agenda para provocar la primera transición al estado de falla luego de un tiempo aleatorio generado según una distribución exponencial con media MTTF.

Cuando dicho mensaje es recibido, el nodo transiciona al estado de falla y el mismo mensaje es reutilizado para programar el evento de recuperación, cuyo tiempo se obtiene a partir de una distribución exponencial con media MTTR. Finalizado el período de reparación, el mensaje vuelve a reprogramarse para la siguiente falla.

Este mecanismo genera una secuencia alternante de eventos de falla y recuperación, manteniendo un único mensaje activo por nodo y evitando sobrecarga innecesaria en la simulación.

4.4.5. Independencia y escalabilidad

El modelo de fallas se aplica de forma independiente a cada nodo de la red, lo cual permite capturar escenarios en los que las fallas no se encuentran sincronizadas. Esta característica resulta especialmente relevante en redes satelitales.

4.5. Scripts auxiliares

Con el fin de automatizar el proceso de compilación, generación de escenarios experimentales y análisis de resultados, se desarrolló un conjunto de scripts que acompañan la implementación del simulador.

4.5.1. Generación de escenarios de simulación

El script `generate-antop.py` se encarga de generar automáticamente los archivos de configuración (`.ini`) utilizados por OMNeT++. A partir de parámetros ingresados por el usuario —como la cantidad de planos orbitales, el número de satélites por plano y el desfase entre planos— el script construye una constelación satelital completa.

Asimismo, permite habilitar la funcionalidad de fallas en los nodos de la red, configurando distintos porcentajes mediante la parametrización de los tiempos medios entre fallas (MTTF) y de recuperación (MTTR). De este modo, es posible evaluar el comportamiento de los algoritmos de ruteo bajo distintos niveles de degradación de la red.

Además, se genera tráfico aleatorio entre los nodos de la constelación, definiendo para cada origen múltiples flujos de bundles con destinos, tiempos de inicio y tamaños aleatorios, manteniendo constante la cantidad total de bundles enviados por nodo.

Finalmente, el script configura la ruta de salida donde se almacenarán las métricas recolectadas durante la simulación en formato JSON, facilitando su posterior análisis.

4.6. Contact plans a partir de Atlas

Para poder evaluar experimentalmente el desempeño de ANTop y CGR bajo condiciones comparables, resulta necesario ejecutar ambos protocolos sobre escenarios de simulación equivalentes. Sin embargo, cada uno de estos protocolos modela la conectividad de manera diferente: mientras que CGR utiliza *contact plans*, que describen explícitamente los intervalos de tiempo durante los cuales dos nodos pueden comunicarse, ANTop depende de la posición geoespacial de los satélites para determinar dinámicamente la existencia de enlaces.

Debido a esta diferencia en la representación de la conectividad, fue necesario establecer un mecanismo que permitiera traducir el movimiento orbital de los satélites en un conjunto de *contact plans* equivalentes. Para ello, se desarrolló un escenario de simulación independiente cuyo objetivo es generar planes de contacto a partir del mismo modelo de movimiento satelital utilizado en las simulaciones de ANTop. De esta forma, los *contact plans* obtenidos reflejan

exactamente las mismas condiciones de conectividad que surgen de la simulación basada en posiciones, permitiendo que los experimentos con CGR y ANTop se ejecuten sobre escenarios estrictamente equivalentes.

En este contexto, se desarrolló la red `csm`, la cual se ejecuta como una simulación independiente del simulador DTN principal, durante un horizonte temporal configurable y con una cantidad parametrizable de nodos satelitales. La red `csm` contiene un conjunto de nodos del tipo `ContactMobilityNode` los cuales representan un satélite, y encapsulan los componentes necesarios para modelar el movimiento orbital de los mismos.

4.6.1. Extensión del modelo de movilidad satelital

Cada nodo `ContactMobilityNode` está compuesto por un módulo `ContactSatelliteMobility`, el cual es una extensión del módulo `SatelliteMobility` original, encargado de computar el movimiento orbital de un satélite a lo largo de una simulación. La implementación de dicha extensión agrega la lógica necesaria para convertir dicho movimiento orbital en contactos de un *contact plan*. En cada actualización de movilidad, el módulo `ContactSatelliteMobility`:

1. Obtiene la posición geográfica actual del satélite (latitud y longitud).
2. Mapea dicha posición a su correspondiente celda H3.
3. Calcula el conjunto de celdas vecinas.
4. Almacena en forma de contacto a aquellos satélites que se encuentran en alguna de estas celdas vecinas. Este criterio es consistente con el modelo de conectividad utilizado por ANTop, en el cual los enlaces se establecen entre satélites ubicados en celdas H3 adyacentes.

Dos satélites se consideran en contacto si, durante un intervalo de tiempo, ambos se encuentran en celdas H3 vecinas.

4.6.2. Construcción del plan de contactos

Para cada par de satélites detectado en contacto, el módulo registra intervalos temporales discretos $[from, to]$ durante los cuales la vecindad se mantiene.

Al finalizar la simulación, cada nodo contribuye a la construcción de un archivo de plan de contactos. Este archivo contiene, para cada par de nodos, dos entradas simétricas (una por cada dirección), indicando:

- Tiempo de inicio del contacto.
- Tiempo de finalización del contacto.
- Nodo origen.
- Nodo destino.

- Tasa de transmisión asociada al contacto.

El formato generado es compatible con los planes de contacto utilizados por los algoritmos de ruteo CGR.

De esta forma, la simulación `csm` actúa como una etapa previa de preprocesamiento, generando planes de contacto que pueden ser utilizados por el simulador DTN para evaluar distintos algoritmos de ruteo bajo condiciones comparables.

4.7. Métricas

Con el objetivo de asegurar una comparación consistente entre los distintos escenarios simulados, se desarrolló sobre una clase preexistante denominada `MetricCollector`, encargada de centralizar la recolección y el almacenamiento de las métricas de desempeño.

Esta clase mantiene, para cada bundle identificado de forma única, un conjunto de estructuras que permiten registrar información relevante durante su procesamiento, como se explicará en la siguiente sección.

5. Simulaciones

Para la evaluación de los algoritmos de enrutamiento considerados en este trabajo se definió un marco experimental estructurado en torno a cuatro dimensiones principales de análisis. Estas dimensiones permiten organizar las simulaciones realizadas, así como la presentación y discusión de los resultados obtenidos.

5.1. Métricas de desempeño

La primera dimensión del análisis corresponde a las métricas utilizadas para evaluar el comportamiento de los algoritmos bajo estudio. En particular, se consideraron las siguientes medidas de desempeño, las cuales serán utilizadas de forma consistente a lo largo del análisis y presentadas mediante representaciones gráficas:

- **Tiempo de procesamiento** (*elapsed time*), que mide el tiempo total de cómputo requerido por los nodos de la red para procesar y tomar decisiones de enrutamiento sobre un paquete a lo largo de su recorrido.
- **Número de saltos** (*number of hops*), que representa la cantidad de nodos intermedios atravesados por un paquete durante su recorrido por la red.
- **Tiempo de arribo** (*arrival time*), que mide el tiempo de simulación transcurrido desde la generación de un paquete hasta su llegada al nodo destino.
- **Tasa de entrega** (*arrival time*), que mide el porcentaje de paquetes que llegan a su destino final a lo largo de la simulación.

Estas métricas permiten caracterizar tanto la eficiencia temporal como el costo de enrutamiento de cada algoritmo, proporcionando una base para la comparación entre distintas configuraciones.

5.2. Algoritmos de enrutamiento

La segunda dimensión de análisis está dada por los algoritmos de enrutamiento evaluados. En este trabajo se comparan CGR y ANTop, tal como se mencionó previamente a lo largo del informe.

El simulador DTNSim cuenta con varias implementaciones del algoritmo CGR, incluyendo versiones que incorporan distintas optimizaciones destinadas a reducir el costo computacional del cálculo de rutas. En una primera instancia se intentó utilizar una implementación de CGR estándar; sin embargo, debido al elevado costo computacional asociado al cálculo de rutas sobre planes de contacto extensos, los tiempos de simulación resultaron prohibitivamente altos para los escenarios considerados en este trabajo.

Por este motivo se optó por utilizar la versión optimizada denominada *CGR Rev17*, la cual introduce diversas mejoras orientadas a reducir la complejidad del algoritmo sin modificar

su comportamiento funcional. Siguiendo lo descrito en [6], se evaluaron dos configuraciones diferentes:

- routeListType: oneBestPath, volumeAware: off, extensionBlock: off, contactPlan: local
- routeListType: perNeighborBestPath, volumeAware: off, extensionBlock: off, contactPlan: local

En la configuración *oneBestPath*, se calcula un único mejor camino para cada destino. La entrada correspondiente en la tabla de ruteo se actualiza cuando finaliza o se agota alguno de los contactos que componen la ruta.

Por otro lado, en la configuración *perNeighborBestPath* se calcula un mejor camino por vecino para cada destino. En este caso, la entrada en la tabla de ruteo se actualiza cuando finaliza o se agota alguno de los contactos pertenecientes a las rutas calculadas.

Los demás parámetros se seleccionaron bajo los siguientes criterios técnicos:

- **volumeAware (off):** Este parámetro desactiva el seguimiento del consumo de volumen en todos los contactos de la ruta.
  - En redes satelitales DTN, es prácticamente imposible sincronizar la visión local de un nodo sobre la ocupación de volumen de toda la red debido a las altas latencias y la conectividad intermitente.
  - Modelar el consumo de volumen de forma completa suele basarse en supuestos simplistas que no reflejan la utilización real en contactos largos o continuos, lo que puede degradar el rendimiento si los datos están desincronizados.
- **extensionBlock (off):** Se refiere a la desactivación de las extensiones de *source routing*.
  - Aunque estas extensiones pueden minimizar el esfuerzo de cálculo en redes continuamente conectadas, en redes altamente interrumpidas como las simuladas, solo ofrecen mejoras marginales y no contribuyen necesariamente a una mejor entrega de datos.
- **contactPlan (local):** Este parámetro asegura que cada satélite tome decisiones de ruteo de forma distribuida basándose en su propia versión local del plan de contactos.
  - Esto es fundamental para que la comparación sea justa frente a ANTop, permitiendo que la conectividad y las decisiones emerjan de la interacción entre los nodos en lugar de un controlador centralizado.

En resumen, estos ajustes permiten reducir la complejidad algorítmica y los tiempos de simulación sin modificar el comportamiento funcional básico del protocolo.

La comparación entre los algoritmos se realiza utilizando las mismas métricas, escenarios y configuraciones de red, con el objetivo de garantizar condiciones experimentales equivalentes.

5.3. Escenarios con y sin fallas

La tercera dimensión corresponde a los escenarios de simulación considerados. Se definieron tanto escenarios ideales como escenarios con fallas, con el fin de analizar la robustez de los algoritmos frente a condiciones adversas. En particular, se simularon los siguientes casos:

- **Escenario sin fallas**, que representa un entorno ideal de operación en el cual todos los nodos de la red se mantienen disponibles durante la simulación.
- **Escenarios con fallas**, en los cuales algunos nodos dejan temporalmente de estar disponibles, simulando fallas o interrupciones operativas. En este trabajo se consideraron tasas de falla del **5 %, 10 % y 20 %**.

De esta manera, es posible estudiar cómo se degradan las métricas de desempeño a medida que aumenta la proporción de fallas, y comparar la capacidad de adaptación de cada algoritmo.

5.4. Constelaciones

La cuarta dimensión de análisis está asociada a la topología de la red satelital. Se utilizaron constelaciones del tipo **Walker**, variando progresivamente el tamaño de la red mediante el número total de satélites, lo cual permite evaluar el comportamiento y la escalabilidad de los algoritmos de enrutamiento a medida que aumenta la dimensión de la red.

Las constelaciones se describen mediante la notación **Walker** $i : t/p/f$, donde:

- i es la inclinación orbital, fijada en 53°, un valor ampliamente utilizado en constelaciones de órbita baja ya que permite cubrir la mayor parte de las regiones de la Tierra con mayor densidad poblacional.
- t es el número total de satélites, con valores 120, 180, 240 y 360.
- p es el número de planos orbitales igualmente espaciados, tomando los valores 12 y 20.
- f es el espaciamiento relativo entre satélites en planos adyacentes.

Para las configuraciones con $p = 12$ se utilizó $f = 7$, mientras que para aquellas con $p = 20$ se adoptó $f = 11$. Asimismo, para las constelaciones con $t \geq 240$ satélites, se incrementó el número de planos a 20. Estos valores se seleccionaron de manera de distribuir los satélites de forma uniforme entre los distintos planos orbitales y evitar alineamientos repetitivos entre satélites de planos adyacentes.

Combinando las distintas dimensiones de análisis —métricas de desempeño, algoritmos de enrutamiento, escenarios de fallas y configuraciones de constelación— se definió un conjunto total de **16 simulaciones**. En todos los casos se utilizó una carga de tráfico constante de **100 paquetes** por nodo generados a lo largo de la simulación.

5.5. Resultados

Los gráficos presentados en las siguientes secciones utilizan escalas logarítmicas, ya que las métricas analizadas presentan diferencias de varios órdenes de magnitud entre los algoritmos evaluados, lo que dificulta su visualización en una misma escala lineal.

En todos los casos, cada algoritmo fue evaluado bajo cuatro configuraciones de fallas en la red: sin fallas, con un 5 %, un 10 % y un 20 % de fallas. Para cada escenario se comparan las métricas obtenidas por ambas configuraciones de CGR y por ANTop.

Cabe destacar que no se reportan resultados correspondientes a la configuración de CGR *perNeighborBestPath* en los escenarios con 360 satélites, debido a que el tiempo de ejecución requerido por estas simulaciones resulta prohibitivamente alto bajo los recursos de cómputo disponibles.

Por otro lado, todas las simulaciones de ANTop fueron realizadas utilizando celdas H3 de resolución 0. Esta decisión se tomó debido a que los escenarios considerados no poseen una cantidad suficiente de satélites para garantizar una distribución adecuada sobre las más de 800 celdas existentes en las demás resoluciones, lo que podría generar una fragmentación excesiva del espacio y afectar la conectividad entre nodos.

5.5.1. Arrival Time

En la Figura 13 se presenta la comparativa del tiempo de llegada (*arrival time*) en milisegundos (*ms*) para los algoritmos evaluados.

En todos los escenarios analizados, ANTop presenta las menores medianas de tiempo de llegada, seguido por CGR *per-neighbor*, mientras que CGR *one-best* exhibe consistentemente los valores más elevados. Esta relación se mantiene para todos los tamaños de constelación y niveles de fallas considerados.

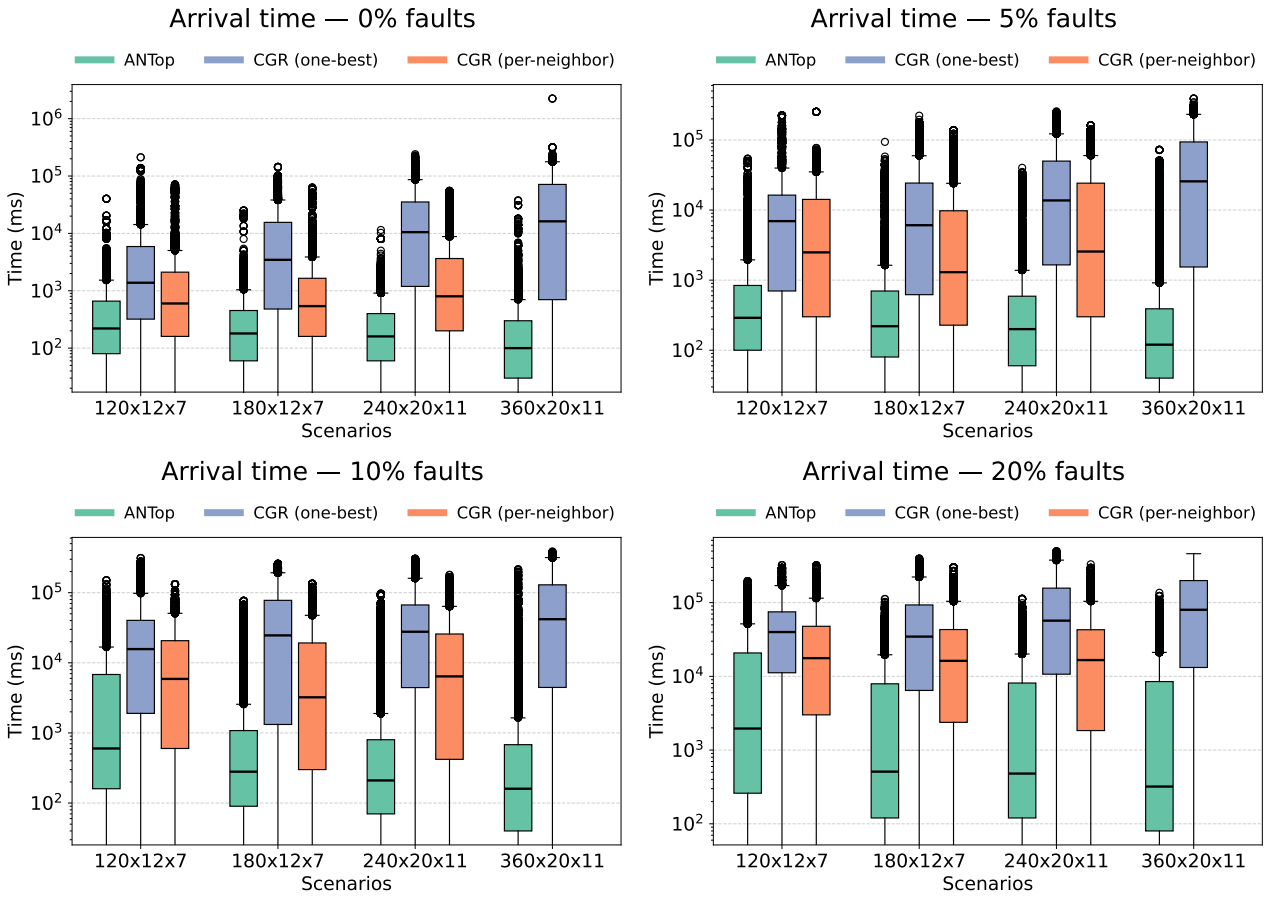

Figura 13: Comparación de Arrival Time para distintos porcentajes de fallas en ANTop y CGR.

En particular, ANTop no solo presenta mejores medianas, sino también una distribución global más favorable: en varios escenarios, el percentil 75 de ANTop se mantiene por debajo del percentil 25 de los algoritmos CGR. Esto indica una mejora consistente en el tiempo de entrega de paquetes.

Un aspecto especialmente relevante es el comportamiento frente al aumento del tamaño de la red. Mientras que el tiempo de llegada de los algoritmos CGR tiende a incrementarse a medida que crece la cantidad de satélites, ANTop muestra la tendencia opuesta, reduciendo sus tiempos de entrega en redes más grandes. Este comportamiento puede atribuirse al paralelismo inherente al proceso de descubrimiento de rutas de ANTop, que permite aprovechar la mayor conectividad disponible en constelaciones más densas.

Por otro lado, al aumentar el porcentaje de fallas se observa un incremento en la dispersión de los tiempos de llegada para ANTop, evidenciado por la expansión de los boxplots. Esto indica una mayor variabilidad en los tiempos de entrega, donde algunos paquetes logran entregarse rápidamente mientras que otros experimentan retrasos mayores. Sin embargo, incluso en el escenario con 20 % de fallas, las medianas de ANTop continúan siendo significativamente

menores que las observadas en los algoritmos CGR.

Finalmente, la diferencia entre las variantes de CGR también resulta consistente con su diseño interno. La configuración *per-neighbor* logra mejorar los tiempos de entrega respecto a *one-best*, lo cual sugiere que considerar múltiples rutas permite encontrar alternativas más eficientes.

5.5.2. Elapsed Time

En la Figura 14 se presenta la comparación del tiempo de procesamiento (*elapsed time*) por paquete, medido en milisegundos (*ms*) bajo una escala logarítmica. Esta métrica permite evaluar el costo computacional asociado al proceso de decisión de ruteo en cada nodo.

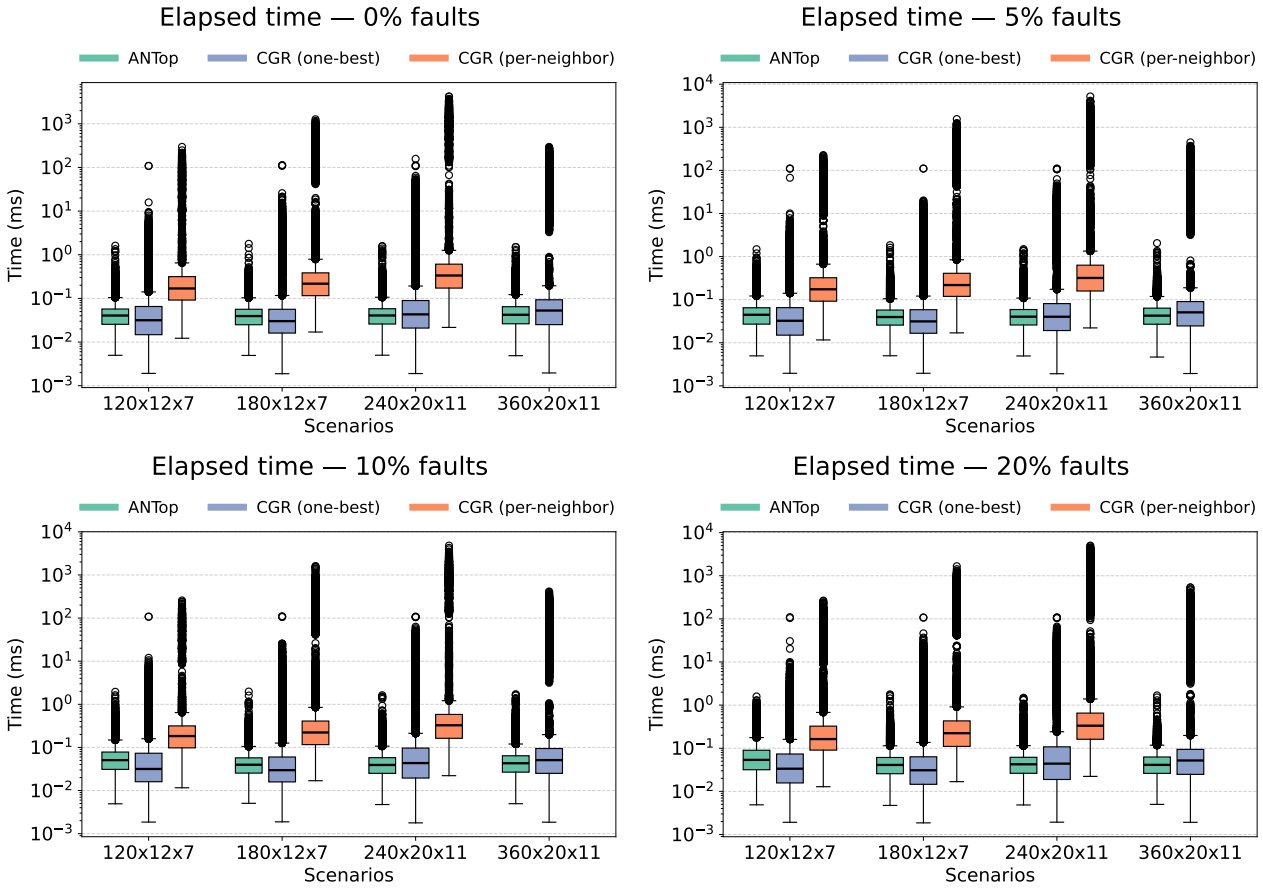

Figura 14: Comparación de Elapsed Time para distintos porcentajes de fallas en ANTop y CGR.

En términos generales, ANTop y CGR *one-best* presentan los menores tiempos de procesamiento, mientras que CGR *per-neighbor* exhibe consistentemente el mayor costo computacional. Esta diferencia se vuelve más pronunciada a medida que aumenta el tamaño de la constelación.

Por su parte, CGR *one-best* presenta en muchos escenarios las menores medianas de tiempo de procesamiento, especialmente en configuraciones de red más simples. Sin embargo, su distribución muestra una dispersión considerablemente mayor que la observada en ANTop, con una gran cantidad de valores atípicos que alcanzan hasta 100 ms. Esto indica que, si bien el costo promedio del algoritmo puede ser bajo, el proceso de cálculo de rutas puede generar episodios de cómputo significativamente más costosos.

Finalmente, ANTop exhibe un tiempo de procesamiento estable y predecible en todos los escenarios considerados. Sus medianas se sitúan consistentemente entre 0,01 y 0,1 ms, con una variabilidad muy reducida. A diferencia de las variantes de CGR, la distribución de sus valores se mantiene compacta, con una presencia limitada de outliers. Asimismo, el costo computacional de ANTop se mantiene prácticamente constante a medida que aumenta el tamaño de la red, llegando incluso a presentar medianas ligeramente inferiores a las de CGR *one-best* en los escenarios con mayor cantidad de satélites.

5.5.3. Number of Hops

En la Figura 15 se presenta la comparación del número de saltos (*number of hops*) recorridos por los paquetes para alcanzar su destino final.

En términos generales, ANTop presenta el menor número de saltos en la mayoría de los escenarios considerados. Sus medianas se mantienen típicamente entre 1 y 10, con una dispersión relativamente reducida y una baja presencia de outliers. Este comportamiento se mantiene incluso a medida que aumenta el tamaño de la constelación o el nivel de fallas en la red, lo que sugiere que el algoritmo logra mantener rutas relativamente cortas y estables.

Por su parte, la variante CGR *per-neighbor* presenta un desempeño intermedio. En términos de la mediana, los resultados son comparables a los observados en ANTop en varios escenarios. Sin embargo, la distribución muestra una mayor dispersión y una presencia más significativa de outliers.

Finalmente, CGR *one-best* exhibe sistemáticamente los valores más elevados de número de saltos. Si bien las medianas no difieren de manera extrema respecto a las otras variantes, este algoritmo presenta una mayor cantidad de valores atípicos y una dispersión considerablemente mayor. En algunos escenarios se observan casos extremos que superan los 10000 saltos, un valor significativamente superior al tamaño de las constelaciones simuladas. Este comportamiento podría estar asociado a decisiones de reencaminamiento sucesivas sobre el plan de contactos, que generan trayectorias particularmente largas antes de que el paquete alcance su destino final.

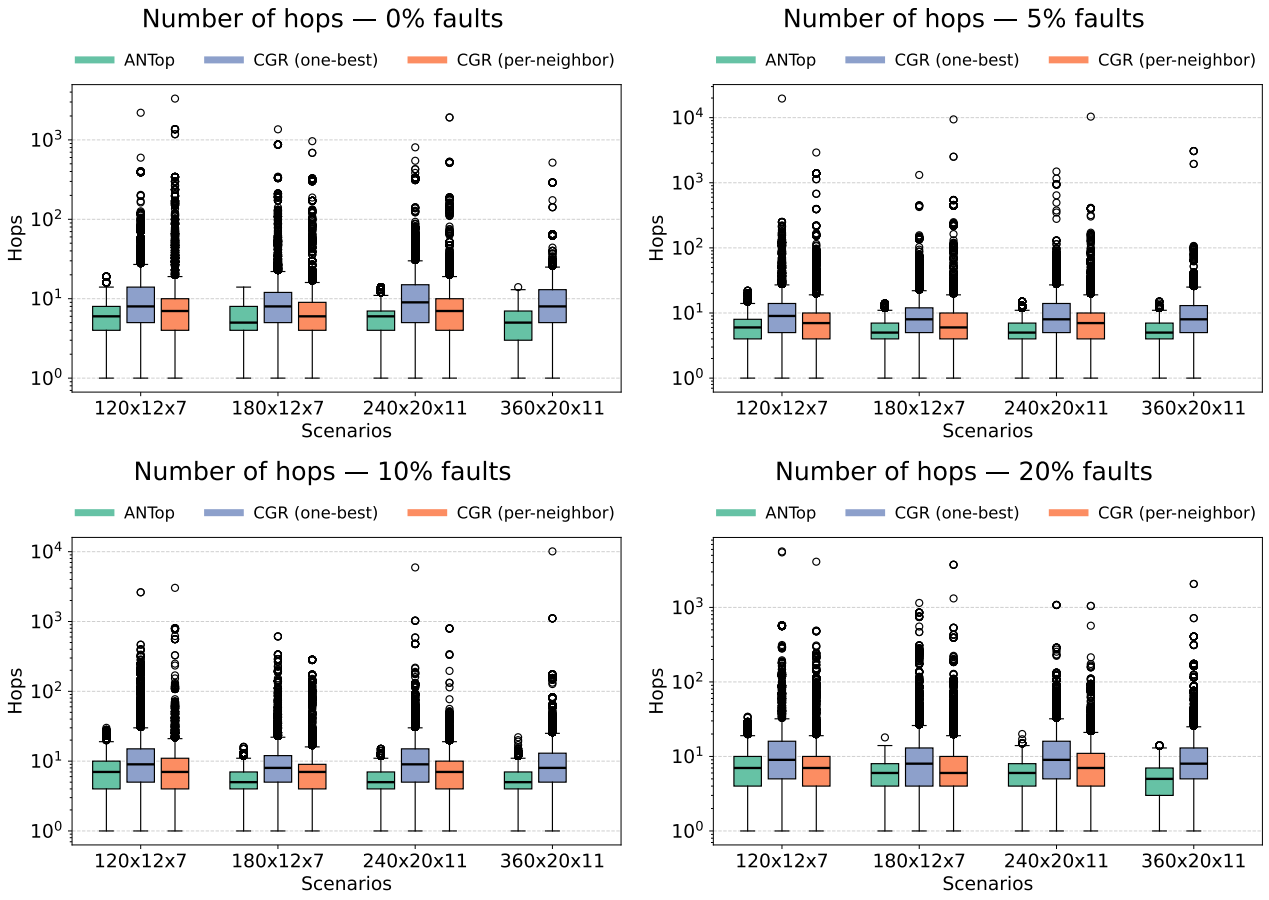

Figura 15: Comparación de Number of Hops para distintos porcentajes de fallas en ANTop y CGR.

5.5.4. Delivery ratio

En la Figura 16 se presenta la comparación de la tasa de entrega de paquetes (*Delivery Ratio*) para los distintos algoritmos evaluados bajo diferentes niveles de fallas.

ANTop mantiene un desempeño constante del 100 % en todos los escenarios y niveles de falla considerados (0 %, 5 %, 10 % y 20 %). Este comportamiento indica que el algoritmo logra encontrar rutas válidas hacia el destino incluso bajo condiciones adversas de la red, manteniendo una alta robustez frente a la degradación de enlaces.

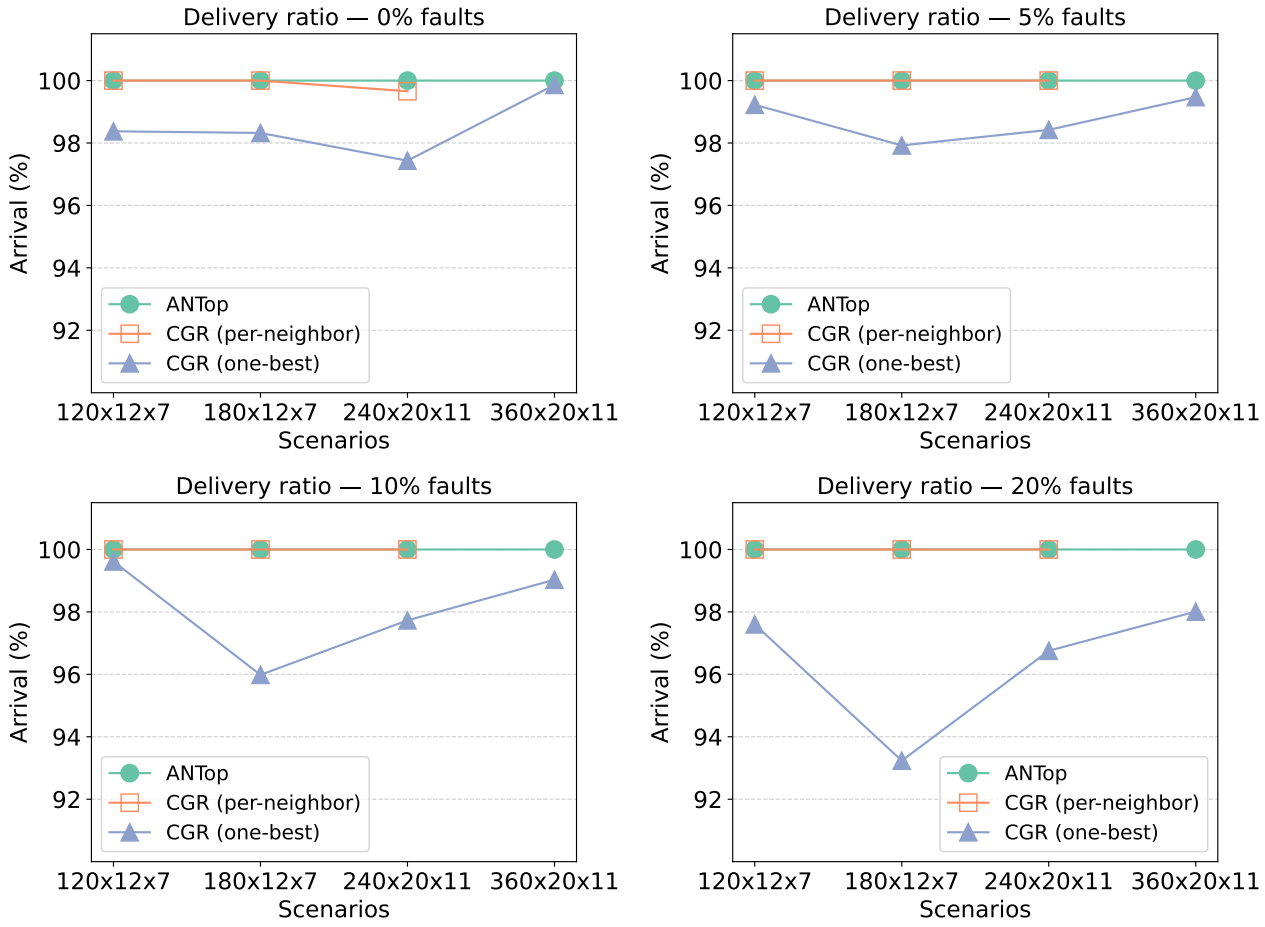

Figura 16: Comparación de Delivery Ratio para distintos porcentajes de fallas en ANTop y CGR.

Por su parte, CGR *per-neighbor* presenta un comportamiento muy similar al de ANTop, manteniendo una tasa de entrega del 100 % en prácticamente todos los escenarios evaluados. Solo se observa una leve fluctuación en el escenario de 240 satélites sin fallas, aunque la diferencia respecto al valor ideal es mínima.

En contraste, CGR *one-best* es el algoritmo que presenta la mayor variabilidad en esta métrica. A diferencia de las otras variantes, no alcanza el 100 % de entrega en ninguno de los escenarios evaluados. Su desempeño tampoco sigue una tendencia estrictamente monótona con respecto al tamaño de la constelación. En particular, los valores más bajos de tasa de entrega se observan en los escenarios de 180 satélites para configuraciones con 5 %, 10 % y 20 % de fallas, mientras que los resultados mejoran nuevamente en las simulaciones con 360 satélites. Un comportamiento similar se observa en el escenario sin fallas, donde el punto más bajo se presenta en la configuración de 240 satélites. En los casos más desfavorables, la tasa de entrega desciende hasta valores cercanos al 93 %.

6. Mediciones de error de distancias

La Figura 17 muestra mediciones del error relativo de las distancias entre los nodos del grafo de direcciones ANTop respecto a las distancias reales en H3.

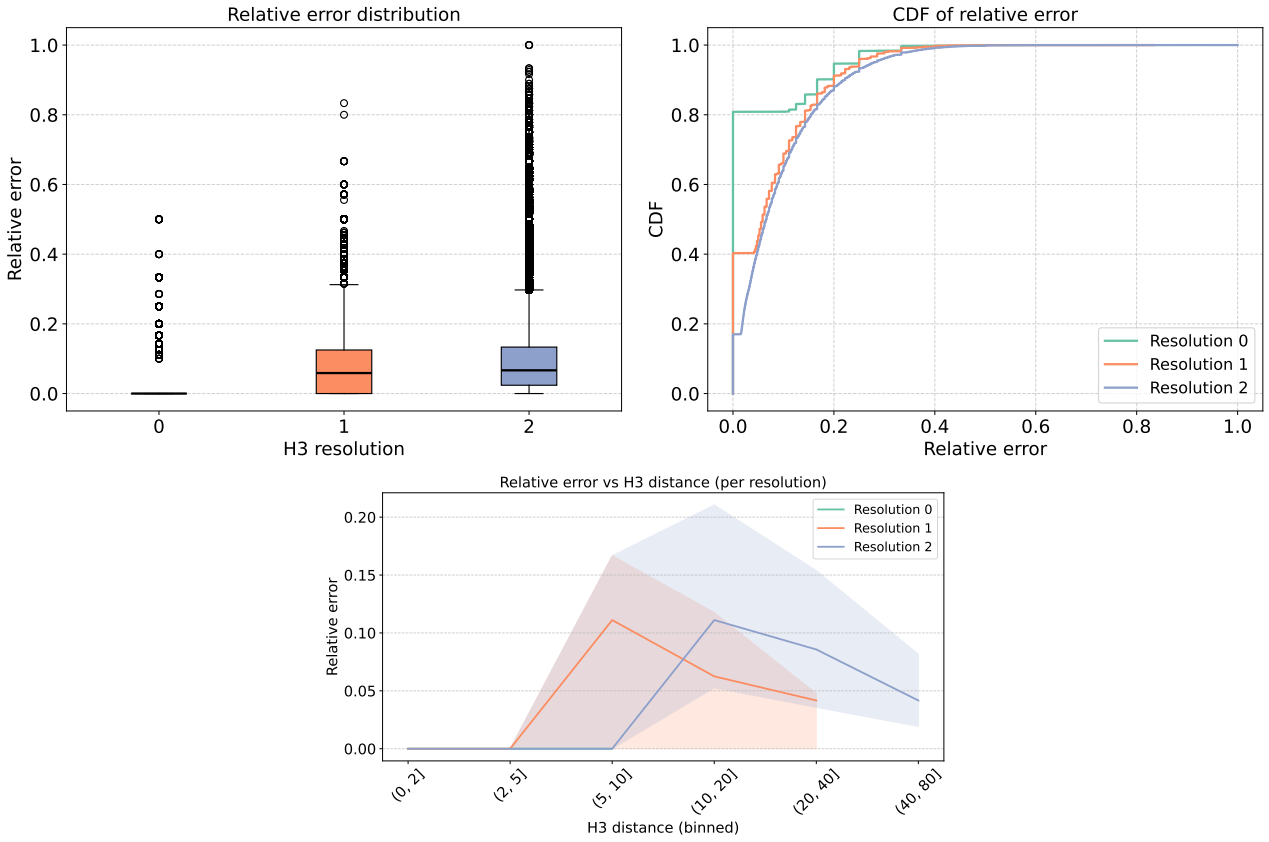

Figura 17: Mediciones de error relativo entre distancias por dimensión.

El análisis del error relativo muestra que al aumentar la resolución de la grilla, se pierde exactitud en las distancias comparadas con H3. Cuando el detalle es mayor, la probabilidad de tener un error de cero baja drásticamente, lo que indica que dividir el espacio de forma más fina introduce más deformaciones en las distancias.

Sin embargo, una vez que las diferentes resoluciones empiezan a mostrar fallos, las curvas de error tienden a parecerse y a estabilizarse. En general, el error se mantiene en niveles aceptables y no crece de forma exagerada. Por esto, podemos decir que estas diferencias no son graves para el uso práctico, ya que el sistema sigue representando bien la forma general de los datos a pesar de los pequeños errores que aparecen localmente.

7. Conclusiones y trabajos futuros

7.1. Conclusiones

El presente trabajo tuvo como objetivo evaluar el desempeño del protocolo ANTop en redes satelitales de tipo DTN y compararlo con dos variantes de CGR: *one-best route* y *per-neighbor route*. Para ello se realizaron simulaciones sobre constelaciones satelitales de distinta escala y bajo diferentes niveles de fallas, analizando métricas clave como *Arrival Time*, *Elapsed Time*, *Number of Hops* y *Delivery Ratio*.

Los resultados obtenidos muestran que ANTop presenta el mejor desempeño global entre los algoritmos evaluados. En términos de latencia, logra mantener consistentemente los menores tiempos de llegada de paquetes, incluso a medida que aumenta el tamaño de la constelación. Asimismo, el número de saltos requerido para alcanzar el destino se mantiene bajo y con una variabilidad reducida, lo que indica que el algoritmo es capaz de encontrar rutas eficientes de forma consistente.

Desde el punto de vista computacional, ANTop demuestra un comportamiento estable y predecible. A diferencia de las variantes de CGR, cuyos tiempos de procesamiento tienden a aumentar o a presentar mayor dispersión a medida que crece la red, el costo computacional de ANTop se mantiene prácticamente constante en todos los escenarios evaluados. Esta propiedad resulta particularmente relevante en sistemas satelitales, donde los recursos de procesamiento disponibles a bordo suelen ser limitados.

En términos de fiabilidad, ANTop mantiene una tasa de entrega del 100 % en todos los escenarios considerados, incluso bajo niveles elevados de fallas en los enlaces. Por su parte, CGR *per-neighbor* logra resultados comparables en la mayoría de los casos, aunque con un costo computacional significativamente mayor. En contraste, la variante CGR *one-best* presenta un comportamiento menos robusto, con reducciones apreciables en la tasa de entrega y una mayor variabilidad tanto en latencia como en costo de procesamiento.

En conjunto, estos resultados sugieren que el enfoque distribuido de ANTop permite alcanzar un equilibrio favorable entre eficiencia, robustez y estabilidad computacional en redes satelitales con niveles relativamente altos de conectividad. En los escenarios evaluados, este enfoque logra superar de forma consistente a las variantes de CGR consideradas, especialmente en términos de latencia y previsibilidad del comportamiento del sistema.

7.2. Trabajos futuros

Si bien los resultados obtenidos en este trabajo permiten caracterizar el comportamiento de ANTop y compararlo con distintas variantes de CGR, existen diversas líneas de investigación que podrían profundizar y extender los hallazgos presentados.

En primer lugar, los escenarios evaluados en este trabajo corresponden a constelaciones satelitales con niveles relativamente altos de conectividad, donde múltiples rutas potenciales

suelen existir entre origen y destino. Una línea natural de investigación futura consiste en analizar el desempeño de ANTop en entornos de mayor discontinuidad, donde los nodos deban esperar períodos prolongados para la aparición de nuevos enlaces. En este contexto, podría explorarse el desarrollo de mecanismos adicionales que permitan aumentar la resiliencia del algoritmo frente a largas interrupciones de conectividad, acercando su comportamiento al de enfoques basados en planes de contactos globales.

Otra posible extensión consiste en analizar alternativas al sistema de discretización espacial utilizado en este trabajo. La implementación actual se basa en la biblioteca H3, que divide la superficie terrestre en celdas hexagonales jerárquicas. Si bien esta representación ofrece ventajas importantes en términos de uniformidad de área y regularidad en la distancia entre vecinos, presenta algunas limitaciones prácticas. En particular, la resolución más baja del sistema contiene 122 celdas, lo que puede resultar excesivo para constelaciones satelitales relativamente pequeñas. Por ejemplo, el sistema GPS opera actualmente con aproximadamente 31 satélites, por lo que una discretización más flexible podría permitir representar de forma más natural ciertos escenarios.

Adicionalmente, el sistema H3 no proporciona un sistema de coordenadas global único, lo que introduce complejidades adicionales al momento de representar la topología completa de la red. En la implementación desarrollada en este trabajo fue necesario utilizar múltiples sistemas locales y estructuras auxiliares para construir una representación global coherente. En este sentido, resultaría interesante explorar otras bibliotecas o esquemas de particionamiento espacial que permitan simplificar esta representación o adaptarla mejor a las necesidades específicas de redes satelitales.

Asimismo, una línea de trabajo relevante consiste en profundizar el análisis del comportamiento de las distintas variantes de CGR en los escenarios evaluados. En particular, sería valioso investigar con mayor detalle las causas de la degradación observada en métricas como *Arrival Time* y *Delivery Ratio*, con el objetivo de identificar posibles limitaciones estructurales de estos algoritmos y proponer mejoras que permitan adaptar su funcionamiento a constelaciones satelitales modernas de gran escala.

Referencias

[1] Jose I. Alvarez-Hamelin, Aline C. Viana, and Marcelo D. de Amorim. Dht-based functionalities using hypercubes. Universidad de Buenos Aires; IRISA/INRIA-Rennes; CNRS/LIP6 – Université Pierre et Marie Curie (Paris VI).

[2] Scott Burleigh. Contact Graph Routing. Internet-Draft draft-burleigh-dtnrg-cgr-01, Internet Engineering Task Force, July 2010. Work in Progress.

[3] Xiaoli Cao, Yitao Li, Xingzhong Xiong, and Jun Wang. Dynamic routings in satellite networks: An overview. *Sensors (Basel)*, 22(12):4552, June 2022.

[4] Uber Engineering. GitHub - uber h3: Hexagonal hierarchical geospatial indexing system — github.com. https://github.com/uber/h3.

[5] Uber Engineering. H3: Hexagonal hierarchical spatial index. https://www.uber.com/en-SE/blog/h3/, n.d. Accessed: 2026-01-30.

[6] Juan A. Fraire, Pablo G. Madoery, Amir Charif, and Jorge M. Finochietto. On route table computation strategies in delay-tolerant satellite networks. *Ad Hoc Networks*, 80:31–40, 2018.

[7] H3Geo. H3 documentation. https://h3geo.org/docs, n.d. Accessed: 2026-01-30.

[8] Mark Handley. Delay is not an option: Low latency routing in space. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, HotNets '18, page 85–91, New York, NY, USA, 2018. Association for Computing Machinery.

[9] Alejandro Marcu. Desarrollo y simulación de un protocolo para redes ad-hoc. Tesis de grado de ingeniería electrónica, Buenos Aires, Argentina, Julio 2007.

[10] OMNeT++ Community. OS3 - the open source satellite simulator. https://omnetpp.org/download-items/OS3.html, 2015.

[11] Uber Technologies, Inc. Overview of the h3 geospatial indexing system, 2026. Último acceso: 17 de febrero de 2026.

[12] Aiden Valentine and George Parisis. Developing and experimenting with LEO satellite constellations in OMNeT++. In *Proceedings of the 8th OMNeT++ Community Summit Conference*, Hamburg, Germany, 2021.

[13] Wikipedia contributors. Simplified perturbations models — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Simplified_perturbations_models&oldid=1336329609, 2026. [Online; accessed 8-March-2026].

[14] Qi Xiaogang, Ma Jiulong, Wu Dan, Liu Lifang, and Hu Shaolin. A survey of routing techniques for satellite networks. *Journal of Communications and Information Networks*, 1(4):66–85, 2016.