EVALUACION COMPARATIVA DE LOS PROTOCOLOS DE ENRUTAMIENTO EN WSN DE TIPO GEOGRAFICO 2-f-MFR, 2-f-GEDIR, GDSTR, GEAR Y GAF

CAMILO ANDRES RUIZ DIAZ

ADRIANA GUSSONI VENEGAS

Director:

Ing. Luis Carlos Trujillo Arboleda

Pontificia Universidad Javeriana

Ingeniería Electrónica

Bogotá

2010

ARTÍCULO 23 DE LA RESOLUCIÓN No. 13 DE JUNIO DE 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado.

Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

Agradecimientos:

Agradezco a mi papa y a mi mama, por ser los seres más incondicionales del mundo, por apoyarme siempre que lo necesite, por consentirme todos mis caprichos, por forjar esta carrera conmigo y entender que aquí lo que vale es ganar la guerra y no la batalla. Les agradezco porque la persona que entrega hoy este documento es producto de todo el esfuerzo que han hecho durante estos 25 años, definitivamente sin unos papas como ustedes esto no habría sido posible. Este merito es todo suyo!!

Agradezco a mis compañeros en Tigo, a Sergio Monsalve, el mejor jefe. Gracias a todos por el tiempo invirtieron para que hoy yo pueda entregar mi proyecto de grado y realizarme como profesional. Sergio, definitivamente sin ti esto habría sido un infierno!

Gracias a mi compañero de tesis. Cami: por tu paciencia y por entender que estos últimos seis meses fueron los seis meses más pesados de mi vida.

Ahora creo que el desgaste emocional, físico y mental valió la pena!

"La vida es un espejo, si le sonríes te sonríe"

Adriana Gussoni Venegas

Doy gracias a mis padres, los cuales siempre me han apoyado, colaborado y guiado durante toda mi vida y especialmente estos últimos años en la universidad, dándome coraje y ánimos para lograr todos mis objetivos.
Gracias a mis hermanos que me han apoyado y han sido una compañía para mí.
Finalmente, a mi compañera Adriana Gussoni que fue mas de una vez una fuente de fuerza para continuar, por su apoyo y carisma, por su colaboración a lo largo de todo el proyecto, por su dedicación y los sacrificios que debió realizar para poder finalizar este proyecto y por haber llegado a ser una amiga más que una compañera de trabajo.

Camilo Andres Ruiz Diaz

INDICE

1. INTRODUCCIÓN 9

2. MARCO TEÓRICO 10

2.1. Redes WSN [15] 10

2.2. Protocolos de enrutamiento en WSN [15] 11

2.3. Selección de protocolos de enrutamiento geográfico 13

2.4. Protocolos: 2-f-GEDIR y 2-f-MFR [10] 13

2.4.1. Características de 2-f- GEDIR y 2-f-MFR 14

2.4.2. Funcionamiento General 14

2.4.3. Funcionamiento detallado 15

2.4.4. Diagrama de flujo 18

2.5. Protocolo GEAR [12] 18

2.5.1. Características principales de GEAR 18

2.5.2. Funcionamiento general 18

2.5.3. Funcionamiento detallado 19

2.5.4. Diagrama de flujo 22

2.6. Protocolo GAF [9] 22

2.6.1. Características de GAF 22

2.6.2. Funcionamiento General 23

2.6.3. Funcionamiento detallado 23

2.6.4. Diagrama de flujo 27

2.7. Protocolo GDSTR [11] 27

2.7.1. Características de GDSTR 27

2.7.2. Funcionamiento general 27

2.7.3. Funcionamiento detallado 28

2.7.4. Diagrama de flujo 31

3. ESPECIFICACIONES 31
3.1. Herramienta de Simulación: Opnet Modeler [20, 21] 31
3.1.1. Modelo de Red 32
3.1.2. Modelo de Nodos 32
3.1.3. Modelo de Procesos 33
3.2. Consideraciones y especificaciones de los protocolos a analizar 34
3.3. Creación de escenarios 35
3.4. Parámetros a analizar 36
3.5. Proceso de Validación 37
4. DESARROLLOS GENERALES 40
4.1. Generales 40
4.1.1. Uso del "rx_group" 40
4.1.2. Uso de nodo fuente y nodo destino 40
4.1.3. Modelo de consumo de energía 41
4.1.4. Desarrollo de protocolo de reconocimiento de vecinos y del sumidero 41
4.1.4.1. Paquete de reconocimiento 41
4.1.4.2. Funcionamiento 42
4.2. Específicos 43
4.2.1. Desarrollo de los protocolos 2-f-GEDIR y 2-f-MFR 43
4.2.1.1. Conocimiento de vecinos a dos saltos 44
4.2.1.2. Desarrollo del protocolo y paquete de información 45
4.2.1.3. Nodo cóncavo y *flooding* (inundación) restringido 47
4.2.2. Desarrollo del protocolo GEAR 48
4.2.2.1. Creación de listas de costo estimado 48
4.2.2.2. Desarrollo del protocolo y paquete de información 49
4.2.2.2.1. Actualización del costo estimado y paquete de actualización 50

4.2.2.2.2. Caso nodo cóncavo 51
4.2.3. Desarrollo del protocolo GAF 51
4.2.3.1. Identificación de grillas virtuales 51
4.2.3.2. Desarrollo del Protocolo 52
4.2.3.2.1. Estado “Discovery” 52
4.2.3.2.2. Estado “Active” 53
4.2.3.2.3. Estado “Sleep” 54
4.2.3.2.4. Acople de GAF al protocolo de enrutamiento y Paquete de información 54
4.2.4. Desarrollo de GDSTR 55
4.2.4.1. Módulo *“Spanning_tree”* 55
4.2.4.1.1. Proceso de selección del nodo raíz 55
4.2.4.1.2. Definición del *convex hull* propio 57
4.2.4.1.3. Proceso de mantenimiento del árbol 58
4.2.4.2. Módulo “*protocolo_enrutamiento”* 58
5. Análisis de Resultados 60
5.1. Escenarios determinísticos 60
5.1.1. 2-f-GEDIR 60
5.1.2. 2-f-MFR 62
5.1.3. GEAR 63
5.1.4. GDSTR 64
5.1.5. GAF 65
5.2. ESCENARIOS ALEATORIOS 67
5.2.1. Network Use 67
5.2.2. IPDV 70
5.2.3. End-to-end delay 71
5.2.4. Hop count 73

5.2.5. Nodo cóncavo 75
5.2.6. Energía restante 77
5.2.7. Tasa de entrega 80
6. CONCLUSIONES 82
7. Bibliografía 84

INDICE DE FIGURAS

Figura 1 Aplicaciones de las WSNs 9
Figura 2 Clasificación protocolos de enrutamiento para WSN 12
Figura 3 Vecino a *2-hop* 14
Figura 4 Vecinos de S y D 16
Figura 5 Protocolo GEDIR (S-A-B-C-D) y 2-f-GEDIR (S-B-C-D) 16
Figura 6 MFR: Progreso de los nodos vecinos a S 17
Figura 7 Ejemplo GEAR 21
Figura 8 Costos aprendidos y estimados. 21
Figura 9 Ejemplo de grilla virtual en GAF 24
Figura 10 Maquina de estados de GAF 25
Figura 11 Nodos con sus *convex hulls* 28
Figura 12 Ejemplo de un buen (b) y un mal (a) árbol 29
Figura 13 Modelo de Red 32
Figura 14 Modelo de nodo 33
Figura 15 Modelo de proceso 33
Figura 16 Cambio de topología del protocolo 2-f-GEDIR 38
Figura 17 Paquete de reconocimiento: *"pkt_recon"* 42
Figura 18 Modelo de proceso del módulo *"proto_recon"* 43
Figura 19 Paquete de reconocimiento vecino 2-hop *"vec2h_packet"* 44
Figura 20 Modelo de proceso de *"proceso_vec_2h"* 45
Figura 21 Modelo de proceso 2-f-GEDIR y 2-f-MFR 45
Figura 22 Paquete protocolos 2-f-GEDIR y 2-f-MFR *"pkt_GEDIR"* 46
Figura 23 Modelo de proceso GEAR 48

Figura 24 Paquete de información GEAR *"pkt_GEAR"* 49
Figura 25 Paquete de actualización *"pkt_update"* 50
Figura 26 Modelo de proceso de GAF 52
Figura 27 Paquete de descubrimiento *"pkt_discovery"* 53
Figura 28 Modelo de proceso *"spanning_tree"* 55
Figura 29 Paquete *"pkt_root_selec"* 56
Figura 30 Paquete *"pkt_hull_trees"* 57
Figura 31 Definición *convex hull* propio 57
Figura 32 Convergencia de los *convex hulls* 57
Figura 33 Paquete *“pkt_keepalive”* 58
Figura 34 Paquete *"pkt_proto_gdstp"* 59
Figura 35 Modelo de proceso de "*protocolo_enrutamiento"* 59
Figura 36 Densidad correspondiente a 110 nodos 68
Figura 37 Densidad correspondiente a 200 nodos 68
Figura 38 Densidad correspondiente a 280 nodos 69
Figura 39 Densidad correspondiente a 110 nodos 70
Figura 40 Densidad correspondiente a 200 nodos 70
Figura 41 Densidad correspondiente a 280 nodos 71
Figura 42 Densidad correspondiente a 110 nodos 72
Figura 43 Densidad correspondiente a 200 nodos 73
Figura 44 Densidad correspondiente a 280 nodos 73
Figura 45 Densidad correspondiente a 110 nodos 74
Figura 46 densidad correspondiente a 200 nodos 74
Figura 47 Densidad correspondiente a 280 nodos 75
Figura 48 Densidad correspondiente a 110 nodos 76
Figura 49 Densidad correspondiente a 200 nodos 76
Figura 50 Densidad correspondiente a 280 nodos 77
Figura 51 Densidad correspondiente a 110 nodos 78
Figura 52 densidad correspondiente a 200 nodos 78
Figura 53 Densidad correspondiente a 280 nodos 79
Figura 54 Densidad correspondiente a 110 nodos 80
Figura 55 Densidad correspondiente a 200 nodos 81

Figura 56 Densidad correspondiente a 280 nodos..... 81

1. INTRODUCCIÓN

Gracias a la reducción de costos en elementos electrónicos de comunicación inalámbrica, es posible construir sensores con múltiples funciones y propósitos que operan con poca energía, de tamaño pequeño, y con una capacidad de comunicación a corta distancia [4]. Estos sensores han permitido el surgimiento de nuevos tipos de redes, conocidas como redes de sensores inalámbricas WSN (Wireless Sensor Networks, por sus siglas en inglés), las cuales constan de cientos o miles de nodos (sensores), que se encuentran coordinados para realizar una acción determinada.

Las redes de sensores inalámbricas miden características del ambiente que las rodea y transforman estas medidas en señales que pueden ser procesadas y transmitidas para obtener información específica de una región geográfica [9]. Las áreas más importantes en las que se desarrollan aplicaciones para las WSN son: medio ambientales, militares, salud, hogar e industrial [1,3]. Por ejemplo una aplicación medio ambiental consiste en desplegar los sensores en superficies volcánicas, Figura 1, para estudiar las emanaciones de gases tóxicos y predecir alteraciones importantes que indiquen un cambio en la actividad del volcán; para la milicia también son muy importantes debido a que pueden ser usadas para detectar movimientos enemigos en campos de batalla.

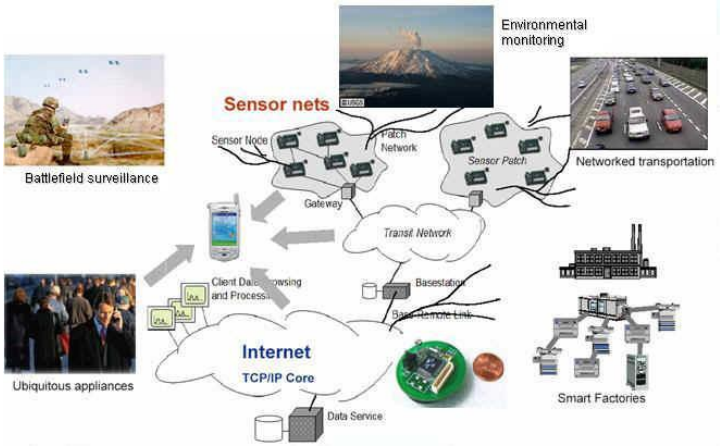


Figura 1 Aplicaciones de las WSNs[1]

Las WSN no funcionan de igual manera que las redes tradicionales, ya que los nodos se encuentran equipados con una fuente de energía limitada. En algunos escenarios donde el acceso es limitado o nulo, cambiar estas fuentes es prácticamente imposible, por lo tanto el desempeño de la red muestra una gran dependencia con la conservación de la energía de cada sensor [1, 2, 3]. De igual manera los protocolos de enrutamiento existentes en las redes tradicionales como las redes móviles *ad-hoc* o redes celulares, no son

[1] Imagen tomada de [18].

recomendados para el uso en WSN; esto debido a que las redes de sensores inalámbricas poseen un gran número de nodos, y lograr crear un esquema de enrutamiento global, en el que todos los nodos de la red almacenan una gran cantidad de rutas para llegar a diferentes destinos, implicaría un mayor procesamiento y no cumpliría con las restricciones de energía. En consecuencia una de las tareas más desafiantes en WSN es el enrutamiento.

La tarea de encontrar y mantener rutas en WSNs no es trivial, ya que las restricciones de energía y los cambios súbitos en el estado del nodo (ejemplo, falla del nodo), causan frecuentes e impredecibles cambios de topología. Para minimizar el consumo de energía y efectuar un enrutamiento eficiente, existen diferentes técnicas propuestas para WSNs las cuales emplean estrategias de básicas de enrutamiento y estrategias especiales para WSNs [3]. Los protocolos de enrutamiento geográfico proponen una estrategia especial para las WSN, en la que cada nodo conoce su posición y la de los nodos que se encuentran dentro de su área de cobertura, es decir aquellos con los que puede establecer una comunicación directa.

Este trabajo de grado busca realizar una evaluación comparativa de cinco protocolos representativos de enrutamiento geográfico, con la que se quiere estudiar el comportamiento de estos frente a diferentes condiciones de operación en la WSN, y dejar una base para trabajos futuros que busquen realizar una implementación en nodos reales, analizando si la estrategia de enrutamiento que es más eficiente en la teoría, también lo es en un ambiente real.

El siguiente documento se organiza de la siguiente manera: en el Marco Teórico, sección 2, se explica qué es una red WSN y cuáles son las características generales de los protocolos de enrutamiento geográfico. También se explica la base teórica de los protocolos seleccionados. En las Especificaciones, sección 3, se explica la herramienta de simulación utilizada y el modelo de trabajo que esta propone; así mismo se especifican los diferentes escenarios que son usados para probar los protocolos y las consideraciones que se tienen en cuenta al momento de efectuar la simulación y realizar la evaluación entre los protocolos. En el Desarrollo, sección 4, se explica cómo fueron implementados los protocolos en el simulador. En la sección 5, se muestran los resultados obtenidos. En la sección 6, se concluye que protocolo es más eficiente en cada parámetro analizado.

2. MARCO TEÓRICO

2.1. Redes WSN [15]

En los últimos años las redes de sensores inalámbricas han adquirido gran importancia a nivel global, particularmente gracias al crecimiento de los sistemas micro-electro-mecánicos (*Micro-Electro-Mechanical Systems, MEMS*). Estos sensores son pequeños, con procesamiento y recursos computacionales limitados, y son económicos comparados con los sensores tradicionales; pueden medir y

recolectar información del ambiente, y basados en un proceso de decisiones locales transmiten la información a un usuario.

Debido a que los nodos (sensores) tienen memoria limitada y típicamente se despliegan en locaciones donde el acceso es limitado, son dotados con tecnologías de comunicación inalámbrica, para transmitir la información a una estación base (ej. Un computador personal, un dispositivo de mano, o a un punto de acceso a una infraestructura fija). En la actualidad se ha implementado en los nodos sistemas GPS (*Global Positioning System*), con los cuales cada nodo conoce su ubicación

La batería es la principal fuente de poder en un sensor, por lo que operan con batería limitada. El uso de la energía es una de las principales consideraciones que se deben tener en cuenta en una WSN: cuando un sensor ha agotado su energía, muere y al desconectarse de la red puede impactar significativamente el desempeño de la aplicación que está siendo ejecutada. La vida de la red depende del número de nodos activos y de su conectividad, así que la energía debe ser usada eficientemente a fin de maximizar el tiempo de vida.

Las WSNs poseen, entre otras, las siguientes restricciones: cantidad limitada de energía, rangos de comunicación cortos, poco ancho de banda, almacenamiento y procesamiento limitados en cada nodo. Si la conectividad de la red es baja, pueden llegar a presentarse *loops* (bucles); se habla de un bucle cuando un mismo paquete al no encontrar una ruta al destino, retorna sobre los nodos de una ruta por la cual ya había transitado. Debido a esto, las investigaciones en WSN buscan reconocer estas limitaciones e introducir nuevos conceptos de diseño, creando o mejorando protocolos existentes, construyendo nuevas aplicaciones y desarrollando nuevos algoritmos.

2.2. Protocolos de enrutamiento en WSN [15]

En una red existen múltiples rutas para enviar información entre dos nodos que no se encuentran directamente conectados. El proceso de encontrar estas rutas es llamado enrutamiento, y consiste en que cada nodo involucrado en el proceso de reenvió de paquetes, elige a un nodo directamente conectado para enviar la información a un nodo destino. El nodo elige una ruta al destino haciendo uso de un protocolo de enrutamiento, el cual utiliza una métrica específica para elegir el siguiente nodo vecino que a su vez se continúe con el enrutamiento del paquete; este proceso se ejecuta en cada nodo que se encuentra en la ruta hacia el destino.

El desarrollo de protocolos confiables y eficientes en energía, es importante para soportar diferentes aplicaciones de WSN; dependiendo de la aplicación una red puede tener cientos o miles de nodos, y cada nodo hace uso de los protocolos para comunicarse con los otros nodos y con el nodo destino, o también llamado nodo *sink* (sumidero). Debido a esto, el conjunto de protocolos que se trabaja en las diferentes capas del modelo OSI (transporte, red, enlace de datos, etc.) debe ser eficiente en términos de

comunicación y energía. Este proyecto se enfoca en trabajar con los protocolos de la capa de red, es decir capa 3.

La capa de red maneja el enrutamiento de la información a través de la red, desde el nodo fuente (*source node*) hasta el nodo destino. Los protocolos de enrutamiento de WSN difieren de los protocolos de enrutamiento tradicionales de diversas maneras: los nodos (sensores) no poseen direcciones de protocolo de internet (*Internet Protocol, IP*), poseen una batería limitada y tienen una capacidad de procesamiento pequeaña, por lo que los protocolos basados en IP no deben ser usados. El protocolo de enrutamiento en WSN debe ser escalable, debe poder manejar fácilmente la comunicación entre varios nodos y propagar la información a la estación base, o nodo destino; el protocolo debe conocer las restricciones de la red, como la energía limitada de los nodos, el ancho de banda que poseen, su memoria y las capacidades computacionales; haciendo uso de esta información el protocolo puede alargar el tiempo de vida de la red. En la literatura existen diferentes taxonomías de protocolos de enrutamiento para WSNs, luego de estudiar diferentes fuentes bibliográficas, se elabora la clasificación indicada en la Figura 2.

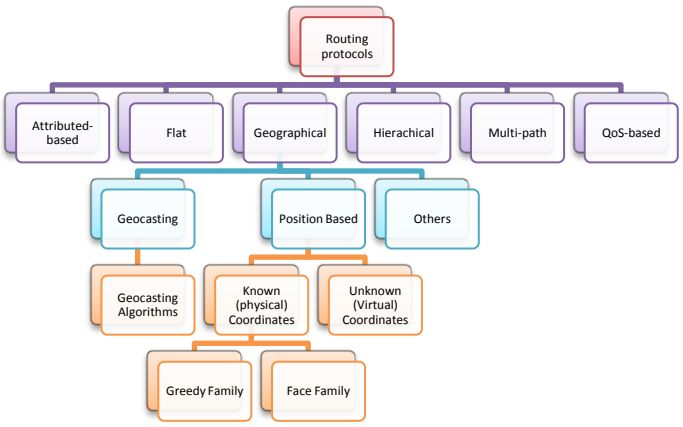


Figura 2 Clasificación protocolos de enrutamiento para WSN[2]

Un grupo de protocolos que tiene en cuenta las restricciones mencionadas anteriormente, es el de protocolos de enrutamiento geográfico. Estos protocolos generan gran interés debido a que usan técnicas que permiten hacer un mejor uso de los recursos de la red a diferencia de otros protocolos de WSNs, los cuales para conocer rutas al destino inundaban la red de paquetes.

Un gran número de protocolos de enrutamiento geográfico utiliza el mecanismo de reenvió *Greedy* para enviar un paquete del nodo fuente al nodo destino; esta estrategia de enrutamiento consiste en reenviar un paquete al vecino que se encuentre más cercano en distancia euclidiana al destino. *Greedy* asume que la

[2] El esquema de clasificación de los protocolos de enrutamiento de las WSN es de realización propia, utilizando las referencias bibliográficas [2] y [5].

red es suficientemente densa, que los nodos conocen su posición propia y la posición de sus vecinos que se encuentran dentro de su rango de cobertura, así como el número de identificación o Id (*Identification)* de estos. Este proyecto evalúa comparativamente cinco protocolos de enrutamiento geográfico

2.3. Selección de protocolos de enrutamiento geográfico

En la bibliografía existen diferentes protocolos de enrutamiento geográfico, debido a esto fue necesario realizar un estudio en el que se pudiese observar la evolución de las distintas subfamilias de protocolos; al realizar esta tarea se elaboró la tabla del Anexo 11 en la que se muestran los protocolos, las subfamilias a las que pertenecen y la documentación que los analiza. El diagrama puede ser visto como una línea de tiempo de la evolución los protocolos, en el que se muestra como han sido modificados los protocolos existentes.

Haciendo uso de este esquema se procuró elegir los protocolos que cumpliesen con las siguientes características o criterios:

- La actualidad de los protocolos de las distintas subfamilias que hacen parte del enrutamiento geográfico.
- Deben tener una buena cantidad de bibliografía, la cual proceda de fuentes confiables y al mismo tiempo se encuentre actualizada.
- Deben ser referenciados en diferentes fuentes bibliográficas.

Los protocolos elegidos son; *2-f-GEDIR (2 hop flooding GEographical DIstance Routing), 2-f-MFR (2 hop flooding Most Forward with in Radius), GEAR (Geographic and Energy Aware Routing) ,GAF (Geographical Adaptative Fidelity) y GDSTR (Greedy Distributed Spanning Tree Routing).*

2.4. Protocolos: 2-f-GEDIR y 2-f-MFR [10]

Los protocolos 2-f-GEDIR y 2-f-MFR pertenecen a la familia de protocolos *Greedy*, cuya principal característica es la de tomar decisiones de enrutamiento locales basándose en la información sobre la posición del nodo que transmite actualmente, la de sus vecinos y la del nodo destino de los paquetes de aplicación. Estos protocolos utilizan los métodos de "*2-hop"* y "*flooding"*, lo que los hace versiones mejoradas de sus antecesores GEDIR y MFR respectivamente, debido a que estos últimos no eran *loop free* (libre de bucles) y no eran eficientes al momento de elegir la ruta más corta al nodo destino. En el caso de GEDIR cada nodo a su turno elige el siguiente nodo de transmisión, según la distancia euclidiana al nodo destino, para eventualmente alcanzar dicho destino, siempre que sea posible. Por otro lado, para MFR, el criterio que se toma en cuenta para elegir el siguiente nodo de transmisión, es la menor métrica

al nodo destino, esta métrica se denomina comúnmente como progreso y será explicada en las siguientes secciones.

2.4.1. Características de 2-f- GEDIR y 2-f-MFR

- Cada nodo conoce (para un espacio de dos dimensiones) su Id, su posición actual, la posición y Id de sus vecinos a un salto, la posición y Id de sus vecinos a dos saltos (2-hop) y la posición y Id del destino.
- Son *loop-free* (libre de bucles) debido a que usan los métodos de *2-hop* y *flooding* (inundación).
- La métrica de GEDIR es la distancia euclidiana hacia el destino.
- La métrica de MFR es el progreso hacia el destino.

2.4.2. Funcionamiento General

Los protocolos 2-f-GEDIR y 2-f-MFR son variantes del algoritmo *Greedy*, en el cual los nodos mediante un proceso inicial de reconocimiento, adquieren la latitud y longitud de los nodos vecinos. Inicialmente todos los nodos de la red conocen su posición (coordenadas X y Y), y el Id y la posición de sus vecinos a uno y dos saltos, donde los vecinos a dos saltos, *2-hop*, son aquellos nodos que se encuentran directamente conectados a los vecinos que están dentro del área de cobertura del nodo que transmite, por lo que se deduce que los vecinos a dos saltos no son alcanzables por el nodo que está transmitiendo. En la Figura 3 se puede observar que el nodo 3 es un vecino a *2-hop* del nodo 1.

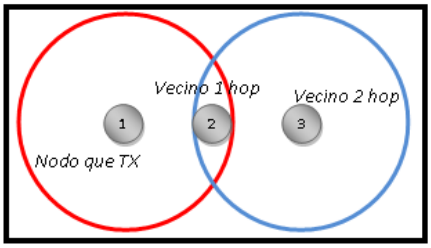


Figura 3 Vecino a *2-hop*

Cuando a un nodo envia un paquete a un nodo destino, este primero calcula y elige entre los nodos a dos saltos aquel que tenga menor métrica al nodo destino, para luego enviar el paquete a este nodo *2-hop* por medio del nodo vecino directamente conectado por quien conoció al nodo a *2-hop*, en caso de exista más de un nodo directamente conectado para llegar a este, se elige aquel que esté más cercano al destino. La métrica definida para 2-f-GEDIR es la distancia euclidiana al destino, y para 2-f-MFR es el progreso, el cual es la distancia del nodo destino a la proyección del nodo a dos saltos sobre la recta que conecta al destino con el nodo que transmite actualmente, de tal manera que el protocolo se encarga de que un paquete sea enviado al nodo vecino con el menor progreso. Los algoritmos se detienen si la mejor opción de re-envió para el nodo actual, es el nodo del cual recibió el mensaje; en este caso se dice que el nodo

actual es nodo un cóncavo. Para evitar entrar a un nodo cóncavo y por lo tanto evitar entrar en un *loop* (bucle), en donde el paquete regresa repetidamente a este nodo sin encontrar una mejor opción para enviar el paquete, los protocolos 2-f-GEDIR y 2-f-MFR proponen una solución en la que el nodo identificado como cóncavo al no encontrar una mejor opción para reenviar el mensaje, lo replica a todos sus vecinos a un salto (i.e dentro de su rango de cobertura); esto se hace mediante el uso de un *flooding* (inundación) restringido del paquete, donde este llega solo a los vecinos directos y ellos al recibirlo e identificarlo como un paquete de *flooding*, no tendrán en cuenta en sus cálculos al nodo cóncavo del cual proviene el mensaje y nuevamente cada uno buscará entre sus vecinos *2-hop* aquel que tenga la menor métrica al destino y reenviará el paquete a ese nodo.

2.4.3.Funcionamiento detallado

El protocolo GEDIR (antecesor de 2-f-GEDIR) funciona de la siguiente manera, inicialmente el nodo S , Figura 4, quien quiere enviar un mensaje al nodo D, y al no tener a D entre sus vecinos directamente conectados, (los vecinos de S son los que se encuentran dentro de su área de cobertura, circunferencia verde, también se identifican por las conexiones verdes, los vecinos de D se encuentran dentro de la circunferencia azul) analiza cual de sus vecinos se encuentra más cercano al nodo D, en este caso el nodo A (ya que la distancia d1 es menor que la distancia d2), y reenvía el mensaje a este nodo, luego A nuevamente analiza cual de sus vecinos se encuentra más próximo a D y reenvía el mensaje, y todos los nodos ejecutan la misma orden hasta alcanzar el destino. En este caso el camino que el protocolo elige es S-A-B-C-D (Figura 3). Sin embargo, como se puede observar de la Figura 5 el camino óptimo sería S-B-C-D. Por lo tanto para elegir el mejor camino se utiliza el método de *2-hop* en el cual se hace uso de los vecinos a dos saltos del nodo S y es implementado en 2-GEDIR. Mediante el uso de un protocolo de reconocimiento (se explica en la sección 4.1.4) se realiza un conocimiento previo de la red y de esta manera el nodo obtiene la posición de sus vecinos directamente conectados y de sus vecinos a dos saltos. De esta manera inicialmente el nodo S, halla a C como el vecino a dos saltos más cercano al destino, y debido a que B es vecino tanto de C como de S, es elegido como el vecino a un salto para reenviar el paquete a C.

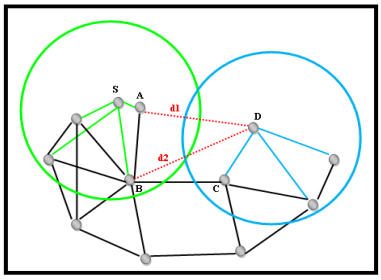


Figura 4 Vecinos de S y D[3]

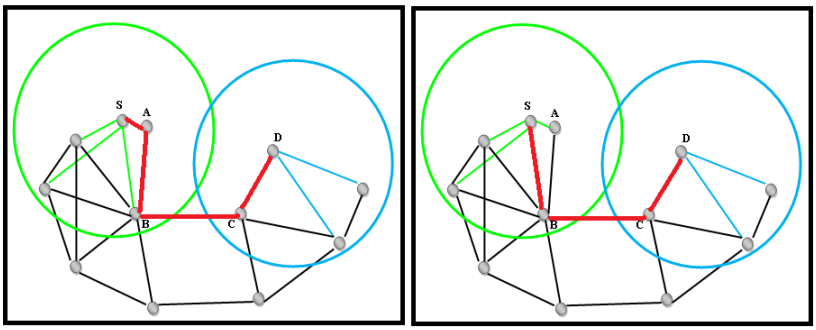


Figura 5 Protocolo GEDIR (S-A-B-C-D) y 2-f-GEDIR (S-B-C-D)

Por otro lado MFR funciona analizando el progreso: el progreso se define como la distancia de D a A', Figura 6, siendo A' la proyección del nodo A sobre la línea que conecta a S con D, es decir la distancia d1. Esta distancia se define de la siguiente manera [16]:

$$d1 = \frac{\overline{DA} \cdot \overline{DS}}{\left\| \overline{DS} \right\|} \qquad (1)$$

El objetivo de este protocolo es realizar un enrutamiento en donde se minimice la métrica. Luego el enrutamiento se efectúa de igual forma en todos los nodos de la red. Para MFR se presentaba la misma situación en donde el camino elegido podía no ser el más eficiente para llegar al destino, por lo que se implementa el método de *2-hop* y se obtiene el protocolo 2-MFR.

[3] Figura de elaboración propia basada en [13]

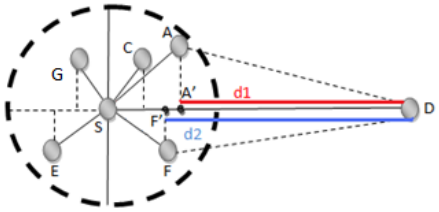


Figura 6 MFR: Progreso de los nodos vecinos a S[4]

El número de saltos hacia el destino de GEDIR y MFR se mejora haciendo que los nodos intercambien información acerca de sus vecinos y cada nodo sea consciente de sus vecinos a dos saltos (los vecinos de sus vecinos). En este caso el nodo S, quien posee el mensaje, puede escoger el nodo más cercano (o con menor progreso) al destino D entre todos sus nodos directos (a un salto) y los nodos vecinos a dos saltos; cuando se haya seleccionado el nodo que se encuentre más próximo a D, el mensaje se enviará inicialmente al vecino que se encuentra conectado directamente, para que este lo re-envié o al destino (en el caso que él mismo sea el nodo más cercano) o al nodo que se encuentra a 2 saltos del nodo de quien originalmente recibió el mensaje. En caso de empates, es decir que para llegar al vecino de dos saltos existe más de un vecino a un salto, se escoge el nodo que se encuentre más cerca al destino. El mensaje deja de ser reenviado por el nodo intermedio X al vecino con menor métrica al destino si, el nodo Y elegido por X para reenviar el mensaje, es exactamente el nodo que previamente había enviado el mensaje a X. En este caso se habla de un nodo cóncavo. Para evitar que el mensaje se pierda en este nodo, la mejora que proponen 2-f-GEDIR y 2-f-MFR, es utilizar un campo de nodo cóncavo en el mensaje y recurrir a una solución *flooding* (inundación), en la que el mensaje se replica a todos los vecinos del nodo que ejecutó el *flooding*. Un *flooding* total, iniciado en el nodo cóncavo y ejecutado después por cualquier nodo que haya recibido el mensaje, sin duda será suficiente para alcanzar el destino, pero la tasa de *flooding* será afectada, ya que el mensaje se replicara tantas veces como el número de vecinos que tenga el nodo que inició el *flooding*, por lo que si un nodo tiene 5 vecinos inicialmente existirán 5 copias del paquete, luego cada copia se multiplicará por el número de vecinos que tenga cada nodo que recibió el paquete. Así se propone realizar un *flooding* restringido solo en los nodos cóncavos, lo que significa que el nodo cóncavo reenvía el mensaje únicamente a todos sus vecinos directamente conectados y no inunda toda la red, ya que estos no lo reenvían haciendo uso de un *flooding*, sino haciendo uso de la métrica de 2-f-GEDIR o 2-f-MFR para renviar el paquete. Los nodos vecinos identifican el nodo cóncavo debido a que el campo de nodo cóncavo del mensaje esta marcado con el id de este, de esta manera no lo tienen en cuenta al momento de calcular las métricas para el re-envió. Ya identificado el nodo cóncavo, cada nodo calcula nuevamente cual de sus vecinos tiene menor métrica al destino y reenvía el paquete a este nodo; a partir de este punto el protocolo continúa con su funcionamiento habitual.

[4] Figura de realización propia basada en [11]

2.4.4. Diagrama de flujo

En el diagrama de flujo del Anexo 1 se explica el funcionamiento de los protocolos 2-f-GEDIR y 2-f-MFR.

2.5. Protocolo GEAR [12]

GEAR es un protocolo de enrutamiento geográfico perteneciente al grupo de protocolos de geocasting. Este grupo se caracteriza por proponer mecanismos de enrutamiento para entregar mensajes a todos los nodos de una región destino. La métrica que GEAR utiliza tiene en cuenta el nivel de energía y la posición de los nodos, para tomar decisiones de enrutamiento eficientes en energía.

2.5.1. Características principales de GEAR

- Supone que los nodos de la red son estáticos.
- Cada nodo conoce tanto la posición y energía de sus vecinos, como la propia. Con esta información cada nodo calcula el costo estimado de cada vecino al destino.
- Cada nodo actualiza la información de su nivel de energía luego de transmitir un paquete.
- Permite obtener una alta tasa de entrega y un eficiente uso de energía de la red de sensores. [9]

2.5.2. Funcionamiento general

El proceso de enrutar paquetes hacia los nodos de una región especifica consiste de dos fases, la primera se encarga de enrutar el paquete a través de la red; para efectuar esto los nodos involucrados en el proceso de transmisión de paquetes, calculan una métrica que tiene en cuenta la posición geográfica de sus vecinos y el nivel de energía que estos poseen. En esta fase existen dos casos a considerar en el nodo que tiene el paquete:

a) Cuando existe un nodo con menor costo al destino entre todos sus vecinos a un salto: el paquete se envía a este nodo.
b) Cuando todos sus vecinos tienen un mayor costo hacia el destino que él mismo: En este caso, se dice que existe un "hueco", GEAR escoge el vecino a un salto cuyo costo al destino sea el menor entre todos sus vecinos.

La segunda fase se encarga de diseminar el paquete dentro de la región destino, de este modo el operador de la red o la aplicación que está siendo usada, envía mensajes a todos los nodos de esa zona; para lograr esto en la mayoría de las caso se utiliza un algoritmo de direccionamiento geográfico recursivo. Sin embargo cuando la densidad de nodos en la red es baja, se usa un *"flooding"* restringido. Para el caso de este proyecto, no es de interés la manera en la cual se disemina el mensaje en la región destino, sino la manera en la cual el paquete llega a la región.

2.5.3.Funcionamiento detallado

En la primera fase, se asume que el nodo *N* esta reenviando un paquete P cuya región destino es R. Como se dijo anteriormente el objetivo de este proyecto no es el de estudiar la forma en la cual los paquetes son diseminados en la región destino, por esto la región se analiza como un solo nodo, específicamente como el nodo destino.

Al recibir un paquete P, el nodo *N* envía P progresivamente hacia la región destino y al mismo tiempo intenta balancear el consumo de energía a través de todos sus vecinos escogiendo el mínimo valor del costo aprendido para cada vecino *Ni*. Cada nodo *N* mantiene el costo el cual se denomina el costo aprendido a la región R y actualiza su valor cada vez que se encuentre involucrado en el proceso de transmisión de un paquete. Si un nodo no conoce para un vecino *Ni*, este calcula el costo estimado y lo usa como valor predeterminado para , definiendo el costo estimado de la siguiente manera (2):

(2)

Donde α es un peso ajustable, es la distancia euclidiana desde *Ni* hasta la región R normalizada por la mayor distancia entre todos los vecinos de *N*, y es la energía consumida en el nodo *Ni* normalizada por el mayor consumo de energía entre los vecinos de *N*.

Después que un nodo elige un vecino a un salto *Nmin*, el establece su propio a:

(3)

Donde el último termino es el costo de transmitir el paquete desde *N* hasta *Nmin* y envía un mensaje de actualización en el cual informa a todos sus vecinos el valor de su energía restante, de esta manera estos pueden actualizar el costo estimado de este nodo al destino. También puede ser una función que combina los niveles de energía restantes de *N*, *Nmin* y la distancia entre estos dos vecinos.

Para minimizar la función de costo estimado se tienen en cuenta los siguientes aspectos:

- Cuando todos los nodos tienen igual energía, se ejecuta a un enrutamiento geográfico clásico *Greedy*, en el que se reenvía el paquete al vecino más cercano al destino.
- Cuando todos los vecinos son equidistantes, esto conlleva a en una balanceo de carga (*load splitting*) entre vecinos, para minimizar el consumo de energía en la red. Dado que GEAR hace decisiones de reenvió solo basado en conocimiento local se obtiene una aproximación al camino que tenga el menor costo en cuanto al consumo de energía.

Cuando el nodo tiene el costo aprendido o el costo estimado para cada vecino, los casos de reenvió de paquetes son los siguientes:

a) Cuando existe un vecino más cercano al destino:

Cada vez que un nodo *N* recibe un paquete, el nodo escogerá el próximo salto entre sus vecinos que están más próximos al destino, minimizando al mismo tiempo el valor del costo aprendido . Sin huecos, el costo aprendido de cada vecino es la combinación de energía consumida y de la distancia al destino.

b) Cuando todo sus vecinos están más lejos del destino que él mismo:

En este caso el nodo *N* está en un hueco, por lo tanto el costo aprendido debe ser actualizado a todos sus vecinos para que el siguiente paquete que estos envíen, no regrese al nodo que se encuentra en el hueco. Intuitivamente, cuando no hay un hueco en el camino hacia R, el costo aprendido del nodo es equivalente al costo estimado . Sin embargo, cuando existe un hueco en el camino hacia R, el costo aprendido del nodo que se encuentra en el hueco es mayor al de sus vecinos, por lo tanto existe una resistencia a seguir el camino hacia él. Por lo tanto cuando se detecta un hueco el nodo cambia su costo propio a (2) y lo propaga,

(4)

de esta manera la red tiende a converger a la mejor ruta después de cierto tiempo de estar enviando paquetes. El costo del nodo que se encuentra en el hueco se actualiza y su valor se envía a los vecinos directamente conectados para que estos cambien el valor de costo estimado de este nodo. Cuando el paquete ha llegado a la región destino el objetivo del enrutamiento se ha cumplido.

Ejemplo:

Suponga que la distancia entre dos vecinos más cercanos es 1, y cada nodo puede alcanzar a cada uno de sus 8 vecinos, Figura 7. Los nodos en negro (G, H, I) son nodos cuya energía está agotada, por lo cual no pueden transmitir paquetes. El nodo S quiere enviar un paquete a T. De nuevo por simplicidad, se ilustra el algoritmo usando enrutamiento *Greedy*, estableciendo α en 1 y usando distancia euclidiana sin normalizar.

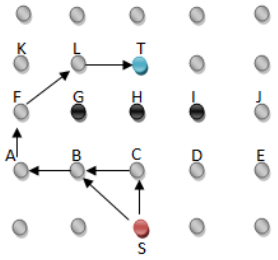


Figura 7 Ejemplo GEAR[5]

Inicialmente, en un tiempo 0, para el nodo S, los vecinos que se encuentran más cerca a T son B, C, D y sus costos respectivos son, Figura 8:

- ; ;

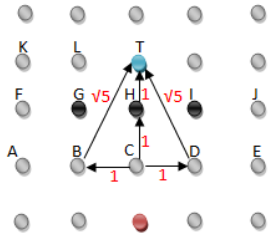


Figura 8 Costos aprendidos y estimados.

Al recibir un paquete destinado a T, S lo reenvía al vecino que tenga el costo más bajo, es decir al nodo C. En C, existe un hueco ya que todos los vecinos de C se encuentran más lejos de T que él mismo, por lo cual reenvía el paquete al nodo con el mínimo . Cuando hay empates, se decide por medio de alguna orden predefinida (ejemplo, ID del nodo). Para este caso se escoge al nodo B como siguiente salto y luego C, actualiza su propio costo aprendido , donde y es decir que el costo de la transmision de un salto es 1. Entonces el primer paquete que es enviado por S, llega al destino pasando por los nodos, S-C-B-A-F-L-T.

En un tiempo posterior, en un tiempo 2, cuando el nodo S recibe el paquete destinado a T, los valores de costo aprendido a sus vecinos serán:

-

De este modo, esta vez el nodo S reenvía el paquete directamente a B en vez de a C para evadir el hueco. Cada vez que un paquete es entregado a un nodo involucrado en el proceso de retransmisión, la energía restante del nodo será transmitida a todos los vecinos directamente conectados y luego el paquete de información será enviado. Se debe notar que la convergencia del costo aprendido no afecta el hecho de

[5] Figura de creación propia basada en [10]

que el paquete sea enrutado exitosamente fuera de los huecos, únicamente índica que tan efectivo es el enrutamiento evadiéndolos. La acción de propagar la actualización del valor de los costos aprendidos, permite que exista una posibilidad de evadir los huecos más rápidamente, y al mismo tiempo evitar agotar la energía de los nodos cercanos a los huecos.

Finalmente, en la segunda fase cuando el paquete alcanza la región destino, puede ser propagado usando una transmisión geográfica recursiva (*Recursive Geographic Forwarding*) o con una inundación restringida (*Restricted Flooding*). La inundación restringida, consiste en propagar un mensaje de tipo *broadcast* dentro de la región destino y se usa generalmente cuando la densidad de la red es baja, debido a que al inundarla de mensajes los nodos consumen una mayor cantidad de energía al escuchar cada mensaje de *flooding* que generan sus vecinos. Para redes de alta densidad es mejor usar la transmisión geográfica recursiva, ya que es más eficiente en energía. La transmisión geográfica recursiva consiste en dividir la región destino en cuatro subregiones, de tal manera que se crean cuatro copias del paquete que debe ser diseminado, así cada subregión se encarga de diseminar su propio mensaje y el número de nodos que efectúan el *flooding* será menor, por lo que el número de paquetes que procesará cada nodo también será menor.

2.5.4. Diagrama de flujo

El diagrama de flujo de GEAR se observa en el Anexo 2.

2.6. Protocolo GAF [9]

Es un algoritmo basado en posición que reduce el consumo de energía en redes *Ad Hoc* y es diseñado principalmente para redes móviles, pero puede ser aplicado en redes de sensores inalámbricas. La reducción del consumo de energía, se realiza identificando grupos de nodos cuya característica es que todos los nodos de un grupo pueden comunicase entre ellos, posteriormente se escoge uno de ellos para manejar el enrutamiento y los demás al ser innecesarios se apagan; de esta manera el tiempo de vida de la red se prolonga. GAF también puede ser considerado como un protocolo jerárquico, debido a qué en un momento el nodo que se encarga del enrutamiento tiene mayor importancia que los nodos que se encuentran apagados; sin embargo GAF no acumula información como si lo hacen los protocolos jerárquicos.

2.6.1. Características de GAF

- Los nodos de la red son móviles, pero por simplicidad son considerados estáticos.
- Cada nodo conoce su posición actual, su energía, y la posición de sus vecinos.
- Utiliza grillas virtuales dentro de las cuales los nodos cambian su estado de actividad.
- Permite un eficiente uso de la energía de la red de sensores.

- También puede ser considerado un protocolo jerárquico [3].
- Existen perdidas de datos no intencionadas.
- GAF supone una red con densidad alta de nodos, debido a que la vida de la red incrementa proporcionalmente su densidad.

2.6.2.Funcionamiento General

Inicialmente el área se divide en zonas fijas formando grillas virtuales. Dentro de cada zona, los nodos se comunican entre ellos para elegir un nodo que permanezca activo por cierto periodo de tiempo, mientras que los demás entran a un estado inactivo. Este nodo únicamente se comunica con los nodos que se encuentran activos en las grillas adyacentes.

En GAF los nodos de cada grilla poseen tres estados diferentes, los cuales son:

- *Discovery* (Descubrimiento): Estado en el que los nodos determinan los vecinos de la grilla en la que se encuentran ubicados y se determina el nodo que pasará a estado *active*.
- *Active* (Activo): Estado donde un único nodo en la cuadricula es responsable de monitorear, reenviar paquetes y reportar lo que sucede en la zona.
- *Sleep* (Inactivo): Estado en donde el radio de los nodos está apagado y estos no realizan ninguna actividad.

Mediante el uso de distintos tiempos, almacenados en variables temporales, un nodo logra pasar de un estado a otro y de esta manera lograr el objetivo de ahorro de energía del protocolo.

GAF no se encarga del enrutamiento de paquetes en la red, ya que su función es la de hacer un uso eficiente de la energía de los nodos; debido a esto cualquier protocolo de enrutamiento de redes *Ad Hoc* puede ser ejecutado con GAF. El cambio de estado en los nodos es independiente del protocolo de enrutamiento en uso, por lo tanto puede existir perdida de información cuando un nodo está reenviando paquetes y cambia su estado a *sleep*, en este caso el protocolo de enrutamiento es el encargado de re-enrutar el tráfico en la red .

2.6.3.Funcionamiento detallado

A continuación se explica la manera en la que GAF define las grillas virtuales y con qué criterios se deciden los cambios del estado de los nodos.

Inicialmente GAF establece las grillas a las que pertenecen los nodos, esto se realiza determinando la equivalencia que existe entre estos. GAF usa información local para determinar la equivalencia entre nodos, por lo tanto los nodos conocen su posición relativa a otros nodos. No es trivial encontrar la

equivalencia (grillas) entre nodos en una red haciendo uso únicamente de información local, ya que los nodos que son equivalentes con algunos, pueden no ser equivalentes en comunicación con otros. Este fenómeno se da debido a que el nodo solo puede escuchar a los nodos que se encuentran dentro de su área de cobertura.

Para solucionar esto GAF define grillas virtuales, cuyo principio es que todos los nodos de una grilla pueden comunicarse con todos los nodos de las grillas adyacentes. Es decir el nodo 1 de la grilla A, Figura 9, puede comunicarse con cualquier nodo de la grilla B, pero no con 5 en la grilla C. Por lo que 2, 3 y 4 son equivalentes en comunicación con 1 y 5.

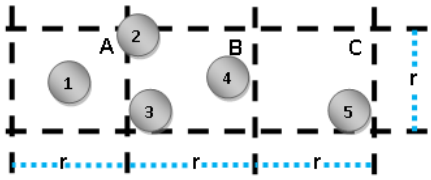


Figura 9 Ejemplo de grilla virtual en GAF[6]

En la vida real, el rango de comunicación de un nodo no es homogéneo o incluso simétrico debido a los efectos de propagación. Para simplificar el modelo del rango de comunicación se asume que es homogéneo.

El tamaño a las grillas virtuales es basado en un rango de comunicación de radio nominal R, el cual corresponde al rango de cobertura de la antena que posee el nodo. Se asume que la grilla virtual es un cuadrado con r unidades en cada lado. Para que se cumpla la definición de una grilla virtual la distancia entre los dos nodos más lejanos en cualesquiera dos grillas adyacentes no debe ser más grande que R.

Al observar la Figura 9, se deducen las siguientes formulas, que cumplen con la definición de grilla virtual:

(5) (6)

Asumiendo que *n* nodos se distribuyen en un área de tamaño *A*, el mínimo número de grillas virtuales *m* es:

(7)

De acuerdo al supuesto de una distribución uniforme de nodos, cada grilla tendrá al menos nodos:

[6] Imagen de realización propia basada en [10]

– —— (8)

Luego de que cada nodo define la grilla a la cual pertenece, debe identificar el estado en el que se encuentra. Los nodos en GAF pueden encontrarse únicamente en un uno de los siguientes estados: *Sleep, Discovery* y *Active*, Figura 10.

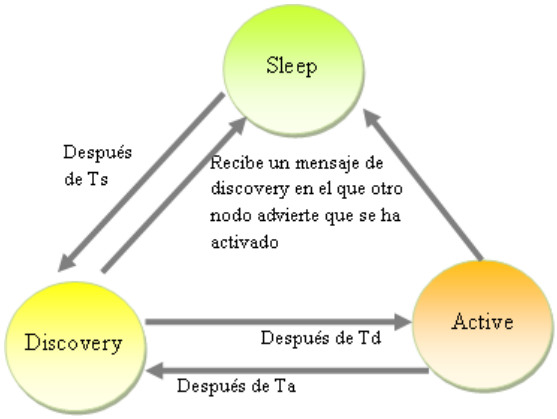


Figura 10 Maquina de estados de GAF[7]

Inicialmente los nodos se encuentran en estado *discovery,* en donde encienden su radio de transmisión e intercambian mensajes de descubrimiento para encontrar otros nodos que se encuentren en su misma grilla. El mensaje de *discovery* contiene: ID de la grilla, ID del nodo, tiempo estimado en estado activo (*enat)*, y estado del nodo.

En estado *discovery*, el nodo ajusta un temporizador a *Td (time discovery)* segundos, el cual índica el tiempo que permanece en este estado y así poder reducir la probabilidad de colisión de los mensajes de descubrimiento. Cuando el temporizador *Td* de un nodo en una grilla finaliza antes que el de los demás, ese nodo cambia a estado *active* y hace una transmisión tipo *broadcast* de su mensaje de descubrimiento a sus vecinos para que estos dejen el estado de *discovery* y pasen a estado *sleep*.

Cuando un nodo cambia a *active* ajusta un "timeout" en *Ta,* para definir el tiempo que el nodo se encuentra en estado activo. Al finalizar *Ta,* el nodo regresa a estado *discovery*. Mientras se encuentra *active* el nodo retransmite su mensaje de descubrimiento en intervalos de *Td* segundos.

Un nodo en estado *discovery* o *active*, puede cambiar a estado *sleep* cuando establece que otro nodo en su grilla se encargará del enrutamiento. Entre nodos de una misma grilla se elige quien se ocupa del enrutamiento, esto depende de una clasificación en la cual los nodos se encuentran ordenados arbitrariamente o de una clasificación hecha para optimizar el tiempo de vida de la red. Cuando el nodo

[7] Imagen de realización propia basada en [10]

pasa a estado *sleep* cancela todos sus temporizadores pendientes y apaga su radio. Luego de un tiempo *Ts* el nodo cambia de estado *sleep* a estado *discovery*.

GAF define ciertos parámetros para identificar en qué estado se encuentra el nodo, cuánto tiempo debe permanecer en ese estado, si es un nodo elegible para estar en estado *active* y cuanto es el tiempo de vida esperado del nodo entre otros.

- Enat (*Estimated Node Active Time*): Es el tiempo estimado en estado *active* de un nodo que se puede establecer al tiempo de vida esperado del nodo (*enlt*). Para mejorar el consumo de vida de la red, GAF utiliza una estrategia de balance de carga.
  - Balance de Carga: El objetivo de esta estrategia es lograr que todos los nodos de la red permanezcan activos el mayor tiempo posible. Para esto se supone que todos los nodos de la red son igualmente importantes. Por lo tanto el nodo activo ajusta su valor de *Ta* a *enat* y lo transmite en su mensaje de descubrimiento. Los nodos que no se encuentre activos en la grilla utilizan *enat* para determinar su periodo en el estado sleep (*Ts*). El valor de *enat* se establece en un tiempo menor que el tiempo necesario para usar toda la energía remanente del nodo (*enlt*). Generalmente su valor es *enlt/2*. De esta manera todos los nodos cambian a estado *discovery* al mismo tiempo, haciendo que la escogencia del nodo que estará en estado *active* se haga únicamente con base en la clasificación de energía.
- Td (Discovery Message Interval): Es un valor aleatorio uniforme entre 0 y una constante, que sirve para evitar las colisiones entre mensajes de descubrimiento. El rango de *Td* también puede ser influenciado por la clasificación (*ranking*) de los nodos, para fomentar que los nodos con clasificación alta repriman a los nodos con clasificación baja, permitiendo que estos últimos cambien a estado *sleep* más rápidamente.
- Ts (*Sleep Duration*): Es el tiempo que dura en el estado *sleep*. En GAF puede ser establecido igual a *enat* del nodo que se activo para lograr el balance de carga.
- Clasificación (*Node Ranking*): Se utiliza para maximizar el tiempo de vida de la red, seleccionando los nodos más apropiados para manejar el enrutamiento. Esta clasificación se hace con base en las siguientes reglas:
  - Los nodos en estado *active* tienen una mayor importancia que los nodos en estado *discovery*.
  - Cuando los nodos se encuentran en el mismo estado, tienen más importancia aquellos con un tiempo esperado de vida mayor (*enlt*). Es decir, esta regla hace que los nodos con mayor tiempo de vida sean seleccionados.
  - Al existir nodos con la misma importancia, se utiliza el id de los nodos para elegir cual pasará a estado *active*.

2.6.4. Diagrama de flujo

En el Anexo 3 se muestra el diagrama de flujo de GAF y en el Anexo 4 se muestra el diagrama de flujo del protocolo *Greedy* utilizado para enrutar los paquetes.

2.7. Protocolo GDSTR [11]

Es un protocolo que propone una forma de proceder en el enrutamiento del paquete al llegar este a un nodo cóncavo donde el algoritmo *Greedy* falla; dado este caso GDSTR cambia a un enrutamiento por medio de un árbol (*spanning tree*) hasta alcanzar un punto donde *Greedy* puede retomar el enrutamiento. Para escoger una dirección sobre el árbol la cual sea más conveniente para alcanzar el nodo destino, cada nodo mantiene información de las áreas cubiertas por cada una de las ramas de sus nodos hijos.

De igual manera, fue creado para garantizar entrega en el mundo real, sin hacer uso de grafos UDG (*Unit Disk Graph*), ni algoritmos que hagan uso de la regla de la mano izquierda. Esto se debe a que los grafos UDG asumen que todos los rangos de comunicación en la red son idénticos y uniformes, y por lo tanto no proveen una aproximación a un modelo real en el cual el consumo de energía genera un cambio en estos rangos.

2.7.1. Características de GDSTR

- Garantiza entrega paquetes en el mundo real.
- Cada nodo conoce su posición actual y la de sus vecinos.
- Usa un nuevo tipo se *spanning tree* llamado *hull tree*.
- Encuentra rutas cortas y para conservar el *hull tree* consume un menor tráfico, que otros protocolos de este tipo.
- Usa múltiples *hull trees*, para que la red tenga un mayor grado de tolerancia a fallas y el enrutamiento sea más eficiente.

2.7.2. Funcionamiento general

GDSTR reenvía paquetes de información usando un enrutamiento *Greedy* simple la mayor parte del tiempo. Debido a que el enrutamiento geográfico *Greedy* falla cuando el paquete llega a un nodo cóncavo, GDSTR conmuta a reenvío basado en *hull tree* solo para re-direccionar paquetes alrededor de estos huecos. Posteriormente conmuta de nuevo a enrutamiento *Greedy* tan pronto como le sea posible hacerlo.

Un *hull tree* es una clase de *spanning tree*, el cual contiene información adicional y se usa en redes donde cada nodo tiene una coordenada asignada. Esta nueva información se agrega cuando se crean los *Convex hull (*el *convex hull* de un conjunto de puntos es el area mas pequeña que los contiene*)* ya que cada nodo almacena la información del *convex hull* en el que se encuentra y también la información de todos sus nodos descendientes en el árbol, con sus respectivos *convex hulls* Figura 11.

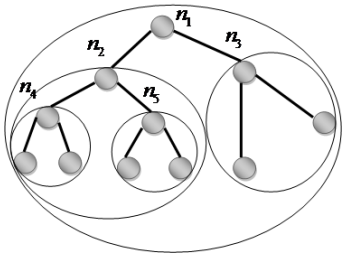


Figura 11 Nodos con sus *convex hulls*

Los *hull trees* proveen una manera de agregar información de localización y son construidos por medio de *convex hull*. Esta información es usada en el enrutamiento para encontrar una nueva ruta más rapidamente haciendo posible atravesar un sub árbol significativamente reducido, compuesto de nodos que se encuentran dentro de *convex hulls* y estos últimos a su vez que contienen el nodo destino.

2.7.3.Funcionamiento detallado

Para el correcto funcionamiento del protocolo de enrutamiento GDSTR, es necesario crear y mantener cada uno de los arboles que se definen. Ya una vez que cada nodo tenga la información respectiva de los arboles, el protocolo de enrutamiento puede hacer uso de esta para el re-envio de paquetes en el caso de encontrarse un nodo cóncavo en la ruta hacia el nodo destino. El protocolo define tres tareas principales: la primera es la creación de los arboles *hull trees* donde se definen y conocen el nodo raíz, el nodo padre y los nodos hijos de cada nodo y de que manera se comunica esta información entre nodos; la segunda es la creación de *convex hulls* es decir su definición y cálculos respectivos; y la última es la manera en que hace el re-envio de paquetes a través del *hull tree*.

a. Creación de Hull Trees:

Elección de nodos raíz

Se crean un par de *hull trees*, ya que como es mencionado en [ ], tener multiples arboles existentes aumenta el desempeño de enrutamiento de la red y la tolerancia a fallas siendo este aumento marginal después del segundo árbol. Esto se debe a que si se usa único árbol como la base del enrutamiento la red es inherentemente frágil, ya que si el nodo raíz falla, el árbol entero podría colapsar y tendría que ser reconstruido y el enrutamiento fracasaría.

Para escoger los nodos raíces se busca tener las raíces de los arboles lo mas separadas posibles ya que le da la capacidad al paquete de evadir huecos de manera más eficiente, es decir en menos saltos, escogiendo el nodo raíz más cercano al nodo destino.

Cada nodo emite un mensaje *keepalive* periódicamente para informar a sus vecinos de su existencia y de comunicar cambios de topología de la red. El nodo incluye en cada mensaje su vista de la raíz de cada árbol, tanto de su distancia en camino como en conteo de saltos y su *convex hull* asociado. La finalidad de este mensaje es la de mantener la estructura de los arboles creados.

El desempeño de la red es óptimo cuando el área en la que se superponen los *convex hull* de diferentes ramas, es mínima. Intuitivamente se quieren crear arboles "compactos" los cuales agrupen los nodos que se encuentren más cercanos a un mismo árbol. Así mismo se busca lograr que el número de saltos de desde el nodo raíz hasta los nodos de la red, disminuya.

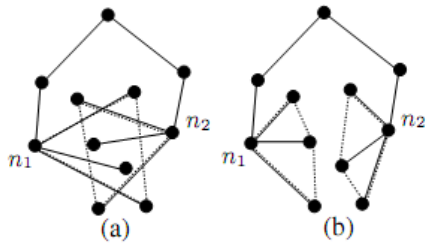


Figura 12 Ejemplo de un buen (b) y un mal (a) árbol[8]

Los autores de GDSTR[9], de los diferente algoritmos *spanning tree* existentes para lograr este objetivo, el que mejor se adapta a esta situación es el algoritmo del árbol del mínimo camino (*minimal-path tree*). Este algoritmo consiste en que cada nodo elige al vecino con el mínimo camino hacia el nodo raíz como su nodo padre.

b. Creación *convex hull*

Inicialmente todos los nodos hijos calculan sus *convex hull* propios. El *convex hull* para un conjunto de puntos es el area mínima que contenga todos los puntos (sus vértices); es mínima porque el *convex hull* es contenido en cualesquiera polígonos convexos que contengan los puntos dados. El *convex hull* puede ser calculado por diversos métodos matematicos[10]. Para la creación de los *convex hull* se tienen en cuenta los siguientes aspectos:

[8] Imagen tomada de [11]
[9] Ben Leong, Barbara Liskov y Robert Morris
[10] R. Graham. An efcient algorithm for determining the *convex hull* of a infinite planar set. Info. Proc. Letters, 1:132–133, 1972

I. Para asegurar que el *convex hull* solo usen la función de almacenamiento en vez de , donde *N* es el tamaño de la red, se limita el número de vértices para un *convex hull* a un máximo de *r* puntos.

II. Para reducir un *convex hull* de *s* vértices a uno con menos vértices, con , se pueden proyectar las líneas limitantes para formar un triangulo adyacente en cada cara. Se escoge el triangulo de menor área en este conjunto de *s* triángulos y se adiciona este triangulo al *convex hull*.

Limitar el número de puntos en un *convex hull* permite ahorrar almacenamiento, pero los hulls resultantes serán más grandes y esto incrementa la probabilidad de que los hulls de dos nodos hermanos en un árbol se intercepten. Estas intersecciones entre *convex hull*s son indeseables porque introducen ambigüedad en el proceso de enrutamiento y lo hacen menos eficiente.

Una vez que los nodos del árbol han formado sus *convex hulls* cada nodo transmite a todos sus vecinos su *convex hull* propio y el id de su nodo padre. Para calcular el mínimo *convex hull* de los nodos padre, un nodo determina la unión de los *convex hull* de sus nodos hijos en esa rama.

Una vez que la raíz de cada árbol adquiere los *convex hull*s de todos sus nodos hijos, el paso final para cada nodo es determinar un conjunto de *convex hulls* de conflicto, que se denota con , y adicionar esta información a sus mensajes *keepalive*. Esta información de los hulls de conflicto es propagada hacia abajo empezando desde la raíz; cada nodo, por turnos, informa a sus nodos hijos acerca de las intersecciones entre sus hulls y otros hulls conocidos. Una vez que la información acerca de los hulls de conflicto ha sido propagada, el *hull tree* está completamente construido y es consistente.

c. Enrutamiento

El protocolo inicia con enrutamiento *Greedy* simple hasta llegar a un mínimo local también llamado nodo cóncavo que se identifica cuando el nodo que recibe el mensaje es el más cercano en distancia al nodo destino, pero le es imposible alcanzarlo. En este punto conmuta a enrutamiento basado en hull trees y escoge un árbol para el enrutamiento de paquetes entre los existentes, de la siguiente manera:

- Aleatoriamente del conjunto de arboles existentes, se escoge uno que contenga a un nodo hijo con un *convex hull* conteniendo el nodo destino.
- De otra manera, si ninguno de los nodos hijos (en ningun árbol) tiene el *convex hull* que contiene el nodo destino, se escoge el árbol que tenga la raíz más cercana al destino.

Luego de haber escogido el árbol, existen los siguientes escenarios posibles:

I. El punto destino no está contenido en los *convex hull*s de ninguno de los nodos hijos: Se reenvía el paquete al nodo padre. Si el paquete alcanza un nodo cuyo *convex hull* contiene el

destino, será direccionado hacia abajo del árbol desde ese nodo para alcanzarlo. Si el paquete llega al nodo raíz y ninguno de los *convex hull*s de sus hijos contiene el destino, se concluye que el paquete no se puede entregar y es descartado.

II. **El punto destino está contenido en el *convex hull* de al menos alguno de sus nodos hijos:** Se ordenan los nodos hijos cuyo *convex hull* contengan el punto destino usando un ordenamiento fijo basado en identificadores de nodos. La decisión de enrutamiento depende de si el paquete de información fue reenviado anteriormente en modo *Greedy* forwarding y desde que nodo fue recibido:

- **Paquete se encontraba anteriormente en modo *Greedy* o fue recibido por un nodo padre:** Se reenvía el paquete al primer nodo hijo cuyo *convex hull* contenga el destino.
- **Paquete fue recibido por un nodo hijo (excepto último nodo hijo):** Se reenvía el paquete al próximo nodo hijo cuyo *convex hull* contenga el destino.
- **Paquete fue recibido por el último nodo hijo entre aquellos hulls que contienen el destino:** Se reenvía el paquete al nodo padre o al primer hijo si el nodo es el nodo raíz y no tiene nodo padre.

2.7.4. Diagrama de flujo

En el Anexo 5 se muestra el diagrama de flujo del enrutamiento de *Spanning Tree.*

3. ESPECIFICACIONES

3.1. Herramienta de Simulación: Opnet Modeler [20, 21]

Opnet Modeler provee un ambiente de desarrollo integral que soporta el modelo de redes de comunicaciones y de sistemas distribuidos. Tanto el comportamiento como el desempeño de sistemas modelados pueden ser analizados realizando simulaciones por eventos discretos. El ambiente de Opnet incorpora herramientas para todas las fases del estudio, incluyendo el diseño del modelo, simulación, recolección de datos y análisis de datos

Opnet hace uso de especificación de modelos para desarrollar una representación del sistema a ser estudiado y soporta el concepto de reutilización de modelos, de tal manera que la mayoría de los modelos se basan en modelos de menor nivel desarrollados con antelación y almacenados en librerías. Sin embargo, todos los modelos están basados en los conceptos básicos y en los bloques de construcción proveídos por el ambiente de Opnet.

La especificación de modelos se soporta con herramientas llamadas editores, los cuales capturan las características del comportamiento de un sistema a modelar. Para presentar el desarrollador de modelos

mediante una interfaz intuitiva, estos editores manejan la información requerida para modelar el sistema, en una manera que es paralela a la estructura de un sistema de red real. En consecuencia, los editores para la especificación de modelos se organizan jerárquicamente: los modelos construidos en el *Project Editor* (Editor de Proyectos) cuentan con elementos especificados en el *Node Editor* (Editor de Nodos); seguidamente, cuando se trabaja con el *Node Editor*, se usan modelos definidos en el *Process Editor* (Editor de procesos). El *Project Editor* trabaja los Modelos de Red, el *Node Editor* trabaja los Modelos de Nodo y *Process Editor* trabaja los Modelos de Proceso

3.1.1.Modelo de Red

El objetivo de este modelo es el de definir la topología de una red. Las entidades que se comunican entre ellas se denominan nodos y sus funciones especificas se desarrollan diseñando su modelo de nodo; un modelo de red puede contener nodos cuyo funcionamiento está basado en un mismo modelo de nodo, así mismo puede contener cualquier número de nodos con modelos de nodo son distintos. Un modelo de red se presenta en la Figura 13.

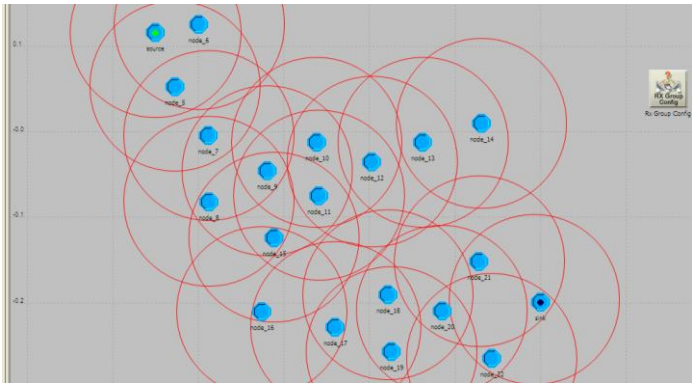


Figura 13 Modelo de Red

3.1.2.Modelo de Nodos

El modelo de nodos permite modelar los dispositivos de comunicación que pueden ser desplegados e interconectados en el nivel de red. Los modelos de nodos se desarrollan en el *Node Editor* (Editor de nodos) y se expresan en términos de pequeños bloques llamados módulos. Algunos de los módulos poseen características predefinidas y solo pueden ser configuradas mediante un grupo de parámetros intrínsecos del modelo; entre estos se incluyen varios transmisores y receptores, que permiten a un nodo vincularse a enlaces de comunicación en el modelo de red. Existen otros módulos llamados procesos, colas, y sistemas externos los cuales permiten ser programados fácilmente. Un modelo de nodo se puede apreciar en la Figura 14.

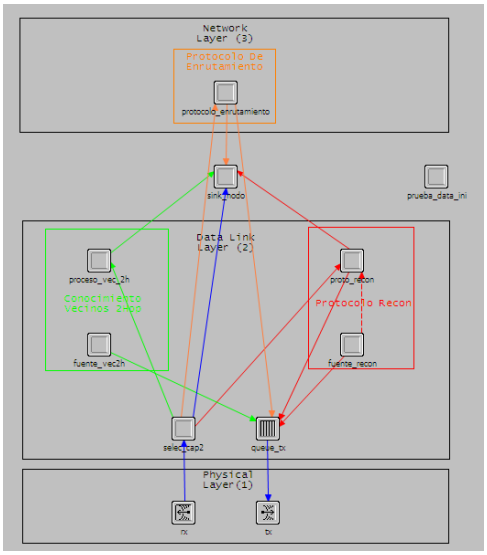


Figura 14 Modelo de nodo

3.1.3.Modelo de Procesos

Las tareas que ejecutan los módulos pueden ser programadas por el usuario, estas tareas son llamadas procesos y se definen como un grupo de instrucciones que se modelan mediante el *Process Editor* (Editor de Procesos). Se considera que un proceso está siendo ejecutado cuando realiza instrucciones que son parte de su modelo de proceso, solo un proceso puede ejecutarse a la vez. En la Figura 15 se muestra un ejemplo de Modelo de Proceso.

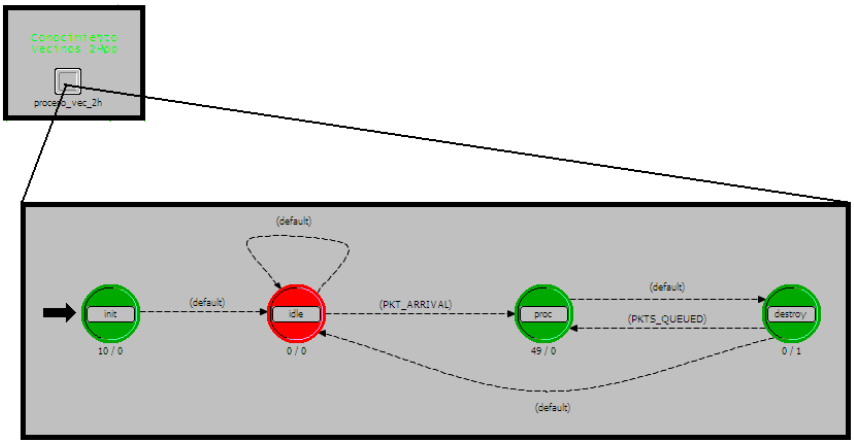


Figura 15 Modelo de proceso

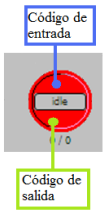
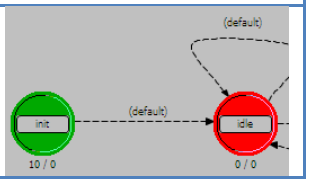
Los procesos en Opnet son diseñados para responder a una invocación o un *interrupt* (interrupción). Los *interrupts* son eventos que son dirigidos a un proceso y que le índican que debe realizar alguna acción. Los *interrupts* típicamente corresponden a eventos que se generan debido a llegada de mensajes, expiración de temporizadores o cambios de estados en otros módulos.

El *Process Editor* trabaja los modelos de proceso en un lenguaje llamado *Proto-C*, el cual es específicamente diseñado para soportar el desarrollo de protocolos y algoritmos. *Proto-C* se basa en la

combinación de diagramas de estados de transición (*state transition diagram STDs*), una librería de comandos de alto nivel conocidos como procedimientos del Kernel, y las facilidades generales del lenguaje de programación C o C++.

Los modelos de proceso hacen uso de una máquina de estados finita (*finite state machine FSM*) la cual representa el comportamiento lógico de un módulo, el proceso depende del estado actual y de los nuevos estímulos que pueden ocurrir. Las FSMs se representan haciendo uso de la notación de diagramas de estados de transición (STDs). Los estados del proceso y las transiciones entre ellos, se representan como objetos gráficos; los objetos usados para crear modelos de proceso se muestran en la Tabla 1.

Tabla 1 Objetos del modelo de procesos

Tipo de Objeto	Descripción	Representación
Estado	Representa un modo del proceso que ha sido realizado debido a estímulos previos y a decisiones correspondientes que lo involucren. Los estados contienen código que expresa el proceso que se ejecuta inmediatamente después de que se ha entrado a ellos, o inmediatamente antes de salir de ellos. Un estado puede ser forzado (color verde) o no-forzado (color rojo); un proceso en un estado no-forzado, se bloquea automáticamente luego de ejecutar el código de entrada, en este punto antes de seguir ejecutándose, el proceso se detiene mientras espera por un nuevo *interrupt*.	
Transición	Índica el posible camino que un proceso puede tomar desde un estado fuente a un estado destino. Cada estado puede ser fuente o destino de cualquier número de transiciones. Una transición posee una condición que especifica los requisitos para que el proceso ejecute transición. Una ejecutiva especifica las acciones que deben ser realizadas cuando el proceso efectúa la transición	

En el Anexo 10, se explica detalladamente las funciones de Opnet usadas para la realización del proyecto.

3.2. Consideraciones y especificaciones de los protocolos a analizar

Para observar el correcto funcionamiento de los protocolos se debe tener en cuenta que solo se estudia la manera en la que se efectúa el enrutamiento de paquetes, es decir solo se tiene en consideración la capa 3 del modelo OSI (*Open System Interconection*). Las capas inferiores (capa 1 y capa 2) no se tienen en cuenta debido a que: primero, el objetivo del proyecto es el de comparar los protocolos de enrutamiento entre ellos, es decir comparar ciertos parámetros que caracterizan el comportamiento de cada protocolo frente al enrutamiento de paquetes; de esta forma se puede analizar que protocolo es más eficiente en cada parámetro. Segundo, el análisis de capa 1 implica modelar un medio físico en el cual se tenga en cuenta la existencia de obstáculos, interferencias debido a otros medios de comunicación, colisiones de paquetes en el medio, entre otras características de un medio no ideal. Tercero, si se asume un medio no ideal, debe existir un protocolo de capa 2, que defina la manera en la que deben ser tratados los eventos que ocurren en capa1 y que interfieren con la comunicación.

Debido a estas razones se supone que:

- El medio es ideal: el rango de cobertura de propagación es una circunferencia perfecta, no existen obstáculos, no existen colisiones entre los paquetes.
- En la comunicación entre uno o varios nodos no existe perdida de información.
- El nodo tiene un *buffer* lo suficientemente grande para servir a todos los paquetes que llegan a él.
- No existe retardo de procesamiento en el *buffer*.
- Los paquetes no tienen prioridad es decir son servidos bajo un sistema FIFO (*first input first ouput*). Por lo que no se maneja un esquema de QoS (*Quality of Service*).

Debido a que el objetivo del proyecto es realizar una comparación entre los diferentes protocolos de enrutamiento, no es necesario hacer uso de un protocolo de capa 2 para analizar el efecto que este tendría las capas inferiores, ya que sería una constante común a todos los protocolos; por lo que la comparación no cambiaría de manera significativa al hacer uso de un mismo protocolo de capa 2. Si el objetivo del proyecto buscara analizar en detalle el comportamiento de un protocolo, es decir viendo su desempeño a través de todas las capas del modelo de referencia OSI, y no realizar un estudio comparativo del enrutamiento de paquetes entre los diferentes protocolos, sería necesario el uso de un protocolo de capa 2 que tenga en cuenta detalles más realísticos de la red, ya que bajo estas condiciones el desempeño del protocolo podría verse afectado.

El comportamiento de capa 2 en [12] no se tiene en cuenta debido a las razones explicadas anteriormente; en [9] se utiliza para analizar el consumo de energía sobre modelos de sensores comerciales, mas no para evaluar el enrutamiento que realiza el protocolo; en [10] se supone que para una red estática el medio es libre de colisiones y que cada mensaje generado es entregado, por lo tanto no se tiene en cuenta el comportamiento de capa 2; en [11] no se tiene en cuenta debido a que su objetivo es el de comparar el comportamiento básico del algoritmo, con el de otros protocolos de enrutamiento geográfico (como el caso de este proyecto).

Se debe tener en cuenta que el radio de cobertura de los nodos es de 100 mts, es decir los nodos pueden comunicarse con cualquier nodo que se encuentre dentro de su radio de cobertura.

Para que los nodos conozcan a los nodos vecinos que se encuentran dentro del área de cobertura, se realiza un protocolo de reconocimiento el cual es explicado en la sección 4.1.4.

3.3. Creación de escenarios

Para analizar el desempeño de los protocolos, inicialmente, se genera un escenario determinístico en el que todos los nodos se encuentran a una misma distancia, el tamaño de este escenario es de 700x700 mts y tiene 64 nodos en total, los nodos se despliegan desde las coordenadas (0,0), Anexo 23. Los protocolos 2-f-GEDIR, 2-f-MFR, GEAR y GDSTR se corren sobre este escenario y los resultados obtenidos se utilizan para identificar de manera general como varían los parámetros de la red, en cada protocolo. Debido a que el protocolo GAF supone el uso de una red con una alta densidad de nodos, fue necesario crear un

escenario diferente para poder evaluar su funcionamiento, el cual se muestra en la figura del Anexo 65. El nodo que posee un punto verde es aquel identificado como el nodo que genera paquetes, es decir el nodo fuente o nodo *source*; el nodo que posee un punto negro es el nodo que recibe los paquetes que género el nodo fuente, este representa a la estación base y es el nodo sumidero o nodo *sink.*

Posteriormente se generan diferentes escenarios aleatorios en los que se varía, el número de nodos y su despliegue (el despliegue siempre se inicia desde las coordenadas (0,0)). El tamaño de estos escenarios será siempre de 1000x1000 mts y se buscará variar la densidad de nodos dentro de esta área. Inicialmente se generan 50 escenarios cada uno con un total de 110 nodos, cada escenario tiene un despliegue diferente. Luego se generan 50 escenarios con despliegues diferentes y cada uno con un total 200 nodos. Finalmente se generan 50 escenarios con despliegues diferentes y cada uno con un total 280 nodos.

Tanto para los escenarios aleatorios como para los escenarios determinísticos la pareja nodo destino y nodo fuente se mantienen siempre a la misma distancia. El nodo fuente se ubica en (0.1,0.9) y el destino en (0.9, 0.1).

Debido a que se busca obtener resultados que puedan ser comparables, en total se generan 150 escenarios, de esta manera se puede observar en cada escenario como se comporta cada parámetro para cada protocolo y comparar los resultados obtenidos entre los parámetros de cada protocolo.

3.4. Parámetros a analizar

Luego de analizar los protocolos los parámetros que se eligen para evaluar comparativamente los protocolos, son los siguientes:

- **Tasa de entrega de paquetes**: parámetro que indica el número de paquetes de información que llegan al nodo sumidero contra el número total de paquetes de información que generó el nodo fuente.
- ***End to End delay*** (retardo de los paquetes): parámetro que indica el tiempo que tarda en llegar un paquete de información del nodo fuente al nodo sumidero
- ***IPDV* ó *Jitter***: parámetro que indica la diferencia entre el retardo de dos paquetes consecutivos. El IPDV (*Instantaneous Packet Delay Variation* [19]) es definido por la IETF en la RFC 3393 como la diferencia entre el retardo en un sentido, entre paquetes sucesivos.
- ***Hop count*** (número de saltos): parámetro que indica el número de nodos por los que viaja un paquete de información para llegar del nodo fuente al nodo sumidero; es decir la cantidad de veces que un paquete es recibido y transmitido por un nodo.
- **Porcentaje de utilización de los nodos**: parámetro que indica en que porcentaje fue usado un nodo. Este parámetro se calcula con respecto al número de paquetes de información que procesa un nodo, contra el número de paquetes de información que generó el nodo fuente.
- **Porcentaje de utilización de la red**: parámetro que indica el numero de nodos que fueron usados para transmitir paquetes contra el número total de nos de la red.

- **Nodos Activos**: Parámetro que indica el numero de nodos que se encuentran activos en la red.
- **Paquetes en nodos cóncavos**: parámetro que indica el número de veces que un paquete de información entro en un nodo cóncavo.
- **Energía local**: parámetro que indica la cantidad de energía consumida por un nodo.
- **Energía de la red**: parámetro que indica el nivel de energía restante de la red.

3.5. Proceso de Validación

Debido a que este proyecto reproduce el comportamiento de ciertos protocolos de enrutamiento, es necesario obtener un grado de confiabilidad entre aquello que es implementado, y aquello que es propuesto por los autores de cada protocolo.

Para lograr esto, se valida el funcionamiento de los protocolos con respecto a la teoría existente, es decir se verifica que la implementación de los protocolos sea la correcta. Para esto se busca extraer la lógica del el enrutamiento presentado en la teoría, verificando que el comportamiento del protocolo corresponda al esperado. Esta lógica se transcribe en un diagrama de flujo (pseudocódigo) para luego ser implementado en el lenguaje de programación requerido por el simulador.

En el proceso de validación se realizaron los siguientes pasos:

1. Estudio de los protocolos
2. Derivación de un pseudocódigo o diagramas de flujo.
3. Pruebas de escritorio del funcionamiento, paso a paso, sobre los escenarios de prueba.
4. Verificación de los resultados obtenidos contra los esperados según el comportamiento del protocolo.
5. Implementación en Opnet del procolo.
6. Ejecución en Opnet, paso a paso de los escenarios de pruebas.
7. Verificación de los resultados obtenidos contra los obtenidos anteriormente.
8. Ajuste tanto de diagramas de flujo como la implementación en Opnet en caso de existir discrepancias.

Inicialmente el proceso de validación se desarrolló estudiando el comportamiento de los protocolos elegidos, el cual se representó en diagramas de flujo o pseudocódigo (los diagramas de flujo se muestran en los Anexo 1-5); haciendo uso de la lógica de los diagramas de flujo, los protocolos fueron implementados en el simulador Opnet Modeler, luego se crearon escenarios de prueba para cada protocolo, y sobre estos se comprobó si las características del enrutamiento coincidían con lo que

presentaban los autores. Al tener los escenarios en papel y en el simulador y conociendo la posición de cada uno de los nodos, se verificaba si los cálculos que se realizaban en papel, coincidían con el enrutamiento que se realizaba en el simulador. Para validar que la implementación fuese correcta, se verificaba si los valores hallados en la prueba de escritorio (los cálculos realizados en papel) correspondían a lo que arrojaba el simulador, de no ser así se verificaba el pseudocódigo, y se realizaban las modificaciones necesarias para lograr el objetivo de enrutamiento del protocolo.

Luego de probar los protocolos sobre los escenarios de prueba, fueron ejecutados sobre los escenarios determinísticos (explicados en la sección 3.3), para poder evaluar las características del comportamiento de los protocolos bajo ciertos parámetros. Para esto se realizó una simulación en la que a medida que los nodos eran utilizados para el enrutamiento consumían energía, por lo que a lo largo de la simulación los nodos agotaban su batería y se apagaban, lo que producía que existiese un cambio de topología; para el caso del protocolo 2-f-GEDIR, la topología resultante luego de correr una hora y media de simulación, es el que se muestra en la Figura 16,

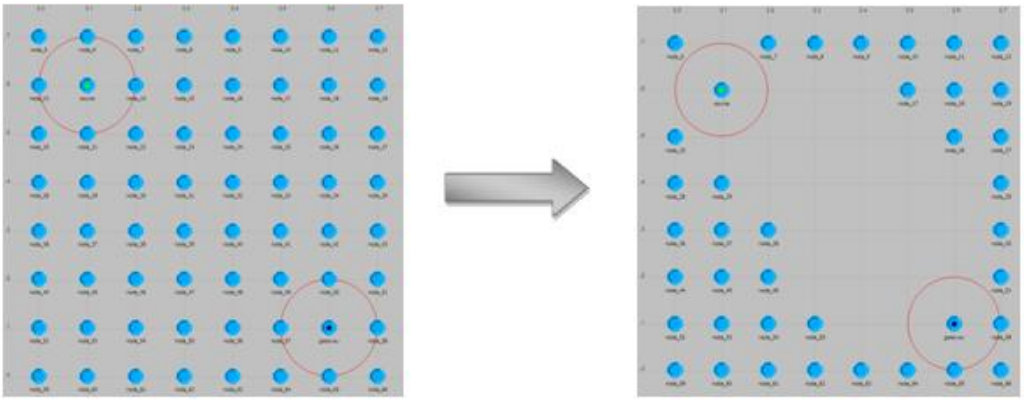


Figura 16 Cambio de topología del protocolo 2-f-GEDIR

Haciendo uso del Debugger del software varios parámetros pudieron ser analizados sobre una prueba de escritorio realizada para el escenario determinístico. Algunos de estos parámetros fueron Nodos Activos, ya que se esperaba que los nodos que agotaran inicialmente su energía fueran los de la diagonal, como efectivamente ocurrió; también se analizó el parámetro de nodos cóncavos, ya que se esperaba que al morir algunos nodos, se entrará en un nodo cóncavo; el parámetro de numero de saltos, fue uno de los más significativos para evaluar que el enrutamiento se estaba realizando de la manera esperada, ya que al identificar en la prueba de escritorio los nodos que debían encargarse del enrutamiento, se estaba calculando el numero de saltos que implicaba enviar un paquete del nodo fuente al nodo destino bajo cierta topología, conociendo esto se verificaba que en el simulador los nodos encargados del enrutamiento,

fuesen los mismos de la prueba. También fueron analizados los parámetros de energía, y porcentaje de utilización, entre otros.

Gracias al escenario determinístico, los protocolos pudieron ser analizados bajo los parámetros elegidos para posteriormente realizar la evaluación comparativa. Esto se realizó de esta manera debido a que, sobre los escenarios de prueba realizados inicialmente para cada protocolo, no podía realizarse una validación de los parámetros, por lo que el escenario determinístico puede ser visto como un ambiente controlable, en donde el comportamiento de los protocolos puede ser predecible.

En general para comprobar que la implementación de los protocolos fuese la correcta se comprobó que en el simulador el enrutamiento se realizara de igual manera que en los escenarios de prueba, para esto cada vez que en el simulador el protocolo implementado, tomaba una ruta diferente a la esperada, se analizaba que problema estaba ocurriendo y se cambiaba el proceso desde el pseudocódigo. El mayor problema que se presentó en la implementación fue el manejo de las listas que tenia cada nodo para conocer a los vecinos dentro de su área de cobertura, este tipo de errores se identificaban debido a que en algunas ocasiones los nodos enviaban de manera incorrecta el paquete o simplemente no lo enviaban.
Para los protocolos 2-f-GEDIR y 2-f-MFR, se presentó un problema en el que un nodo A que se encontraba en la lista de vecinos a un salto de un nodo B, también podía encontrarse en la lista de vecinos a dos saltos del mismo nodo B; para solucionar esto se eliminó al nodo repetido de la lista de vecinos a dos saltos. En el caso del protocolo GEAR, se identificaron ciertos problemas por la variación de los costos, es decir fue necesario ajustar los costos de trasmisión para que el enrutamiento siguiera el comportamiento descrito. La mayor dificultad del protocolo GAF, fue aquella de identificar a los nodos que se encontraban activos en las grillas vecinas del nodo que se encontraba reenviando paquetes, esto debido a que inicialmente los eventos programados dentro de una grilla se encontraban mal configurados, lo que hacía que no siempre hubiesen nodos activos en las grillas. El mayor problemas que presentó GDSTR fue aquel de hacer la reconfiguración de los arboles cuando un nodo agotaba su energía y se apagaba.

Las limitaciones con respecto al proceso de validación se encuentran en no tener la disponibilidad de datos externos con los cuales comparar, si aquello que ha sido implementado concuerda de manera precisa con aquello que describen los autores en los documentos. Dicho de otra forma, los autores de los protocolos no definen las características de un escenario que pueda ser reproducible en su totalidad, por lo que no es posible obtener los mismos resultados que se tienen en la teoría.
Así mismo existen parámetros, como el de Entrega de paquetes, End to end delay e IPDV, cuyo análisis bajo una prueba de escritorio es complicado, ya que implica conocer los tiempos de procesamiento de los

que hace uso el simulador en el momento de enviar paquetes; dicho en otras palabras el tiempo que tarda en llegar un paquete de un nodo a otro se define mediante el modelo del pipeline que trabaja Opnet, de tal manera que conocer estos tiempos, implica un análisis de los modelos predefinidos del simulador para este tipo de transmisiones.

Teniendo en cuenta estas limitantes se puede concluir que el proceso de validación definido permitió identificar los problemas que tenía cada protocolo y poder solucionarlos, y de esta manera tener una implementación apropiada de los protocolos y un buen grado de confiabilidad de los resultados obtenidos.

4. DESARROLLOS GENERALES

4.1. Generales

En esta sección se explica la manera en la que se desarrollaron los aspectos generales de cada protocolo, es decir aquellos ítems que son generales a todos los protocolos y que fueron realizados partiendo de diseños intrínsecos de Opnet o fueron diseñados a partir de la información encontrada en la bibliografía.

Opnet llama al grupo de características del receptor y del transmisor, *Pipeline*, y define ciertos atributos que pueden cambiar estas características para modelar un medio de comunicación entre un canal de transmisión y un canal de recepción. Para el caso de este proyecto el atributo del pipeline más usado es el del RX_GROUP. El pipeline se explica en profundidad en el Anexo 9.

4.1.1. Uso del "rx_group"

Como se explica en el Anexo 9 en el cual se describe el funcionamiento del Pipeline (grupo de características del módulo de recepción y del módulo de transmisión), el uso del RX_GROUP es fundamental para definir el área de cobertura de cada nodo. Como se muestra en el Anexo 8, el atributo "Distance Threshold" es el que permite definir la distancia máxima a la cual los paquetes enviados por el canal de transmisión pueden llegar a ser reconocidos por al canal de recepción. Para este caso, es fijada una distancia de 100mts [17], por lo que si un transmisor y un receptor están separados por una distancia mayor, no podrán intercambiar mensajes entre ellos.

4.1.2. Uso de nodo fuente y nodo destino

El desarrollo de las fuentes para todos los protocolos se hizo en base al módulo que Opnet posee llamado *"simple source"*, Anexo 6. El nodo fuente y el nodo destino se caracterizan porque en todos los protocolos son vistos como nodos con energía ilimitada.

En las aplicaciones de WSN, el nodo fuente es el encargado de enviar la información que el sensor recolecta, para posteriormente generar los paquetes que lleven esta información al nodo destino. La fuente

constantemente genera un paquete cada diez segundos, durante todo el tiempo de la simulación. El módulo que se encarga de este proceso se denomina "*Aplicacion_fuente*", y su proceso se basa en el modelo "*simple source*", Anexo 6 ; en este módulo se realiza la recolección de datos del número total de paquetes generados.

Debido al protocolo de reconocimiento, todos los nodos de la red conocen al nodo destino, es decir conocen su id y sus coordenadas (X ,Y) como se explica en la sección 4.1.4. El nodo destino posee un módulo de llamado "*Aplicación_sumidero*" en el que se realiza la recolección de datos de los paquetes que llegan exitosamente a él; el proceso de modelo de este módulo es el que se muestra en el Anexo 7, el cual es un modelo por defecto de Opnet, al que se le realizaron variaciones para efectuar la recolección de información que llega al nodo. En las aplicaciones de WSN este tipo de nodo es la estación base que recibe la información que generó el nodo fuente.

4.1.3.Modelo de consumo de energía

Para todos los protocolos se creó un modelo de energía, el cual se utiliza como un indicador del tiempo de vida promedio de cada nodo. En este modelo cada nodo posee 1000 unidades de energía y cada vez que se reciba un paquete dirigido al nodo destino, consumirá 1 unidad de energía y en el caso de reenvió consumirá 2 unidades de energía adicionales; de igual manera el nodo realiza un consumo continuo de 0.1 unidades de energía por segundo, al estar escuchando el medio, es decir al estar esperando la llegada de paquetes dirigidos al sumidero.

El modelo de energía en es utilizado para analizar el consumo de energía de cada nodo cuando son usados para el enrutamiento de paquetes; en el caso en el que un nodo se apague se envía un mensaje informando a los vecinos de esta eventualidad, estos a su vez eliminan de su lista de vecinos al nodo que se ha apagado y se encargan de re-enrutar los paquetes a un nodo que se encuentre encendido

4.1.4.Desarrollo de protocolo de reconocimiento de vecinos y del sumidero

El protocolo de reconocimiento permite que cada nodo conozca el id, la posición y la energía de los vecinos que se encuentran dentro de su área de cobertura.

4.1.4.1. Paquete de reconocimiento

EL protocolo de reconocimiento define un tipo de paquete *"pkt_recon"* que contiene los campos que se indican en la Figura 17.

Figura 17 Paquete de reconocimiento: *"pkt_recon"*

Tabla 2 Campos del paquete "pkt_recon"

Campo	Función del campo
Type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. El protocolo de reconocimiento maneja por defecto un "type" igual a cero.
IDorigen	id del nodo que genera los paquetes de *request*.
IDvecino	id del nodo que realiza el *reply*
Pos_x	Coordenada X del nodo del IDorigen cuando es un *request* ó la coordenada X del nodo del IDvecino cuando es un *reply*.
Pos_y	Coordenada Y del nodo del IDorigen cuando es un *request* ó la coordenada Y del nodo del IDvecino cuando es un *reply*.
Energía	Cantidad de energía del nodo del IDorigen cuando es un *request* ó la cantidad de energía del nodo del IDvecino cuando es un *reply*.
Flag	Valor que indica si el mensaje es de tipo *request* o *reply*.
Tag_sink	Valoe que indica si el paquete proviene del sumidero; es usado para el reconocimiento de éste.

4.1.4.2. Funcionamiento

El desarrollo del protocolo de reconocimiento, ejecuta un proceso que se encarga de generar un número determinado de paquetes *"pkt_recon",* basado en una comunicación *"request/reply"*. Para lograr esto los nodos de la red poseen dos módulos llamados *"fuente_recon"* y *"proto_recon"*, Anexo 14.

El módulo "*fuente_recon"*, se basa en el modelo de *"simple source"* de Opnet, Anexo 6, y se encarga del *request* de la comunicación creando los paquetes y marcándolos en el campo de flag con un cero, el cual índica que es un paquete que solicita a sus vecinos informar sobre su id, posición y energía; se aprovechan los campos de este mensaje para que el nodo que genera el paquete, de a conocer su información de IDorigen, pos_x, pos_y y energía a los vecinos que lo puedan escuchar. El campo de IDvecino no se marca en este proceso, debido a que es un paquete de tipo *broadcast* y lo que se busca es solicitar la información de todos los vecinos que puedan escuchar el mensaje. Por último el tag_sink se marca con un uno si el paquete viene de un nodo sumidero, con cero si es cualquier otro nodo de la red, de esta manera todos los nodos de la red conocen al nodo sumidero.

El módulo *"proto_recon"*, posee un modelo de proceso que se índica en la Figura 18, el cual inicialmente se encuentra en el estado de *"idle"* en donde el nodo espera la llegada de paquetes; cuando llega un paquete se pasa al estado de *"proc"*, aquí el paquete es procesado y posteriormente puede ser enviado al estado *"send_down",* en el caso en que deba ser reenviado a otro nodo, o al estado *"destroy"* en el caso en donde el paquete es destruido si no estaba dirigido al nodo que lo recibió .

En el estado de *"proc"* cuando recibe un mensaje *"pkt_recon"* marcado en el campo de flag con un cero (i.e paquete de *request*), obtiene la información del nodo que ha enviado el mensaje y luego en el campo de IDvecino coloca su id propio y cambia los campos de pos_x y pos_y, por sus coordenadas propias; el

IDorigen permanece con el id del nodo que genero el *request* y el campo de flag se cambia por un uno para índicar que es un mensaje de *reply*, luego el mensaje es enviado.

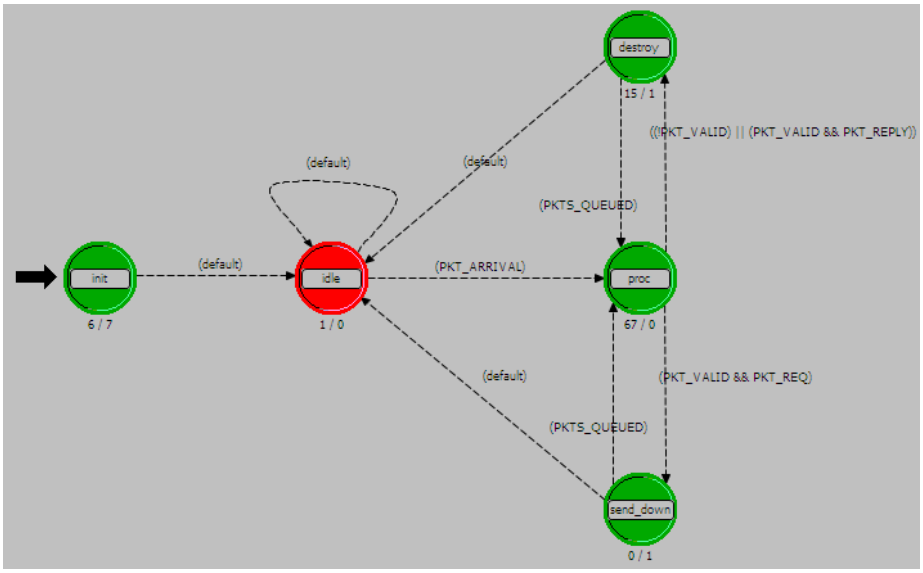


Figura 18 Modelo de proceso del módulo *"proto_recon"*

Si el flag del paquete está marcado con un uno, significa que es un mensaje de *reply*, el nodo adquiere la información y si Tag_sink es igual a cero elimina el paquete. En el caso que el campo del paquete Tag_sink este marcado con un uno, significa que el mensaje fue generado por el nodo destino (sumidero) y que el nodo debe guardar esta información sin importar si el nodo destino no es su vecino directo, en este caso el nodo reenvía el mensaje sin cambiar ningún campo del paquete. Cada vez que llegue un paquete que cumpla con las características de Tag_sink igual a uno y de flag igual a uno, el nodo revisará si él ya posee la información que se encuentra en el paquete y si es así el mensaje será eliminado.

4.2. Específicos

En esta sección se explica la manera en la que se desarrollan los protocolos de enrutamiento seleccionados. Inicialmente para cada protocolo se muestra su modelo de nodo, y se explica el desarrollo en base al modelo de procesos que fue creado para cada protocolo y en base al formato de paquete que se creó para llevar la información a través de la red.

4.2.1. Desarrollo de los protocolos 2-f-GEDIR y 2-f-MFR

El modelo de nodo que se desarrolla para los protocolos 2-f-GEDIR y 2-f-MFR, es el que se muestra en el Anexo 15, el funcionamiento de este bloque se explica en la siguiente sección. Se debe tener en cuenta que los módulos que se encuentran dentro del cuadro de color rojo, "protocolo recon", son los módulos en los que se ejecuta el protocolo de reconocimiento, explicado en la sección anterior.

4.2.1.1. Conocimiento de vecinos a dos saltos

Mediante el protocolo de reconocimiento de vecinos todos los nodos de la red conocen a sus vecinos directamente conectados, es decir los vecinos que se encuentran dentro de su área de cobertura. Los protocolos 2-f-GEDIR y 2-f-MFR para realizar el conocimiento de vecinos a dos saltos, es decir los vecinos de sus vecinos (*2-hop)*, hacen uso de la información adquirida mediante el protocolo de reconocimiento. Para realizar el proceso de reconocimiento de *2-hop* se desarrollaron dos módulos: *"fuente_vec2h"* y *"proceso_vec_2h"*, Anexo 16.

El módulo de *"fuente_vec2h"* genera paquetes con una estructura que contiene los campos indicados en la Figura 19:

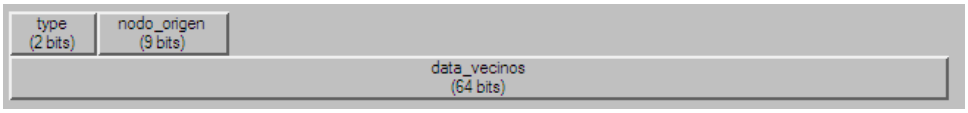


Figura 19 Paquete de reconocimiento vecino 2-hop *"vec2h_packet"*

Tabla 3 Campos del paquete *"vec2h_packet"*

Campo	Función del campo
type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. El protocolo de reconocimiento de vecinos *2-hop* maneja por defecto un type igual a uno.
nodo_origen	id del nodo que está informando su lista de vecinos
data_vecinos	lista de vecinos directamente conectados del nodo_origen.

Cada nodo en el módulo *"fuente_vec2h"* llena el paquete *"vec2h_packet"* en estos campos con su información y realiza un *broadcast*, los vecinos aceptan el paquete y obtienen la información contenida para luego utilizarla en el enrutamiento. El módulo que recibe el paquete es *"proceso_vec_2h"*, y su modelo de proceso es el que se índica en la Figura 20. Aquí el nodo en el estado de *"idle"* espera la llegada de paquetes, cuando recibe un paquete lo procesa en el estado de *"proc"*, y luego lo elimina en el estado de *"destroy"*. Debido a que dos nodos vecinos, pueden tener un tercer vecino en común, existirá una redundancia en las listas de vecinos a un salto y dos saltos, de los nodos que comparten el vecino. Para eliminar la redundancia, siempre que un nuevo vecino sea añadido a la lista de vecinos a 2 saltos, se verificará que este vecino no exista en la lista de vecinos a un salto, de tal manera que el vecino solo exista en la lista de vecinos a un salto y no en la de vecinos a dos saltos.

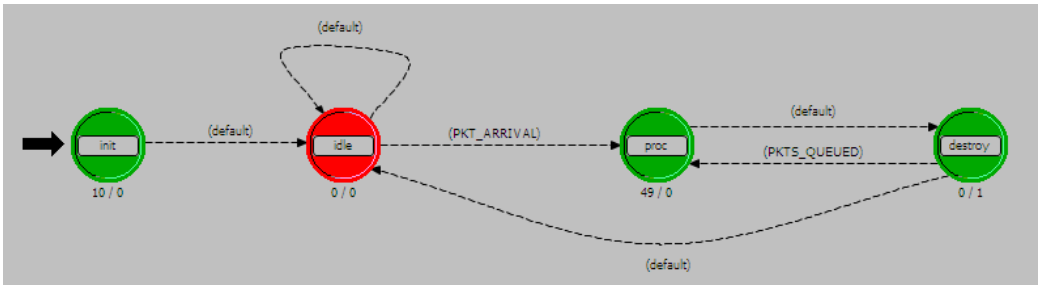


Figura 20 Modelo de proceso de *"proceso_vec_2h"*

Debido a que el id del nodo que se encuentra en el campo nodo_origen es un vecino que se conoció previamente gracias al protocolo de reconocimiento, la información de la lista de vecinos que se encuentra en data_vecinos se asociará al id de este nodo.

4.2.1.2. Desarrollo del protocolo y paquete de información

El módulo que recibe los paquetes de información que genera el nodo fuente es *"protocolo_enrutamiento"*, Anexo 6, y su modelo de proceso es, Figura 21.

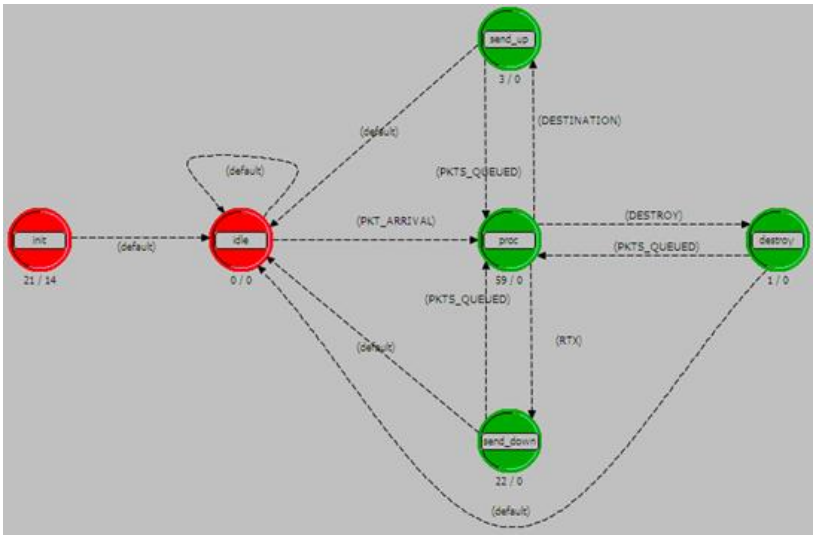


Figura 21 Modelo de proceso 2-f-GEDIR y 2-f-MFR

El paquete que genera la fuente *"aplicacion_fuente"* y que es utilizado para enrutar la información a través de los nodos de la red tiene la siguiente estructura, Figura 22:

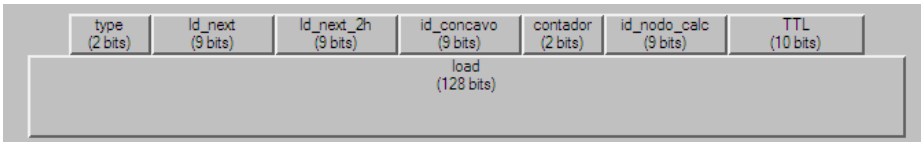


Figura 22 Paquete protocolos 2-f-GEDIR y 2-f-MFR *"pkt_GEDIR"*

Tabla 4 Campos del paquete *"pkt_GEDIR"*

Campo	Función del campo
Type	Tipo de paquete llega al receptor, para ser asignado al módulo encargado de su respectivo procesamiento. En general los protocolos de enrutamiento manejan por defecto un type igual a dos.
Id_next_2h	id del nodo a *2-hop* al cual el vecino a un salto debe enviar el paquete
id_concavo	Íd del nodo que se declara como nodo cóncavo
Id_next	Id del nodo a un salto al cual debe ser enviado el paquete
contador	Valor que indica si es un paquete de cálculos (contador igual a dos) o un paquete de reenvió (contador igual a uno).
Id_nodo_calc	Id del último nodo que hizo los cálculos para reenviar el paquete al nodo de *2-hop*
load	Contiene la información que generó el sensor.
TTL	Valor usado para limitar el número de saltos que un paquete puede realizar en la red.

Inicialmente el nodo se encuentra en el estado de *"idle"*, Figura 21, y cuando un paquete de información *"pkt_GEDIR"* llega al nodo, este lo procesa en el estado *"proc"* donde analiza si él es el sumidero de paquetes de la red, si es así los paquetes son dirigidos al estado *"send_up"* el cual los envía al módulo de capa superior *"aplicación_sumidero"*, aquí se considera que la vida de ese paquete en la red ha terminado. En el caso contrario, en el que el nodo que está procesando el paquete no es el sumidero, el nodo analiza si el paquete que llega es para él es decir si el campo de id_next está marcado con su id propio o con un valor que corresponde a un *flooding* (-99), si es así el nodo acepta el paquete y lo procesa. Luego el nodo toma el campo de id_concavo, ya que si el paquete contiene un valor de id valido en este campo el mismo paquete no deberá regresar al nodo que posee ese id, este caso se explica en la sección 4.2.1.3.

El campo que se toma posteriormente es el contador, ya que el nodo debe identificar si él es un nodo elegido para reenviar o si debe efectuar los cálculos necesarios para elegir el siguiente vecino a *2-hop* al cual debe ser dirigido el paquete. En el caso que contador sea igual a uno, el valor del campo id_next se cambia por el valor que se encuentra en id_next_2h, el valor de contador cambia a dos y el paquete es reenviado; si el contador es igual a dos el nodo inicialmente recorre su lista de vecinos a un salto verificando si el destino puede encontrarse entre ellos, si efectivamente el destino se encuentra entre sus vecinos a un salto el nodo marca el paquete en el campo de Id_next con el id del nodo elegido a un salto y cambia el valor de contador a 1; si el destino no se encuentra entre estos vecinos recorre las listas de vecinos *2-hop* verificando si se encuentra entre estos, si es así el paquete se marca en el campo de Id_next_2h con el id del nodo que se encuentra a *2-hop*, en el campo de Id_next con el id del nodo mediante el cual se llega al vecino a *2-hop* y el campo de contador se marca con un uno. En el caso de que exista más de un vecino a un salto para llegar al nodo de dos saltos, se elige el vecino a un salto que tenga menor distancia o progreso al destino.

Si el destino no se encuentra entre los vecinos a un salto o entre los vecinos *2-hop*, el nodo recorre las listas de vecinos *2-hop* y calcula para el caso de 2-f-GEDIR cuál de todos ellos tiene una menor distancia euclidiana al destino y para el caso de 2-f-MFR un menor progreso al destino, las fórmulas empleadas para esto son Formula 1 y Formula 2 para cada caso. Al elegir el vecino de *2-hop* que cumpla con estas condiciones el paquete se marca en el campo de Id_next_2h con este id, en el campo de Id_next con el id del nodo mediante el cual se llega al vecino a 2 hop y el campo de contador se marca con un uno. En el caso de que exista más de un vecino a un salto para llegar al nodo de dos saltos, se elige el vecino a un salto que tenga menor distancia o progreso al destino.

2-f-GEDIR: Distancia euclidiana

$$distcal2h = \sqrt{(pos\_x_{sink} - pos\_x_{nodo})^2 + (pos\_y_{sink} - pos\_y_{nodo})^2} \quad (9)$$

$$donde \quad sink = sumidero \quad nodo = nodo\ que\ calcula\ actualmente$$

Formula 1 Distancia al destino del nodo que calcula

2-f-MFR: progreso

El progreso se define como la proyección del nodo N2 sobre la recta que conecta al nodo N3 con N1, figura Anexo 17, el cual se calcula por medio de la ecuación (15). La manera de calcularlo se explica en el Anexo 18.

$$progreso = d1 = \frac{\overline{N_3N_2} \cdot \overline{N_3N_1}}{\left\|\overline{N_3N_1}\right\|} = \frac{(pos\_x_{N_3N_2} \cdot pos\_x_{N_3N_1}) + (pos\_y_{N_3N_2} \cdot pos\_y_{N_3N_1})}{\sqrt{(pos\_x_{N_3} - pos\_x_{N_1})^2 + (pos\_y_{N_3} - pos\_y_{N_1})^2}} \quad (15)$$

Formula 2 Progreso del nodo N2 al nodo N3

4.2.1.3. Nodo cóncavo y *flooding* (inundación) restringido

Cuando un nodo recibe un paquete y elige al vecino *2-hop* con menor métrica al destino, también revisa el campo del paquete id_nodo_calc, el cual índica el id del nodo que anteriormente había efectuado los cálculos para elegir al vecino a *2-hop*, y comprueba si el id del nodo elegido a *2-hop* es igual al valor de este campo; en el caso que sean iguales significa que el nodo es un nodo cóncavo y que por lo tanto todos sus vecinos se encuentran más lejos que él mismo al destino. El nodo marca el paquete con su id en el campo de id_concavo (si no es nodo cóncavo su valor por defecto es -1), el campo de contador con 2, el campo de Id_next_2h con un valor no valido (-1) y el campo de Id_next con un -99 lo cual índica a todos los nodos a un salto que deben aceptar el paquete y hacer cálculos para decidir a qué vecino a *2-hop*

enviarlo. El nodo que recibe el mensaje toma el valor de id_concavo para eliminar de su lista de vecinos al nodo que se identifico como cóncavo, de esta manera los paquetes no regresarán a este nodo.

Al marcar el paquete con un -99 se está ejecutando un *flooding* (inundación) restringido, ya que el paquete se replica tantas veces como la cantidad de vecinos a un salto del nodo que se identificó como cóncavo. Cuando estos vecinos reciben el paquete marcado con este valor, evaluarán nuevamente que vecino a 2 saltos poseen con menor métrica al destino, es decir continuan ejecutando el funcionamiento normal del protocolo (2-f-GEDIR o 2-f-MFR, dado el caso), de esta manera el *flooding* solo llega a los nodos de primer salto y la red completa no se inunda el paquetes.

4.2.2.Desarrollo del protocolo GEAR

El modelo de nodo que se realiza para el protocolo GEAR es el que se índica en el Anexo 19.

4.2.2.1. Creación de listas de costo estimado

El protocolo GEAR tiene un módulo llamado "*proto_enrutamiento*" el cual hace uso de la información obtenida mediante el protocolo de reconocimiento, para conocer a sus vecinos. Su modelo de proceso es el que se índica en la Figura 23.

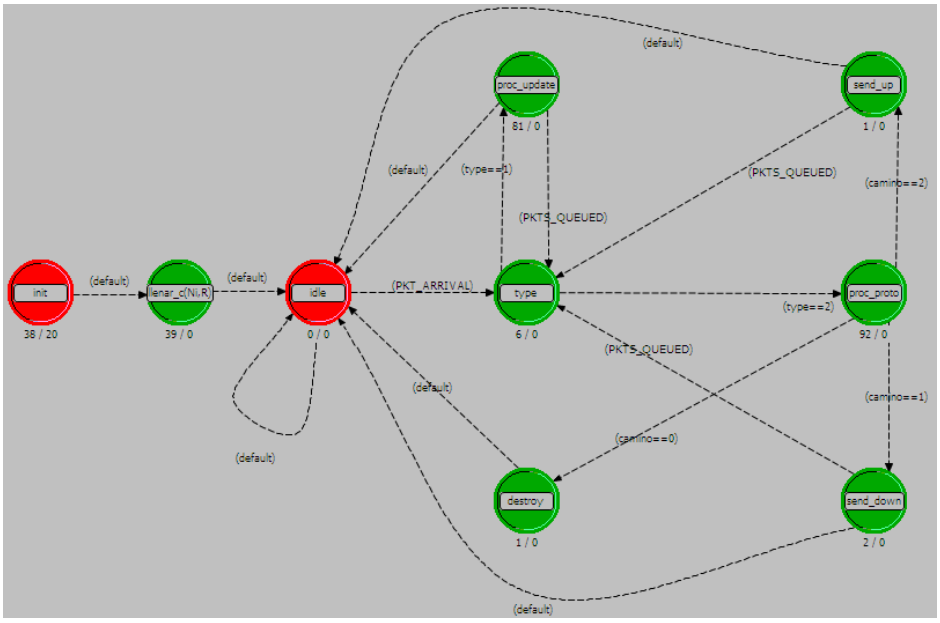


Figura 23 Modelo de proceso GEAR

Inicialmente en el estado *"llenar_c(Ni,R)"*, cada nodo calcula el costo estimado de cada uno de sus vecinos. Este costo estimado se calcula con (16).

(16)

Donde es la distancia euclidiana al destino, la cual se calcula haciendo uso de las coordenadas X y Y de cada vecino y es la energía restante de cada nodo (las coordenadas y la energía fueron obtenidas inicialmente mediante el protocolo de reconocimiento 4.1.4). En la bibliografía es normalizada por la mayor distancia de los vecinos al destino, y se normaliza por el mayor valor de energía consumida de los vecinos; en el caso de este proyecto por los valores de distancia tomados y el modelo de energía tomado, el protocolo tuvo un mejor comportamiento cuando no se normalizaron estos valores.

4.2.2.2. Desarrollo del protocolo y paquete de información

La fuente *"Aplicación_fuente"*, Anexo 6, genera el paquete que se utiliza para enrutar la información a través de los nodos de la red, este tiene la estructura que se muestra en la Figura 24 y se denomina *"pkt_GEAR"*.

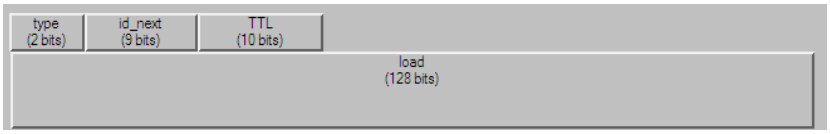


Figura 24 Paquete de información GEAR *"pkt_GEAR"*

Tabla 5 Campos del paquete *"pkt_GEAR"*

Campo	Función del Campo
Type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. En general los protocolos de enrutamiento manejan por defecto un type igual a dos.
id_next	id del nodo vecino al cual debe ser enviado el paquete.
load	contiene la información que generó el sensor
TTL	Valor usado para limitar el número de saltos que un paquete puede realizar en la red.

Inicialmente el nodo se encuentra en estado *"idle"*, en donde espera la llegada de paquetes; cuando un paquete *"pkt_GEAR"* llega al nodo, en el estado *"type"* se verifica si es un paquete de información o si es un paquete *"pkt_update"* (se explica en la sección 4.2.2.2.1) el cual informa a los vecinos del cambio energía o del cambio en el costo del nodo que generó el paquete.

Si es un paquete *"pkt_GEAR"*, se procesa en el estado *"proc_proto"*, aquí se recorre la lista vecinos y se elige al vecino con menor costo para reenviar el paquete, luego en el estado *"send_down"* es enviado al transmisor del nodo. En seguida del envió el nodo calcula nuevamente su costo propio como se índica en (17).

(17)

(18)

Donde es el costo aprendido propio del nodo, es el menor costo entre todos los vecinos y es el costo de transmitir al vecino con menor costo, en este caso el costo de

transmisión será igual a 4 veces la energía consumida al transmitir (este valor se ajusta en 4 debido a que mejoró el comportamiento del protocolo en el escenario de prueba).

Los demás nodos de la red no conocen el nuevo costo aprendido del nodo que ha hecho el cálculo, ya que ellos únicamente conocen el costo estimado de este (solo en los casos de nodo cóncavo los demás nodos conocen este costo aprendido).

4.2.2.2.1. Actualización del costo estimado y paquete de actualización

Debido a que existe un consumo de energía al recibir y enviar paquetes de información, los nodos de la red deben actualizar el costo estimado de sus vecinos, esto se realiza en el estado *"proc_proto"*; para lograr esto se utilizan paquetes de actualización (más pequeños que los paquetes de información, por lo que el consumo de energía que realizan no es relevante en el modelo de energía realizado) en los que se comunica el valor de energía restante, esto se realiza únicamente luego de que un nodo ha recibido y enviado un paquete. Los paquetes de actualización poseen el formato que se índica en la Figura 25 y su nombre es *"pkt_update"*.

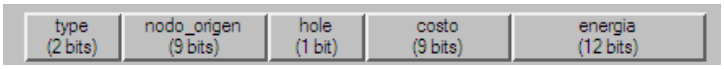


Figura 25 Paquete de actualización *"pkt_update"*

Tabla 6 Campos del paquete *"pkt_update"*

Campo	**Función del campo**
Type	Tipo de paquete llega al receptor, para ser asignado al módulo encargado de su respectivo procesamiento. En este caso el type igual a uno.
nodo_origen	id del nodo que está haciendo la actualización
hole	Valor que indica si es un paquete de actualización de costo aprendido o un paquete de actualización de energía.
costo	Costo aprendido que está propagando el nodo_origen, quien en este caso sería nodo cóncavo.
energia	Energía restante del nodo_origen

El nodo que transmitió el paquete de información *"pkt_GEAR"*, en el estado de *"proc_proto"*, genera un paquete de actualización *"pkt_update"* con el que informa su energía restante. En este marca el campo de nodo_origen con su id, el campo de hole con un cero y el campo de energía con el valor de la energía que posee, inmediatamente el paquete es enviado. Luego todos sus vecinos aceptan el paquete y lo procesan en el estado *"type"*, el cual se encarga de enviarlo al estado *"proc_update"* en donde se actualiza el valor de energía del nodo que generó el mensaje, para posteriormente actualizar el costo estimado de este.

De esta manera a medida que los nodos consumen energía su costo estimado aumenta, por lo que si un nodo entre todos sus vecinos no ha enviado paquetes tendrá un costo estimado menor a todos los demás y dado el caso será elegido para el reenvió creando un balance de energía en la red.

4.2.2.2.2. Caso nodo cóncavo

Un nodo es un nodo cóncavo cuando su costo aprendido al destino es menor que el menor costo estimado entre todos sus vecinos; en este punto debido a que la mejor opción de reenvió es él mismo, el nodo elige al vecino con menor costo estimado y cambia su valor propio de costo aprendido a un valor calculado mediante (19).

(19)

Donde es el costo aprendido propio del nodo, es el menor costo entre todos los vecinos, es el costo de transmitir al vecino con menor costo, el cual es igual a 4 veces la energía de transmisión, y es un valor mayor que se le asigna al costo para penalizar al nodo cóncavo.

Antes de enviar el paquete de información *"pkt_GEAR"*, el nodo envía el paquete de actualización *"pkt_update"* marcando su id en el campo nodo_origen, un uno en el campo de hole y el valor del costo aprendido en el campo de costo. De esta manera los nodos vecinos actualizan el costo del nodo que se identifica como cóncavo y al momento de reenviar el paquete de información *"pkt_GEAR"* no eligen a este nodo, ya que posee un costo mayor al de todos los demás. Se debe tener en cuenta que a medida que los nodos alrededor del nodo cóncavo aumentan su costo, harán que en un momento futuro el nodo cóncavo vuelva a ser elegible ya que su costo será equiparable al de los nodos a su alrededor.

4.2.3.Desarrollo del protocolo GAF

El modelo de nodo que se desarrolló para GAF es el que se muestra en el Anexo 20.

4.2.3.1. Identificación de grillas virtuales

El módulo que describe el comportamiento del protocolo GAF se denomina "*gaf*" y su máquina de estados es la que se muestra en la Figura 26.

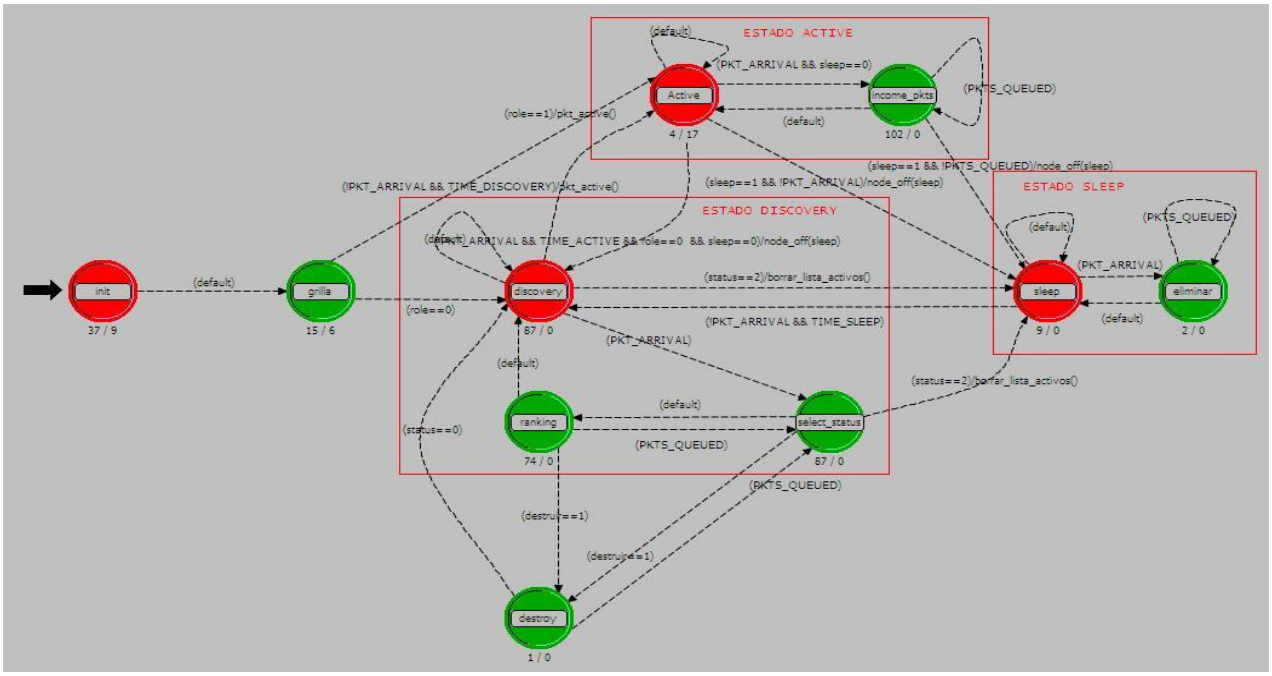


Figura 26 Modelo de proceso de GAF

En el estado *"grilla"* cada nodo identifica la grilla virtual a la que pertenece, para esto inicialmente se debe conocer la longitud del lado de cada grilla, la cual es igual a (20)

= (20)

En donde R es el radio de cobertura que tiene cada nodo, para el efecto de este proyecto este valor será igual a 100mts. Debido a que los escenarios que se generan inician en el punto de coordenadas (0,0) los cálculos que se realizan para identificar el id de la grilla a la que pertenece el nodo, consisten en dividir la coordenada X del nodo sobre la longitud del lado de la grilla, es decir ——, debido a que el valor obtenido no es un valor entero este debe aproximarse por encima al valor entero correspondiente, por lo que si se obtiene es un 0.2 el valor al que se aproxima es a 1, si se obtiene 1.4 el valor al que se aproxima es a 2 y así en adelante; para el caso de la coordenada Y se efectúa el mismo calculo. Para obtener un valor valido para el id de , la grilla se unifican y de la siguiente manera, , se elige un valor de 100 debido a que para el tamaño de escenario elegido, cuyo valor es de 1km por 1km, el máximo número de grillas en la coordenada X son 23, por lo que el id de la ultima grilla será igual a 2323 y el de la primera 101.

4.2.3.2. Desarrollo del Protocolo

4.2.3.2.1. Estado "Discovery"

El siguiente estado en el que se encuentra el nodo luego de esperar la llegada de paquetes es *"Discovery"*, Figura 26. En este estado el nodo genera aleatoriamente un tiempo *td*, el cual índica el tiempo que se encontrará en estado "*discovery"*, el valor de *td* se encontrará entre 0 y enlt/8 ; luego crea un paquete de

descubrimiento mediante el cual informa a todos sus vecinos, su id, el id de la grilla en la que se encuentra y el tiempo que se encontrará en estado *"active"*; este paquete se denomina *"pkt_discovery"* y su formato es el que se muestra en la Figura 27.

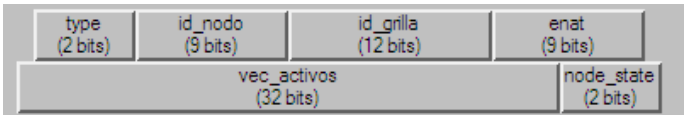


Figura 27 Paquete de descubrimiento *"pkt_discovery"*

Tabla 7 Campos del paquete *"pkt_discovery"*

Campo	**Función del campo**
Type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. En este caso el type igual a uno.
id_nodo	Id del nodo que está generando el paquete
id_grilla	Id de la grilla a la que pertenece el nodo que genera el paquete
enat	Tiempo que el nodo de id_nodo estaría en estado activo
node_state	Estado en el que se encuentra el nodo de id_nodo. Los valores que puede tomar este campo son 0 para *discovery*, 1 para *active*, 2 para *sleep*.
vec_activos	Lista de los vecinos activos del nodo

Los nodos reciben este paquete y en el estado de *"select_satus"* analizan si la comunicación recibida corresponde a un nodo que esta advirtiendo que entró en estado activo (node_state igual a uno), si es así se analiza el campo del paquete que índica el id de la grilla a la que corresponde este nodo, id_grilla, de esta manera se sabe si el nodo que está advirtiendo que es activo es de la grilla a la que pertenece el nodo que está recibiendo el mensaje. Cuando esto sucede el nodo pasa a estado *"sleep"* y programa un temporizador menor al valor de enat indicado en el paquete, de tal manera que cuando el vecino que se ha activado cambia su estado a *"discovery"* o a "*sleep"*, el nodo que estaba en *"sleep"* cambia a estado *"discovery"* un tiempo antes de que esto ocurra.

Si la información que se ha recibido índica que un nodo se ha activado, pero que no se encuentra en la misma grilla del nodo que recibe el mensaje, esto le índica al nodo que hay alguien que está activo a su alrededor y que se encuentra dentro de su radio de cobertura R, por lo que esta información será útil al momento de pasar a estado *"active"* y enrutar paquetes.

En el caso de que el paquete recibido sea solo un paquete de descubrimiento el cual índica que un nodo vecino se encuentra en estado *"discovery"* (node_state igual a cero), el nodo utilizará esta información en el estado de *"ranking"*, en donde se creará un listado de los vecinos de la misma grilla y del tiempo esperado de vida (enlt) que estos poseen. El valor de enlt se conoce debido a que es igual a enat por dos. Esta información se procesa cuando el temporizador *td* se vence, debido a que el nodo deberá pasar a estado *"active"*, pero dado el caso de encontrar a un vecino con un enlt mayor al suyo, el temporizador *td* se reprogramará dándole la oportunidad a los nodos que tienen mayor enlt de entrar en estado activo.

4.2.3.2.2. Estado "Active"

Cuando el temporizador *td* del nodo se vence en el estado de *"discovery"* y no existe un vecino que tenga un entl mayor, el nodo cambia a estado a *"active"* y envía un paquete *"pkt_discovery"*, marcando el campo de node_sate con un uno y el resto de la información con sus datos propios; de esta manera avisa a sus vecinos de grilla que él es el nodo que se encargará del enrutamiento en esa grilla, con este mensaje los demás nodos pasan a estado *"sleep"*. El tiempo *ta* que el nodo se encuentra en estado *"active"* es igual a enlt/2, esto es igual para todos los nodos de la red.

En el caso de que la energía del nodo se agote antes de que el tiempo *ta* termine, el nodo pasará a estado *sleep*. En este caso se puede presentar perdida de paquetes debido a que si los demás nodos de la grilla están en estado "*discovery"* y el nodo que estaba activo pasa a *sleep*, la grilla quedará sin un nodo que se encargue del enrutamiento paquetes hasta que algún otro nodo se active.

En este estado todos los paquetes de descubrimiento de otras grillas que lleguen marcados con node_status igual a uno, es decir nodos que están anunciando que han pasado a "*active"*, serán aceptados en el proceso *"income_pkts"*, en donde se llena la lista de vecinos activos que tiene el nodo. Los paquetes que estén marcados con node_status igual a cero son eliminados.

4.2.3.2.3. Estado "Sleep"

Cuando un nodo pasa a "*sleep"* significa que alguien en su grilla se ha activado o que su energía se ha agotado. El nodo se encuentra en este estado durante un tiempo *ts*, el cual se calcula obteniendo del paquete *"pkt_discovery"* el tiempo enat del nodo que comunico que se había activado, por lo que *ts* se programa en un valor menor a enat, de tal manera que el nodo entre antes a *"discovery"*, que el nodo que estaba en *"active"*. Cuando un nodo se encuentra en *"sleep"*, su radio receptor se encuentra apagado, por lo que cuando llegan paquetes estos se eliminan en el estado *"eliminar"*.

4.2.3.2.4. Acople de GAF al protocolo de enrutamiento y Paquete de información

El protocolo de enrutamiento elegido para probar el funcionamiento de GAF es *Greedy* básico, en donde la elección del vecino del siguiente salto se hace en base a una métrica de distancia, es decir se elige el nodo que tenga menor distancia euclidiana al destino.

El acople de GAF al protocolo de enrutamiento *Greedy* se efectúa cuando el nodo pasa a *"active"*. Para que siempre existan paquetes en la red, el nodo fuente se considera como un nodo que siempre se encuentra en estado *"active"* y cuya energía es inagotable, esto mismo se aplica para el nodo sumidero. El módulo creado para el desarrollo del protocolo se denomina "*protocolo_enrutamiento*" y como su modelo de proceso es Anexo 21.

El protocolo *"gaf"* se encuentra ligado a este proceso. Cuando el nodo se encuentra en "*active"* todos los paquetes de información "*pkt_GAF*" se reciben en el módulo de "*gaf*" y este los entrega al módulo "*protocolo_enrutamiento*", en donde son procesados por aquí se ejecuta el procesamiento del paquete y se envía al nodo que cumpla con la característica de menor métrica al destino.

Si un nodo que se encuentra estado *"active"* en una grilla, no posee vecinos activos a su alrededor con menor métrica al destino que la de él mismo, a quienes pueda reenviar los paquetes, esperará 60 segundos a que alguien se active para realizar el envío, si luego de ese tiempo ningún nodo con menor métrica se ha activado, el nodo elimina los paquetes que tenga encolados. Si su tiempo *ta* se vence antes de que los paquetes puedan ser reenviados, todos los paquetes que poseía el nodo en ese momento serán desechados.

4.2.4.Desarrollo de GDSTR

El modelo de nodo que se realiza para el protocolo GEAR es el que se índica en el Anexo 22.

4.2.4.1. Módulo *"Spanning_tree"*

El protocolo GDSTR tiene un módulo llamado "*spanning_tree*" el cual es el encargado de crear, definir y mantener el árbol a ser utilizado por el protocolo de enrutamiento para el re-envio de paquetes. Su modelo de proceso es el que se indica en la Figura 28.

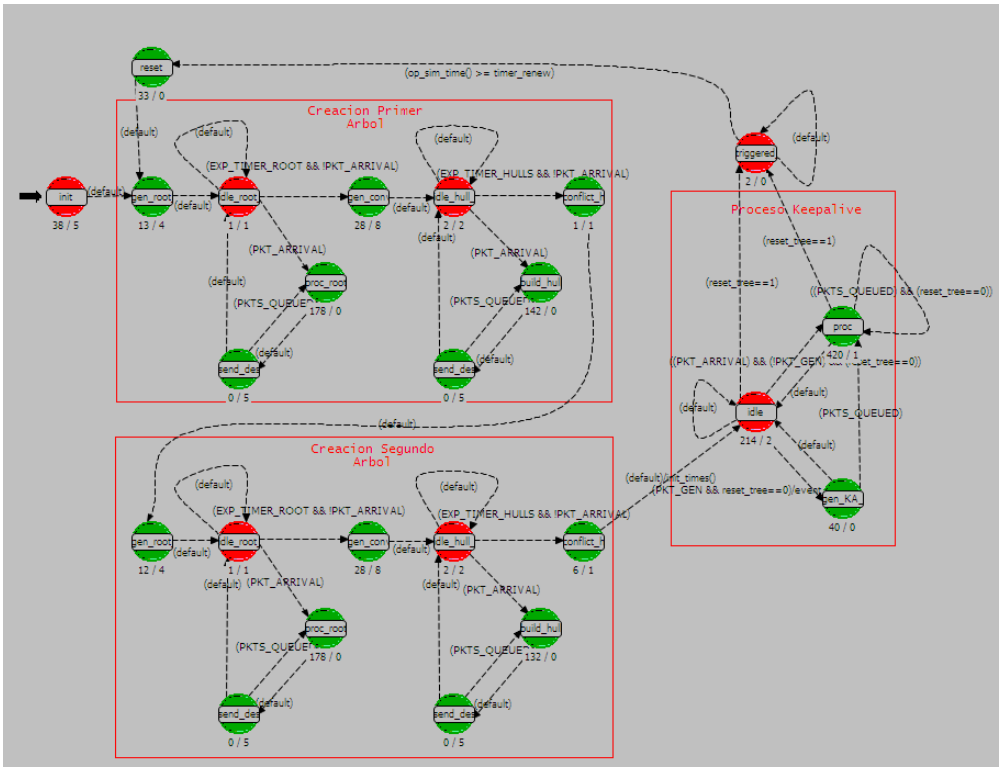


Figura 28 Modelo de proceso *"spanning_tree"*

Para la creación completa y el mantenimiento del árbol de la red conectada se implementaron tres procesos los cuales son: elección del nodo raíz, definición del *convex hull* propio y mantenimiento del árbol.

4.2.4.1.1. Proceso de selección del nodo raíz

El primer proceso en la creación de un árbol es el de escoger el nodo raíz entre todos los nodos que tengan conectividad en la red; este proceso lo realiza por medio de un paquete denominado "*pkt_root_selec*" con el siguiente formato, Figura 29:

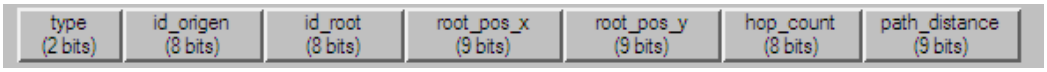


Figura 29 Paquete *"pkt_root_selec"*

Tabla 8 Campos del paquete *"pkt_root_select"*

Campo	Función del Campo
type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. En este caso el type es igual a uno.
id_origen	id del nodo vecino al cual debe ser enviado el paquete.
id_root	Id del nodo raíz que posee el nodo que generó el paquete.
root_pos_x	Coordenada X del nodo raíz que tiene el nodo que generó el paquete.
root_pos_y	Coordenada Y del nodo raíz que tiene el nodo que generó el paquete.
hop_count	Numero de saltos del nodo que generó el paquete al nodo raíz.
path_distance	Distancia en camino del nodo que generó el paquete al nodo raíz.

Para la elección del nodo raíz, inicialmente cada nodo se declara como nodo raíz marcando con su propio id y su posición los campos del paquete; hop_count y path_distance los asigna en cero y envía el paquete a todos sus vecinos.

Cuando un nodo recibe este paquete, comprueba si el nodo raíz que le es comunicado es el mismo que tiene. Si es el mismo almacena esta información en una lista de posibles padres revisando que no se encuentre ya almacenado y si la distancia en camino es menor cambia su nodo padre; finalmente elimina el paquete.

Cuando el nodo raíz no es el mismo que tiene, pasa a comprobar si la posición X es menor que la del nodo raíz que tiene este nodo. Si es menor, el nodo raíz cambia por el nodo del campo id_root, pone como nodo padre al nodo indicado en el campo id_origen, aumenta el número de saltos en uno y calcula la distancia del nodo que procesa el paquete al nodo que lo originó. Seguidamente cambia los campos del paquete con los nuevos valores y lo reenvía. Si es la posición X es mayor, descarta el paquete. En el caso de empates, comprueba también la posición Y.

De esta manera, por medio de la convergencia de información del nodo raíz en forma descendente en el árbol todos los nodos tendrán al mismo nodo raíz con la información respectiva a este y al nodo padre asociado al árbol creado. El nodo está en espera por esta convergencia un tiempo determinado.

4.2.4.1.2. Definición del *convex hull* propio

Para lograr que cada nodo padre de una rama conozca a sus hijos y contenga en su *convex hull* los *convex hull* de estos, todos los nodos hacen uso de un paquete denominado "*pkt_hull_trees*" con el siguiente formato, Figura 30:

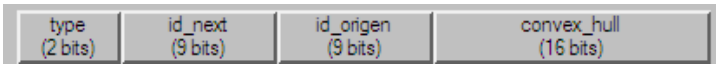


Figura 30 Paquete *"pkt_hull_trees"*

Tabla 9 Campos del paquete *"pkt_hull_trees"*

Campo	Función del Campo
type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. En este caso el type es igual a uno.
id_next	Id del nodo al que es dirigido el paquete.
id_origen	Id del nodo que generó el paquete.
convex_hull	Lista que contiene los vértices respectivos al *convex hull* del nodo que originó el paquete.

Para este proceso cada nodo parte definiendo el *convex hull* propio como los 4 vértices de un cuadrado cuyo centro es el nodo, y la longitud de sus lados son el doble de su cobertura como se muestra en Figura 31. Al tener ya definido el *convex hull* propio, el nodo lo comunica a su nodo padre marcando el campo *convex_hull* del paquete.

Cuando un nodo recibe este tipo de paquete y se encuentra dirigido a él, en una lista de hijos almacena el id y el *convex hull* del nodo respectivo y re-calcula su *convex hull* propio. Este re-calculo lo hace extendiendo su *convex hull* hasta las máximas y mínimas coordenadas de los vértices del *convex hull* que contiene el paquete, esto se puede observar en Figura 32. Si el nodo es diferente al nodo raíz, cambia los campos del paquete y lo re-envía a su nodo padre.

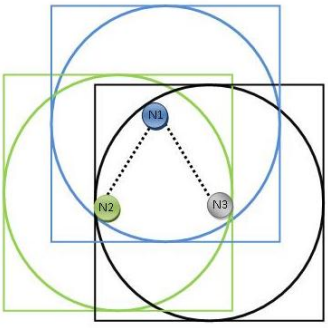


Figura 31 Definición *convex hull* propio

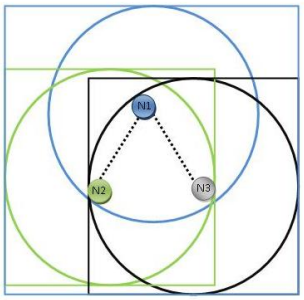


Figura 32 Convergencia de los *convex hulls*

Así, de igual manera que en el proceso de selección del nodo raíz por medio de la convergencia de información del nodo raíz en forma ascendente en el árbol todo los nodos conocerán a sus hijos asociados

al árbol creado y tendrán *convex hull*s que contengan a los de todos ellos. El nodo está en espera por esta convergencia un tiempo determinado.

4.2.4.1.3. Proceso de mantenimiento del árbol

El proceso de un nodo para mantener el árbol y detectar fallas de nodos y cambios de topología lo realiza por medio de un tipo de paquete periódico llamado "*pkt_keepalive*" definido de la siguiente manera, Figura 33 :

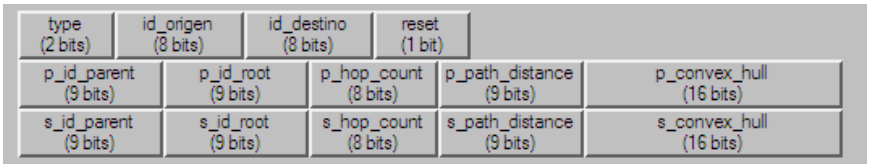


Figura 33 Paquete *"pkt_keepalive"*

Tabla 10 Campos del paquete *"pkt_keepalive"*

Campo	Función del Campo
type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. En este caso el type es igual a uno.
id_destino	Id del nodo al que es dirigido el paquete.
id_origen	Id del nodo que generó el paquete.
Reset	Valor usado para reiniciar los cálculos de todo el árbol o para identificar los cambios de topología de cada arbol.
p_id_origen	Id del nodo padre del nodo que generó el paquete.
p_id_root	Id del nodo raiz del nodo que generó el paquete.
p_hop_count	Numero de saltos que se encuentra el nodo que generó el paquete al nodo raiz.
p_path_distance	Distancia en camino que se encuentra el nodo que generó el paquete al nodo raiz.
p_convex_hull	Lista que contiene los vértices respectivos al *convex hull* del nodo que originó el paquete.

Un nodo se da cuenta del fallo de un vecino cuando no recibe un paquete *pkt_keepalive* durante 90 segundos (3 veces el tiempo de generación del paquete). Cuando esto sucede, el nodo identifica si el vecino que ha fallado es el nodo padre, algún nodo hijo o el nodo raíz. Si es el nodo padre, pasa a seleccionar como nuevo padre el nodo que tenga la menor distancia en camino de su lista de posibles padres, lo elimina de la lista y lo comunica directamente que este nodo es un nuevo hijo mediante un *keepalive unicast*. Si este nodo al recibir esta actualización, se percata que el nodo que le comunica que es un nuevo hijo de él es el nodo que tenía como padre, pasará de igual manera a elegir un nuevo padre entre su lista de posibles padres. Si la lista de posibles vecinos se encuentra vacía, esa rama ha perdido conectividad.

Si el nodo que falló es un nodo hijo reduce y actualiza su *convex hull*. En el caso de que el nodo que falló sea el nodo raíz, el nodo marca el paquete en el campo de "*reset*" con un 99. Esto lo realiza de igual manera para ambos arboles con la diferencia de que para el primer árbol, marca el campo de "*reset*" con un 1 y para el segundo con un 2.

4.2.4.2. Módulo "*protocolo_enrutamiento*"

La fuente *"Aplicación_fuente"*, Anexo 22, genera el paquete que se utiliza para enrutar la información a través de los nodos de la red, este tiene la estructura que se muestra en la Figura 34 Paquete *"pkt_proto_gdstp"* y se denomina *"pkt_proto_gdstp"*.

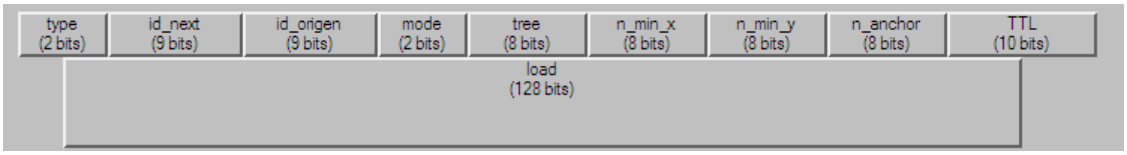


Figura 34 Paquete *"pkt_proto_gdstp"*

Tabla 11 Campos del paquete *"pkt_proto_gdstp"*

Campo	Función del Campo
type	Tipo de paquete que llega al receptor, indicando a que proceso pertenece. En este caso el type es igual a dos.
id_next	Id del nodo al que es dirigido el paquete.
id_origen	Id del nodo que generó el paquete.
mode	Valor usado indicar si el paquete se encuentra enrutando en *Greedy* o en spanning tree.
tree	Id del nodo raíz del árbol en el cual se encuentra enrutando el paquete.
n_min_x	Posicion X del nodo cóncavo.
n_min_y	Posicion Y del nodo cóncavo.
n_anchor	Id del nodo que empieza a recorrer las ramas respectivas de sus hijos.
TTL	Valor usado para limitar el número de saltos que un paquete puede realizar en la red.

El módulo llamado "*protocolo_enrutamiento*" es el encargado de realizar el enrutamiento de manera *Greedy* o de manera *spanning tree*. Su modelo de proceso es el que se indica en la Figura 35.

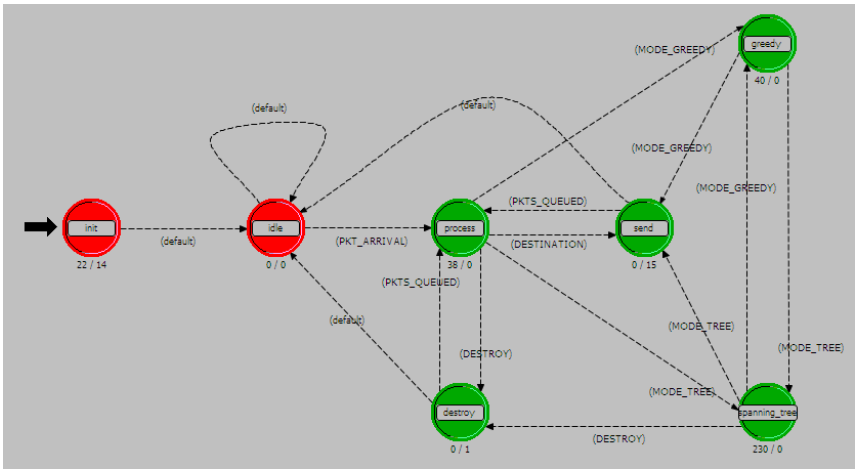


Figura 35 Modelo de proceso de *"protocolo_enrutamiento"*

Inicialmente el nodo se encuentra en el estado de *"idle"*, cuando un paquete de información *"pkt_proto_gdstp"* llega al nodo, este lo procesa en el estado *"process"* donde analiza si él es el destino de paquetes de la red, si es así los paquetes son dirigidos al estado *"send_up"* el cual los envía al módulo de capa superior *"aplicación_sumidero"*, aquí se considera que la vida de ese paquete en la red ha terminado. En el caso contrario, en el que el nodo que está procesando el paquete no es el sumidero, el nodo analiza si el paquete que llega es para él es decir si el campo de id_next está marcado con su id

propio, si es así el nodo acepta el paquete y lo manda al estado respectivo en el que venga el paquete ya se *Greedy* o *spanning tree*. En el estado *Greedy*, el nodo procede a seleccionar de manera *Greedy* el vecino para re-enviar el paquete, acá comprueba la condición de nodo cóncavo. Si es nodo cóncavo, el enrutamiento cambia y pasa al estado *spanning tree* donde el nodo procede inicialmente a revisar si el nodo destino se encuentra dentro de su *convex hull*. Si no se encuentra el nodo re-envía el paquete a su nodo padre. En caso contrario, el nodo identifica las ramas sobre las que se encuentra contenido el nodo destino y por medio del *anchor* empieza a recorrerlas para encontrar el destino.

5. Análisis de Resultados

En esta sección se muestran los resultados obtenidos en los escenarios determinísticos y los resultados de los 150 escenarios aleatorios simulados. Los parámetros que se evalúan fueron explicados en la sección 3.4.

5.1. Escenarios determinísticos

El escenario determinístico que se creó para evaluar los protocolos 2-f-GEDIR, 2-f-MFR, GEAR y GDSTR, se muestra en el Anexo 23.

Se efectúa una simulación en la que la fuente genera paquetes cada diez segundos, donde el tiempo de simulación es de una hora y media y el total de paquetes a enviar es de 500. Luego de correr cada protocolo se obtienen los siguientes resultados.

5.1.1. 2-f-GEDIR

Debido a que este protocolo elige a los nodos encargados de reenviar paquetes mediante una métrica de distancia euclidiana, los primeros nodos que agotan su energía son aquellos que se encuentran sobre la línea imaginaria que va del nodo fuente al nodo sumidero. Esto se puede comprobar en el Anexo 24. En un punto de la simulación el nodo fuente pierde conectividad con el resto de la red, lo que significa que no llegarán más paquetes al destino, esto puede observarse en la gráfica del Anexo 25 (paquetes recibidos y paquetes generados), en donde la gráfica azul representa los paquetes entregados y la gráfica roja los paquetes generados. De esta figura se puede notar que aproximadamente luego de una hora y diez minutos la red pierde conectividad. Esto también puede comprobarse analizando la cantidad de nodos activos que posee la red antes de perder su conectividad, esto puede observarse en el Anexo 26, en donde se observa que aproximadamente en el tiempo una hora y diez minutos, muere un nodo, este nodo representa la última ruta que da conectividad de la red.

Luego de que se pierde la conectividad el nodo fuente se identifica como cóncavo ya que entre todos sus vecinos, él es el único que posee energía y por lo tanto es el único cercano al destino. Esto puede evidenciarse en el Anexo 27, en donde luego de una hora y diez minutos se evidencia la existencia de un nodo cóncavo.

A medida que los nodos empiezan a agotar su energía y por lo tanto a dejar de existir, los caminos al destino empiezan a cambiar, por lo que el número de nodos saltos (*hop count*) hacia el destino empieza a variar. Esto se ve en la Anexo 28, en donde el número inicial de saltos es 10, y luego varía entre 12 y 14, hasta que se pierde la conectividad de la red. Esto se debe a que los paquetes entraron a un nodo cóncavo o que algún nodo agotó su energía y una nueva ruta tuvo que ser hallada.

Así mismo se obtiene el promedio del porcentaje de utilización de los nodos en la Anexo 30, donde se observa que inicialmente pocos nodos tienen un porcentaje de utilización de aproximadamente un 98%, estos nodos corresponden a los nodos de la diagonal, y cuando estos nodos agotan su energía, el porcentaje de utilización de los nodos que son usados para encontrar la nueva ruta o que no son usados durante toda la simulación, varía entre el 0% y el 20%, antes de que se pierda la conectividad.

Para evaluar de manera global el porcentaje de nodos que fueron usados por el enrutamiento, se tiene la gráfica que muestra el uso de la red, Anexo 33, donde se ve que al final de la simulación se hace uso de aproximadamente el 78% de los nodos de la red. Como se puede observar en la gráfica aproximadamente a los 35 minutos de la simulación se pasa de un 17% a un 41% de la utilización de la red lo que indica que se hizo un *flooding* del paquete, y que por lo tanto varios nodos fueron usados para procesarlo.

El parámetro de energía permite evaluar de que manera los nodos de la red consumen energía cuando reciben y transmiten paquetes, o cuando están escuchando el medio. En el Anexo 31 se muestra como cambia la energía de cada nodo de la red. Inicialmente los nodos tienen 1000 unidades de energía, cuando empiezan a transmitir y a reenviar paquetes agotan más rápidamente su energía, en este caso los nodos de la diagonal son aquellos que consumen más rápidamente su energía, en el grafico se ven representados por la línea de color amarillo que se encuentra a la izquierda de todas las demás gráficas. Luego cuando es necesario hallar nuevas rutas para lograr entregar los paquetes, nuevos nodos empiezan a consumir energía, lo que se observa como las ramas que se ven en el grafico.

Otro parámetro es el *end to end delay*, el cual indica el tiempo que los paquetes tardan en llegar del nodo fuente al nodo destino. En este caso se tiene la gráfica del Anexo 29, en donde se puede observar que aproximadamente 270 paquetes tardan en llegar al destino segundos y aproximadamente 70 paquetes tardan en llegar segundos. Esta diferencia se debe a que el número de saltos aumentó por lo que los paquetes tardaron un mayor tiempo en llegar al destino.

Las variaciones del *end to end delay*, se observan en el *jitter* o también llamado IPDV (*Instantaneous Packet Delay Variation*), el cual, como se explicó en la sección 3.4., representa la variación del retardo que tuvo un paquete comparada con la variación del paquete inmediatamente anterior. El Anexo 32 muestra que existe una variación en el retardo de los paquetes cuando cambia el número de saltos al destino, es decir cuando un paquete entra en un nodo cóncavo, o cuando debe encontrar una nueva ruta debido a la muerte de un nodo, los paquetes varían su tiempo de retardo. En la gráfica se observa que aproximadamente a los 47 minutos de la simulación, el IPDV cambia, esto corresponde al aumento de saltos que se observa en el mismo tiempo para la grafica del Anexo 28. Como se puede ver para el escenario determinístico el IPDV no varía constantemente durante la simulación, lo que significa que 2-f-GEDIR, busca elegir rutas cuyo número de saltos al destino sea mínimo, lo que produce que los paquetes sean entregados sin un mayor retardo entre ellos.

5.1.2. 2-f-MFR

Luego de ejecutar la simulación los nodos que permanecen con energía son los que se muestran en el Anexo 34; como se puede notar el comportamiento de 2-f-MFR es similar al de 2-f-GEDIR, debido a que el progreso hacia el destino también tiene en cuenta a los nodos que se encuentran sobre la diagonal. Inicialmente se agota la energía de los nodos que se encuentran sobre la diagonal imaginaria que hay entre el nodo fuente y el nodo destino; luego los nodos que agotan su energía son aquellos que se encuentran sobre las proyecciones de los nodos que se eligen para reenviar los paquetes al nodo destino. Por la figura del Anexo 36, se nota que 2-f-GEDIR puede llegar a elegir nodos más cercanos al destino.

De igual forma los nodos que permiten la conectividad mueren en tiempos similares a los de 2-f-GEDIR, como se puede observar por la gráfica del Anexo 35 aproximadamente a la hora y diez minutos de simulación se dejan de recibir paquetes. En esta gráfica también se puede notar que hubo una leve pérdida de paquetes al estar las dos graficas apartadas después de un determinado tiempo.

Aproximadamente a la hora y diez minutos de simulación, se pierde la conectividad, ya que el nodo que permitía que la recepción de paquetes, ha muerto. Para notar en qué punto se pierde la conectividad de la red, en la gráfica del Anexo 37 junto con la gráfica de paquetes entregados, se observa el momento en que un nodo de la última ruta disponible muere.

El *end to end delay* varia debido a las razones citadas anteriormente, esto puede observarse en la gráfica del Anexo 40, en donde el máximo valor que toma este parámetro es segundos a diferencia del valor máximo de 2-f-GEDIR que es segundos. Esta diferencia se da debido a que, existe un paquete cuya suma de saltos al destino es de aproximadamente 64 saltos, como se observa en la gráfica de *hop count* del Anexo 39; esto sucede porque el paquete es un paquete de *flooding* que entra 13 veces en nodos cóncavos como se muestra en el Anexo 41. De no ser por el paquete que entra en nodo cóncavo, el

número de saltos iniciaría en 10 y luego variaría entre 12 y 14 saltos de la misma manera que lo hace 2-f-GEDIR.

Este fenómeno se puede ver reflejado en el IPDV, Anexo 38, en donde el retardo entre paquetes es mayor (valor máximo de 2-f-GEDIR 0,00000065 segundos, valor máximo de 2-f-MFR 0,000017 segundos) debido al paquete que realiza 64 saltos al destino y que también viaja 13 veces por nodos cóncavos.

Finalmente el porcentaje de utilización de los nodos y el consumo de energía, son similares al de 2-f-GEDIR, debido a que como se dijo anteriormente los nodos que son más usados son los de la diagonal y muchos nodos se usan en un pequeño porcentaje o no son usados en absoluto para la transmisión de paquetes. Así mismo el porcentaje de utilización de la red es mayor, Anexo 44, en donde se observa que es aproximadamente un 80% debido al paquete que entra en nodos cóncavos.

5.1.3. GEAR

Luego de ejecutar la simulación los nodos que permanecen encendidos son los que se muestran en el Anexo 45. El balance de energía que efectúa GEAR, hace que los nodos alrededor de la fuente sean los primeros en agotar su energía, por lo que el nodo fuente se queda sin conectividad así los demás nodos de la red aún posean energía para transmitir paquetes. Otro efecto adverso que tiene este fenómeno es que se afecta la tasa de entrega de paquetes rápidamente, como se muestra en la gráfica de paquetes generados y paquetes recibidos en el Anexo 46. Debido a esto GEAR posee una alta tasa de entrega si el nodo fuente tiene un gran número de vecinos a su alrededor.

En la gráfica de nodos activos en el Anexo 47, se observa que aproximadamente a los 54 minutos de simulación han muerto los 4 nodos que permitían la conectividad de la fuente, por esta razón en la gráfica del Anexo 46 los paquetes dejan de ser recibidos en ese mismo momento. De igual manera al observar la gráfica de nodo cóncavo, Anexo 51, se puede notar que en el mismo tiempo en el que se pierde la conectividad un paquete entró en un nodo cóncavo. Debido a la definición de nodo cóncavo de GEAR, la probabilidad de que un nodo se declare como cóncavo es alta, como se ve en el Anexo 51.

El balance de energía que efectúa GEAR, hace que la mayoría de los paquetes elijan rutas distintas para llegar al nodo destino, por lo que el número de saltos y el *end to end delay* cambia para cada paquete (ver Anexo 48, Anexo 49). Por la misma razón la gráfica del IPDV varía durante toda la simulación, como se muestra en el Anexo 50.

En las gráficas de energía y porcentaje promedio de utilización de los nodos (Anexo 52 y Anexo 53) es aún más evidente el balance que hace GEAR, esto debido a que todos los nodos tienden a consumir la misma cantidad de energía y a ser utilizados en el mismo porcentaje. Este efecto es positivo desde el punto de vista de la red en su totalidad, ya que si la fuente no perdiera su conectividad, la vida de la red se

alargaría debido que no se consumiría toda la energía de un nodo, si no que la mayoría de los nodos tendrían energía suficiente para realizar el enrutamiento si pueden ser utilizados por rutas auxiliares para salir de nodos cóncavos.

Finalmente la gráfica del Anexo 54 indica que GEAR utiliza a todos los nodos de la red para mantener un balance de energía, como se puede observar el porcentaje de utilización es de aproximadamente el 98%.

Es importante notar que el balance que se observa se debe al que fue elegido, ya que si este valor es igual a 1 como se indica en la formula (2) de la sección 2.5.3, no se le estaría dando peso a la energía que poseen los nodos y por lo tanto el enrutamiento seria *Greedy* básico; es decir el costo que posee cada vecino se basaría únicamente en distancia. El elegido es igual 0,5, debido a que con este valor se busca dar igual peso a la distancia y a la energía, aunque cabe notar que la distancia máxima de un nodo al destino es 1km y que la energía máxima consumida es 1000, por lo que el máximo costo que puede tener un nodo antes de agotar su energía, es 500.5, en consecuencia la energía siempre tendrá más peso sobre el valor del costo estimado.

5.1.4.GDSTR

Luego de ejecutar la simulación los nodos que permanecen activos son aquellos que se muestran en la figura del Anexo 55. Debido a que GDSTR es un protocolo que la mayor parte del tiempo de enrutamiento hace uso del protocolo de enrutamiento *Greedy* para enviar los paquetes al destino (cuando no existen nodos cóncavos en la red), es de esperarse que al igual que los protocolos mencionados anteriormente, haga mayor uso de los nodos de la diagonal para enrutar los paquetes. Debido al funcionamiento de GDSTR, cuando un nodo consume toda su energía, deja de enviar mensajes de *keep alive* a sus vecinos, estos a su vez se enterarán de la muerte del nodo 90 segundos después de no recibir mensajes de *keep alive* (sección 4.2.4.1.3); debido a esto la tasa de paquetes se ve alterada, ya que un nodo seguirá enviando paquetes por un máximo de 90 segundos a un nodo que puede que ya haya agotado su energía. Esto se evidencia en la gráfica de paquetes generados y paquetes recibidos, Anexo 56, en donde aproximadamente a los 35 minutos de simulación, el nodo sumidero deja de recibir paquetes, debido a que un nodo ha muerto en ese instante como puede observarse por la gráfica de nodos activos del Anexo 57. Esto no significa que se haya entrado en un nodo cóncavo, lo que realmente representa es que un nodo transmite paquetes a un vecino que conoce, pero del cual no sabe que ya ha muerto.

La mejor manera de identificar la razón por la cual se altera la tasa de entrega es analizando la gráfica que indica si un paquete entró a un nodo cóncavo, Anexo 62. Como se puede observar de esta gráfica un paquete entra en un nodo cóncavo aproximadamente a los 48 minutos de simulación, lo que altera la gráfica de paquetes recibidos en el mismo instante, Anexo 56. En ese momento el protocolo conmuta de *Greedy* a GDSTR y busca una nueva ruta al destino, en este caso esa ruta implica efectuar dos saltos más,

como se ve en la gráfica de *hop count* del Anexo 60. En esta gráfica también se puede observar la falta de información que se produce cuando un nodo muere y sus vecinos tardan un tiempo en enterarse de esto; los vacios que se ven aproximadamente a los 35 minutos y a los 55 minutos reflejan este comportamiento. Como se define en la Seccion 4.2.4.2 cuando se ha superado el obstáculo impuesto por el nodo cóncavo, el protocolo retorna a *Greedy* y continúa entregando paquetes.

La red pierde conectividad aproximadamente en una hora y 15 minutos, esto se ve en la grafica de nodos activos Anexo 57 y en la grafica de paquetes recibidos Anexo 56.

El *end to end delay* se ve afectado cuando se presenta un cambio de ruta, es decir cuando se llega un nodo cóncavo, debido a que al conmutar de *greedy a* GDSRT, los paquetes demoran mas en llegar al destino. Esto puede comprobarse en la grafica del Anexo 61 (*End to end delay*) y en la grafica de IPDV en donde el tiempo de simulación en el que se presenta el cambio es el mismo que el tiempo en el que se encuentran los nodos cóncavos Anexo 62.

El porcentaje promedio de utilización de los nodos, es similar al de los protocolos analizados en las secciones anteriores, esto se debe al uso de *Greedy* para enrutar la información en un escenario libre de huecos. Por esta razón al observar la grafica de energía de los nodos, Anexo 63, se puede notar que inicialmente los nodos de la red consumen la misma cantidad de energía, cuando empiezan a morir, y a aparecer nodos cóncavos, nuevos nodos deben ser usados para el enrutamiento, lo cual explica las ramificaciones de la grafica. Así mismo al hacer uso de pocos nodos para lograr evadir los huecos, el porcentaje de utilización de la red es de aproximadamente 63%, como se muestra en la grafica del Anexo 58. En comparación con los demás protocolos, GDSTR presenta el mejor porcentaje de utilización del a red, debido a que a diferencia de los protocolos de 2-f-GEDIR y 2-f-MFR, no hace uso de *flooding* para evadir los huecos, y a diferencia de GEAR, no hace uso de múltiples rutas para balancear la energía.

5.1.5.GAF

El escenario sobre el cual GAF fue ejecutado es el que se muestra en el Anexo 65. Esto se debe a que el funcionamiento de GAF debe ser evaluado sobre una red en la que la conectividad de los nodos sea sumamente alta, por lo que el escenario determinístico del Anexo 23, en donde cada nodo tiene máximo 4 vecinos no puede ser utilizado para probar a GAF; por esta misma razón se explicara posteriormente porque en los escenarios aleatorios bajo las densidades elegidas GAF no puede ser implementado; como se observa por la figura del Anexo 65, el nodo con id 23, posee 28 vecinos dentro de su área de cobertura, esto es así debido a que dentro de cada grilla deben existir varios nodos para que puedan ocurrir los cambios estado, es decir una grilla en la que si un nodo pasa de *active* a *sleep*, tenga a algún vecino que se active para encargarse del enrutamiento de paquetes.

Como se observa de la grafica del Anexo 66, la tasa de entrega de paquetes varia, es decir muchos de los paquetes enviados no logran llegar al destino. Esto se da debido a que cuando un nodo cambia de estado *active* a *discovery* todos los paquetes que estaba enrutando, son eliminados. Esto también sucede porque en cuando un nodo en una grilla está enviando paquetes a un nodo de una grilla adyacente, y este ultimo cambia a estado *discovery* o *sleep*, los paquetes que envía el nodo no tendrán quien los reciba. Por esto la red tiende a tener nodos cóncavos, como se muestra en la grafica del Anexo 67, en la que puede observarse que los paquetes entran a nodos cóncavos.

Cuando un nodo determina que entre todos sus vecinos activos él es el que tiene menor métrica al destino, espera durante un lapso de 60 segundos a que algún vecino con menor métrica se active en una de las grillas adyacentes; cuando esto sucede los paquetes presentan un *end to end delay* que puede tomar valores cercanos o menores a los 60 segundos, como se indica en la grafica de *end to end delay* del Anexo 71. Esto en el caso de que el nodo encuentra a algún otro nodo a quien reenviar los paquetes antes de que hayan transcurrido los 60 segundos, si no se encuentra a ningún vecino con menor métrica al destino luego de 60 segundos, el nodo reenvía el paquete al vecino que tenga menor métrica al destino (la cual es obviamente mayor a la del nodo que esta reenviando el paquete); esta implementación se realizó para evitar perder todos los paquetes que el nodo posee y que no puede entregar si no encuentra a nadie con menor métrica. De esta manera otro nodo se encarga del enrutamiento del paquete, por lo que el número de saltos que efectúa el paquete al destino varía constantemente. En la grafica de número de saltos del Anexo 70, se puede notar que los paquetes efectúan una gran cantidad de saltos a través de los nodos que se encuentran activos, debido a que el paquete los nodos siempre tratan de entregar el paquete, por las razones expresadas anteriormente. Por la misma razón el IPDV entre paquetes cambia constantemente ya que el paquete viaja por los nodos que se encuentren activos y estos durante toda la simulación no son los mismos, como se indica en la grafica de nodos activos del anexo Anexo 68.

El objetivo de GAF es el de consumir poco a poco la energía de los nodos, debido a esto las graficas de porcentaje de utilización de los nodos, Anexo 72, no tienden a disminuir como lo hacen en los otros protocolos, si no que aumentan o disminuyen a medida que el nodo cambia de estado. Esto también puede observarse en la grafica de energía, en donde las pendientes representan cuando un nodo se encuentra activo y transmitiendo paquetes; cuando se encuentran en un valor fijo significa que el nodo está en estado *sleep*. Por último se puede observar en la grafica del Anexo 74, en la que se muestra el porcentaje de utilización de los nodos, que aproximadamente son usados 53% de los nodos de la red. Esto es de esperarse debido a que todos los nodos de la red no están activos al mismo tiempo, y por esto no todos se usan.

5.2. ESCENARIOS ALEATORIOS

A continuación se especifica el desarrollo de los escenarios aleatorios sobre los cuales se ejecutan los protocolos de enrutamiento para su evaluación comparativa en base a los parámetros elegidos. El procedimiento fue generar 50 escenarios con diferentes topologías para cada una de las densidades de nodos a trabajar y bajo los mismos ejecutar cada uno de los protocolos generando 500 paquetes de información en intervalos de 10 segundos recogiendo los datos necesarios para su análisis; esta recolección de datos es definida en Opnet como la suma de muestras en un intervalo de tiempo. Los resultados obtenidos son mostrados en gráficas donde se observa el parámetro respectivo contra el escenario simulado.

Se procede a hacer un análisis comparativo de cada parámetro teniendo en cuenta su variación con respecto a la densidad de la red, donde este análisis es hecho sobre el comportamiento general de cada parámetro. Se presentaron casos erráticos e inesperados que no siguen el comportamiento descrito por los autores en donde se busca justificar porque se dio la existencia de estos. Los parámetros a comparar son: *Network Use*, IPDV, *Hop Count*, Tasa de Entrega, Nodos Activos, Nodo Concavo, *End-to-End delay* y Energia Restante.

Para todas las graficas se tiene que

Tabla 12 Identificación de los protocolos en las graficas

	2-f-GEDIR		**2-f-MFR**		**GDSTR**		**GEAR**

5.2.1. Network Use

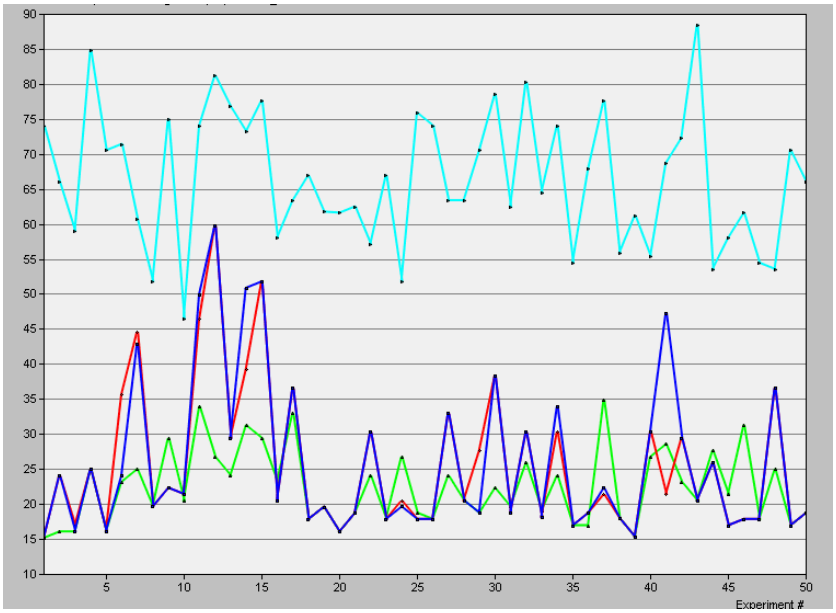


Figura 36 Densidad correspondiente a 110 nodos

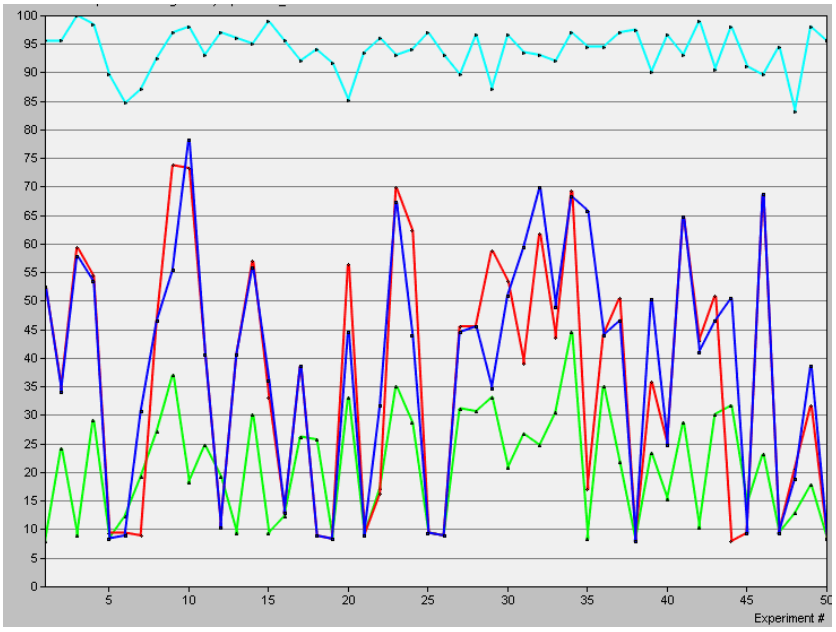


Figura 37 Densidad correspondiente a 200 nodos

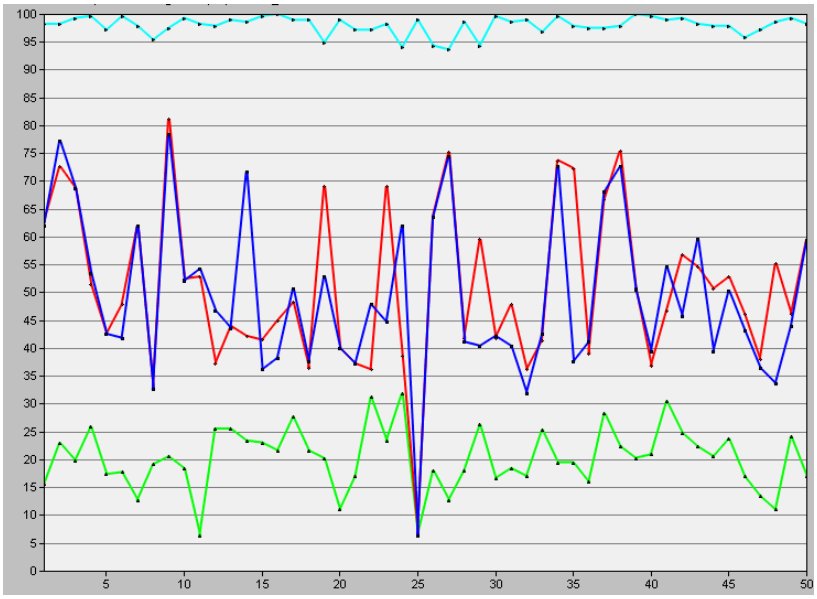


Figura 38 Densidad correspondiente a 280 nodos

Al analizar la Figura 36, la Figura 37 y la Figura 38, se observa que el protocolo que usa mayor cantidad de nodos de la red es GEAR lo cual es correspondiente al funcionamiento descrito por sus autores. Esto se debe a que GEAR siempre busca nuevas rutas al hacer balance de energía entre nodos con base en las distancias y las energías consumidas de estos, por lo tanto usa mayor cantidad de nodos. Los protocolos 2-f-GEDIR y 2-f-MFR tienen un comportamiento muy similar en la mayoría de los casos a excepción de algunos donde la métrica usada es la que determina la variación entre uno y otro. Así mismo, para estos protocolos se presentan sobre picos cuya causa es el uso del método *flooding* para salir de un nodo cóncavo; esto es comprobado al observar el escenario 12 en las demás gráficas.

Por último, el protocolo GDSTR es el protocolo que hace menos uso de nodos de la red, ya que es un protocolo que no inunda la red con paquetes y mantiene rutas del origen al destino final hasta agotar algún nodo y cambiar de topología. Para la tercera densidad, Figura 38, en el escenario 25 se presenta un caso donde tanto 2-f-GEDIR como 2-f-MFR y GDSTR hacen uso de la misma cantidad de nodos; este caso se debe a que el funcionamiento básico de los tres protocolos es basado en el algoritmo *Greedy* y todos tienen unas rutas directas del nodo fuente al nodo destino. La variación de estos comportamientos con respecto al cambio de densidad de la red se mantiene, haciendo de GDSTR el protocolo que menos uso hace de la red.

5.2.2. IPDV

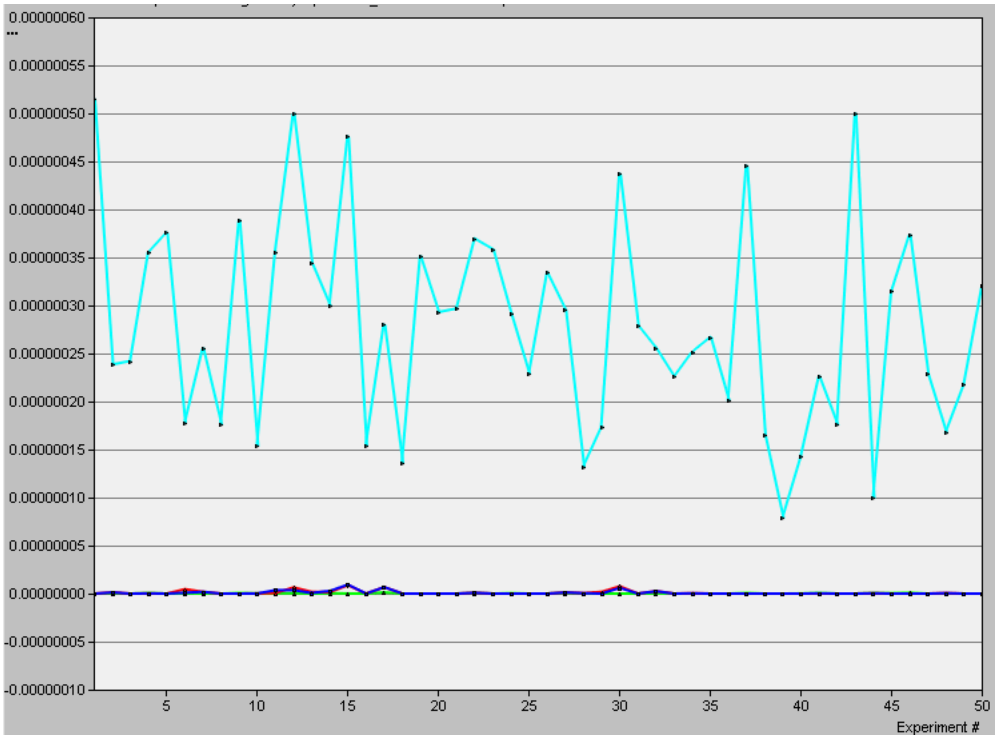


Figura 39 Densidad correspondiente a 110 nodos

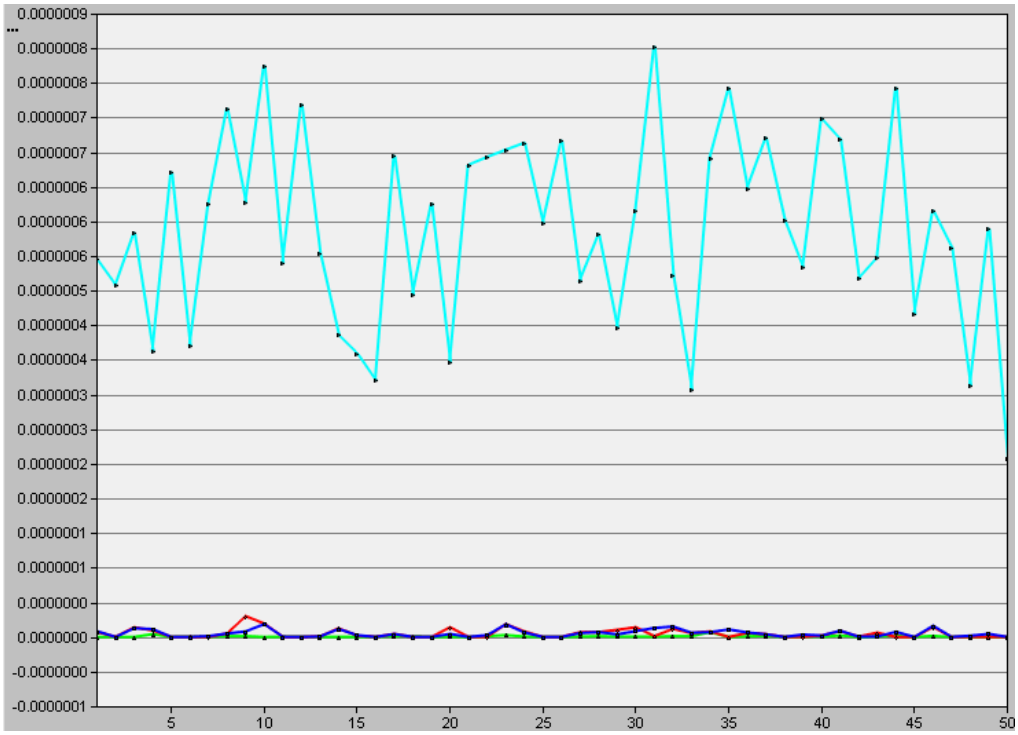


Figura 40 Densidad correspondiente a 200 nodos

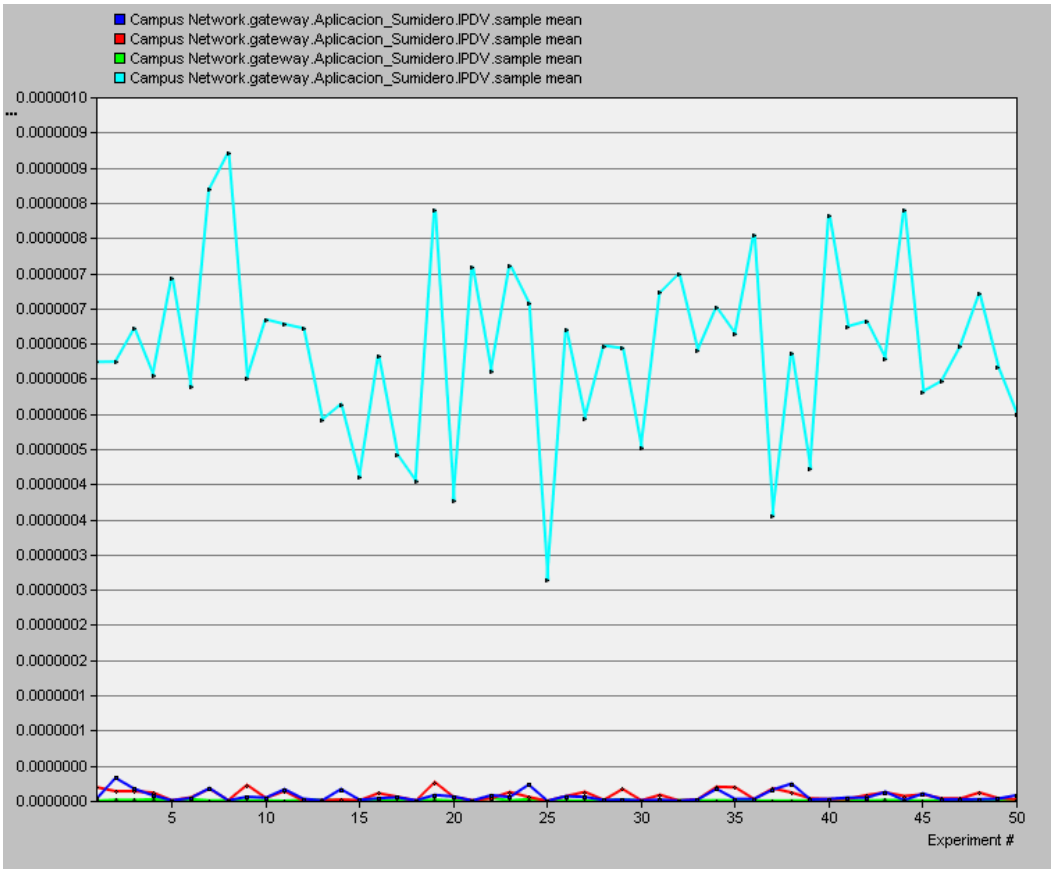


Figura 41 Densidad correspondiente a 280 nodos

Se observa de las graficas del IPDV que el protocolo GEAR es el que posee los mayores valores de este parámetro, haciendo de este protocolo poco confiable si se requiere una oportuna entrega de paquetes. Esto sucede ya que GEAR cambia de rutas constantemente, variando el ETE delay y por lo tanto cambiando el IPDV. El parámetro posee un comportamiento muy similar entre los protocolos 2-f-GEDIR, 2-f-MFR y GDSTR ya que mantienen valores en los mismos rangos, siendo esto coherente con sus respectivos funcionamientos. El protocolo con menor variación de IPDV, es GDSTR (aunque no es significativa) debido a que los paquetes toman la misma ruta llegando en los mismos intervalos de tiempo, hasta cuando hay un cambio de topología donde encuentra una nueva ruta. Para 2-f-GEDIR y 2-f-MFR existen aumentos repentinos, los cuales se deben al realizar un *flooding* donde el paquete tarda en encontrar una nueva ruta.

5.2.3. End-to-end delay

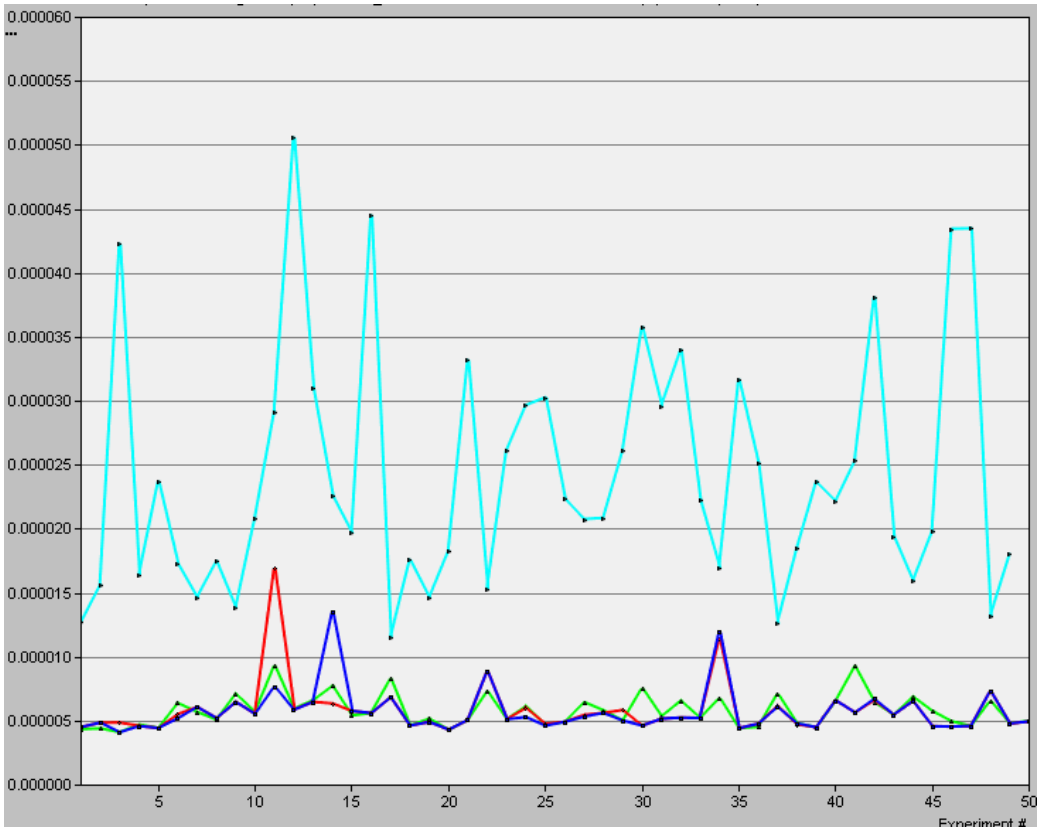


Figura 42 Densidad correspondiente a 110 nodos

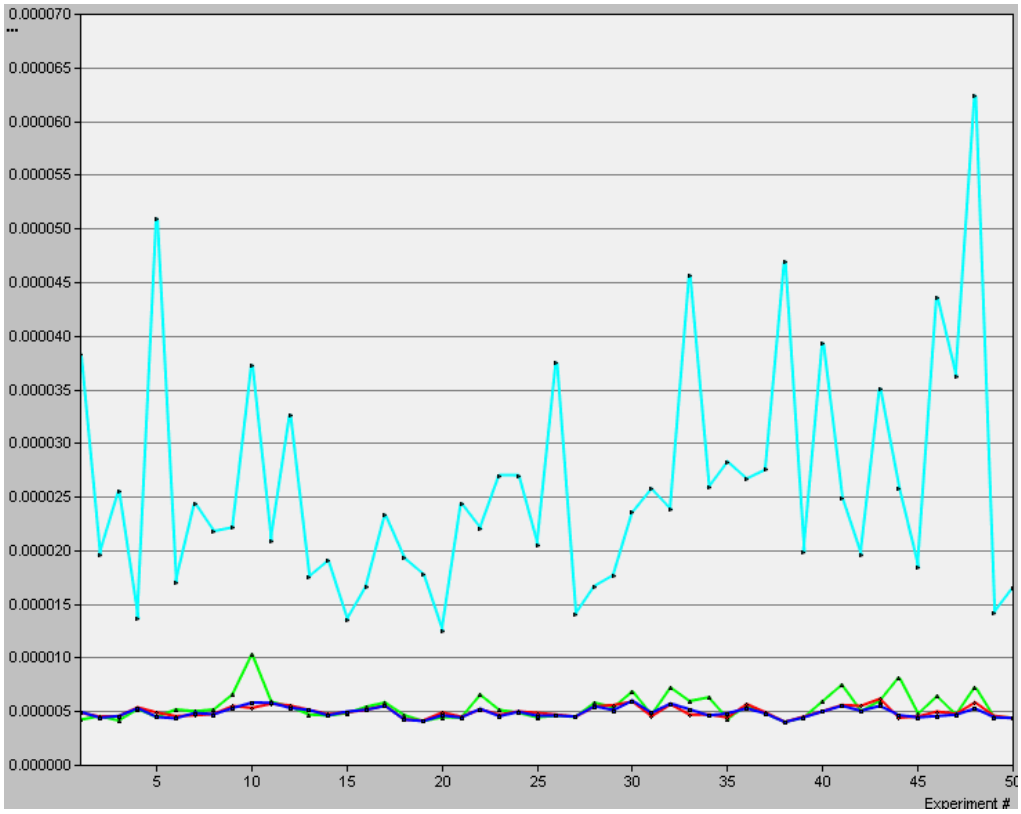


Figura 43 Densidad correspondiente a 200 nodos

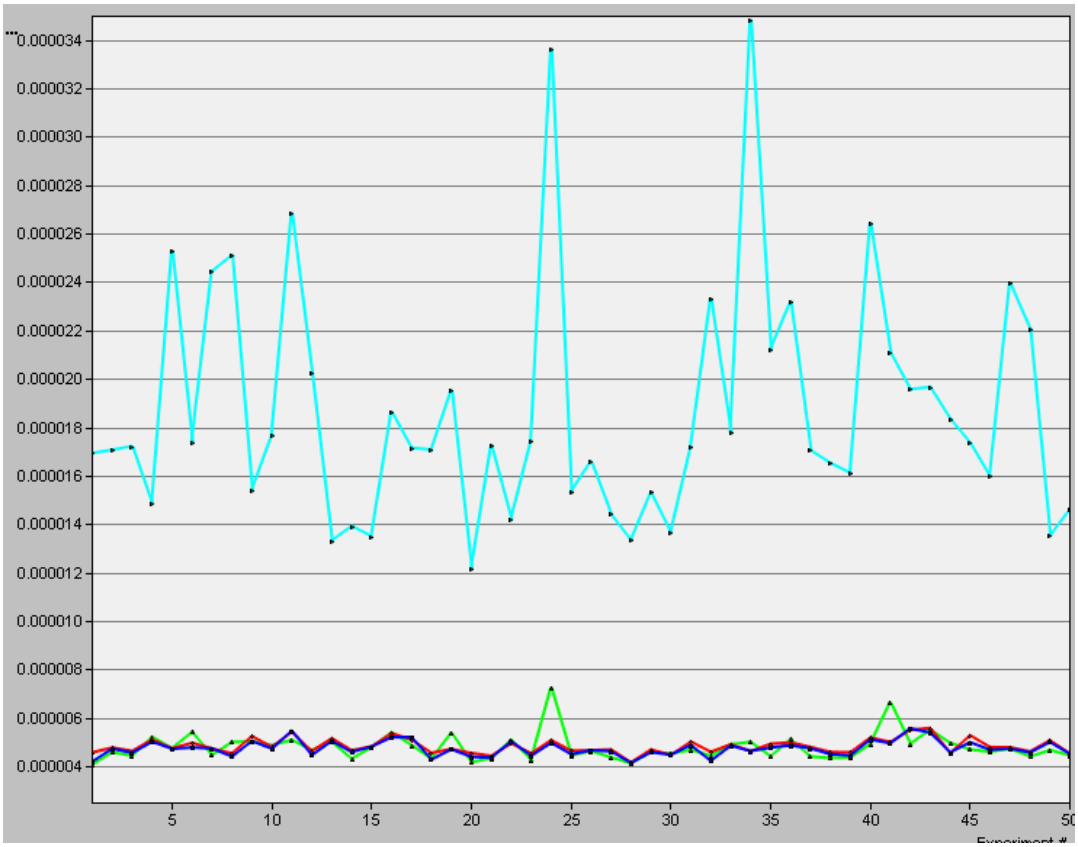


Figura 44 Densidad correspondiente a 280 nodos

En las gráficas se observa el correcto comportamiento de GEAR donde al estar cambiando de ruta constantemente el paquete cambia el tiempo que tarda en ser entregado por lo tanto su ETE delay. Para el primer escenario, los protocolos 2-f-GEDIR y 2-f-MFR tienen en promedio un mejor desempeño exceptuando ciertos casos donde se generan multiples *floodings*, es decir donde el paquete llega a un nodo cóncavo multiples veces o donde un solo *flooding* inunda de replicas la red haciendo que la recolección de datos sea imprecisa; para saber cuál es la causa correspondiente a estos casos se debe referir a las gráficas de paquetes en nodos cóncavos. El caso contrario se observa en GDSTR donde, en algunos escenarios al crear las topologías de los arboles, los paquetes pueden llegar a tardarse más al tener que recorrer más ramas las cuales pueden no ser la ruta necesaria para alcanzar el nodo destino.

5.2.4. Hop count

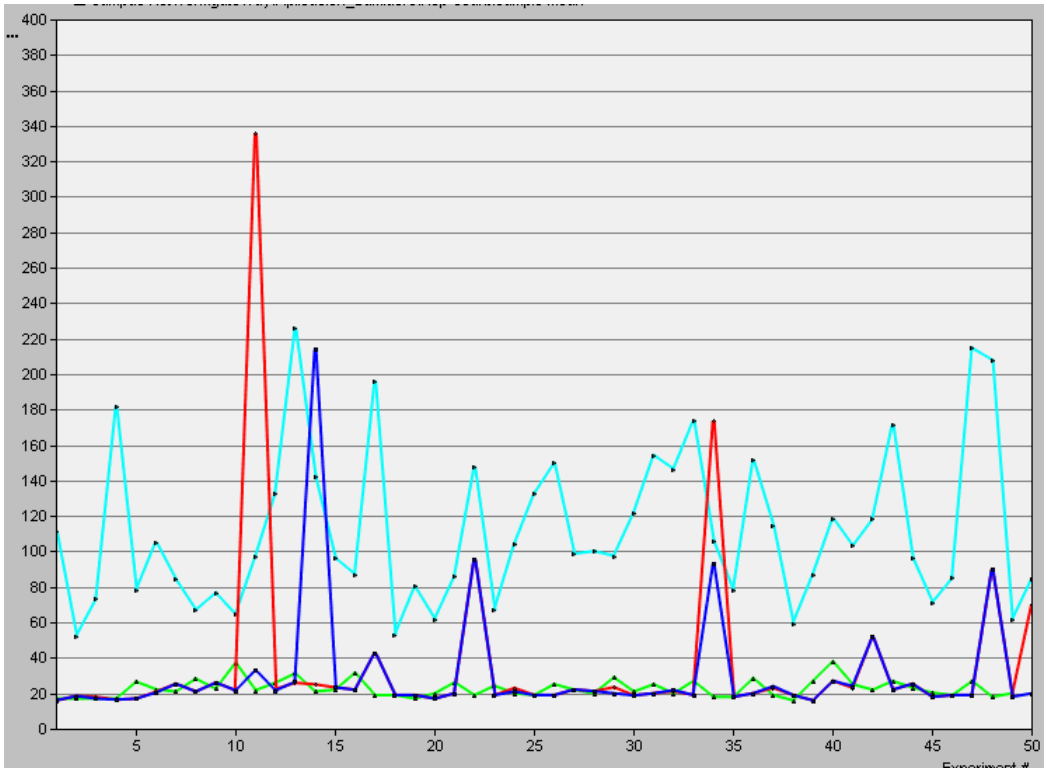


Figura 45 Densidad correspondiente a 110 nodos

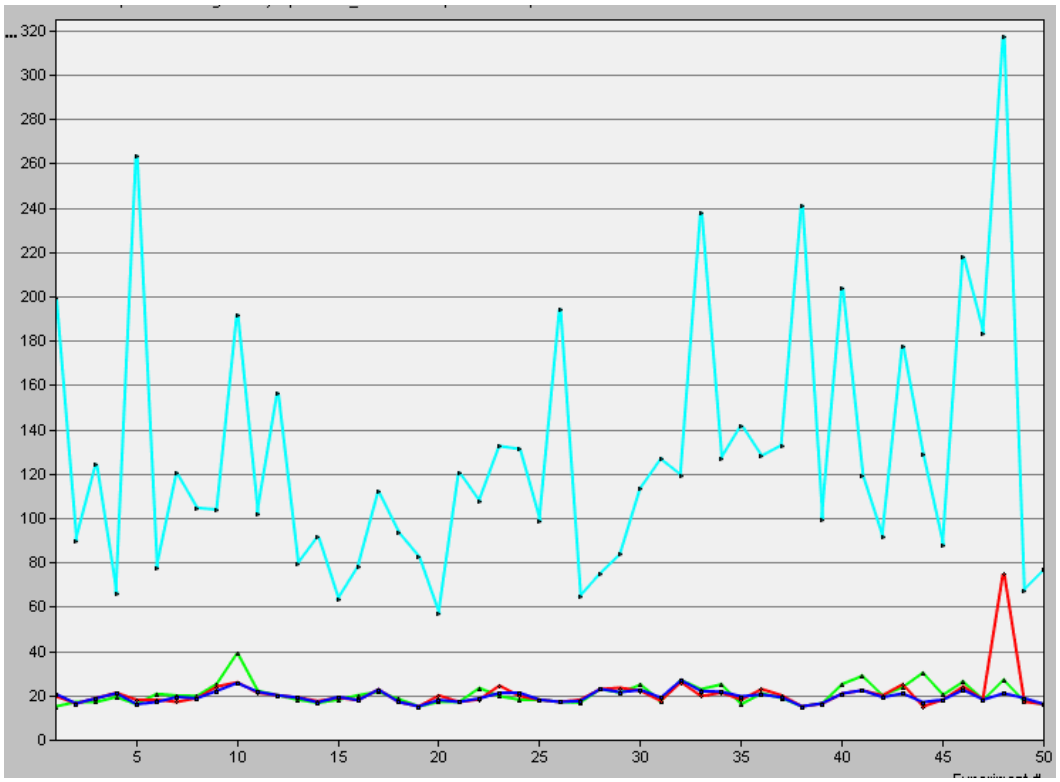


Figura 46 densidad correspondiente a 200 nodos

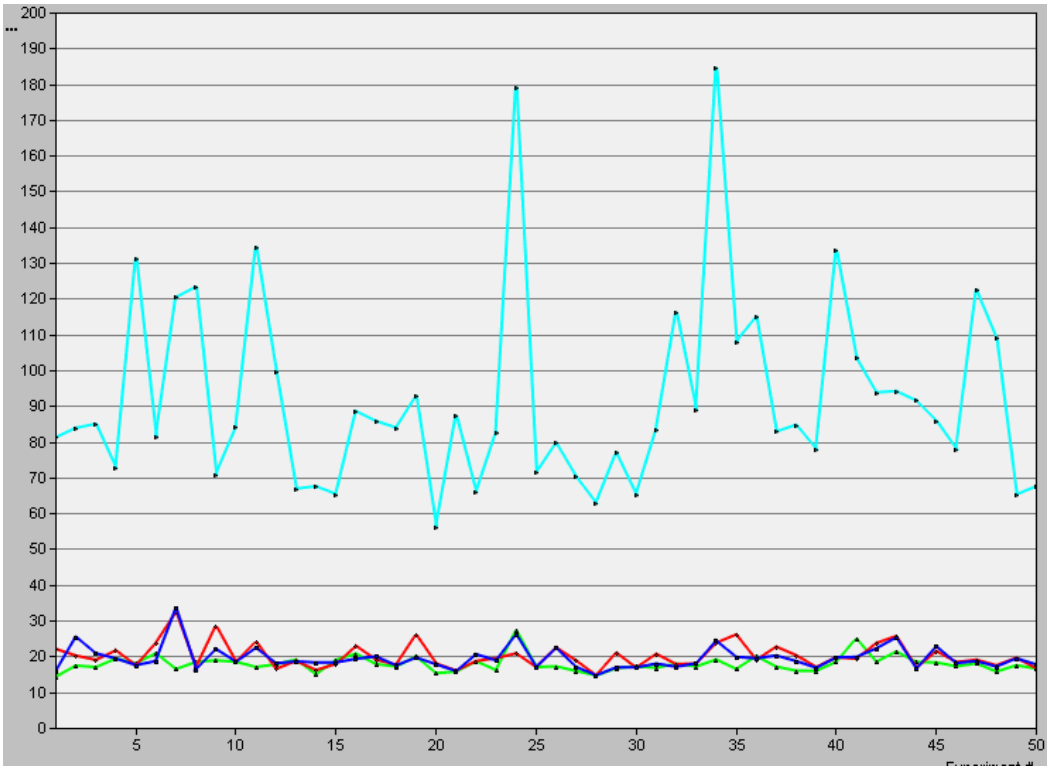


Figura 47 Densidad correspondiente a 280 nodos

En el caso del promedio de número de saltos realizados en un escenario, existen grandes diferencias al cambiar la densidad de la red. Para la densidad de 110 nodos, los protocolos 2-f-GEDIR y 2-f-MFR tienen un bajo desempeño al tener un número de saltos significativamente alto en ciertos escenarios. En estos escenarios se están replicando un gran número de paquetes al realizar los *floodings* y cabe notar que debido a la manera en que Opnet recoge datos para el análisis, este valor es la suma del número de saltos realizados de cada paquete en ese intervalo de tiempo, haciendo de este valor más un indicio de un comportamiento diferente que de un valor conciso del promedio de numero de saltos. Para GEAR este valor es mayor a los demás debido a las razones citadas anteriormente. Al aumentar la densidad de la red, el número de saltos se hace comparable entre 2-f-GEDIR, 2-f-MFR y GDSTR ya que la posibilidad de existencia de nodos cóncavos va disminuyendo y por lo tanto también disminuye la posibilidad de utilización de *floodings* y de *spanning trees*.

5.2.5. Nodo cóncavo

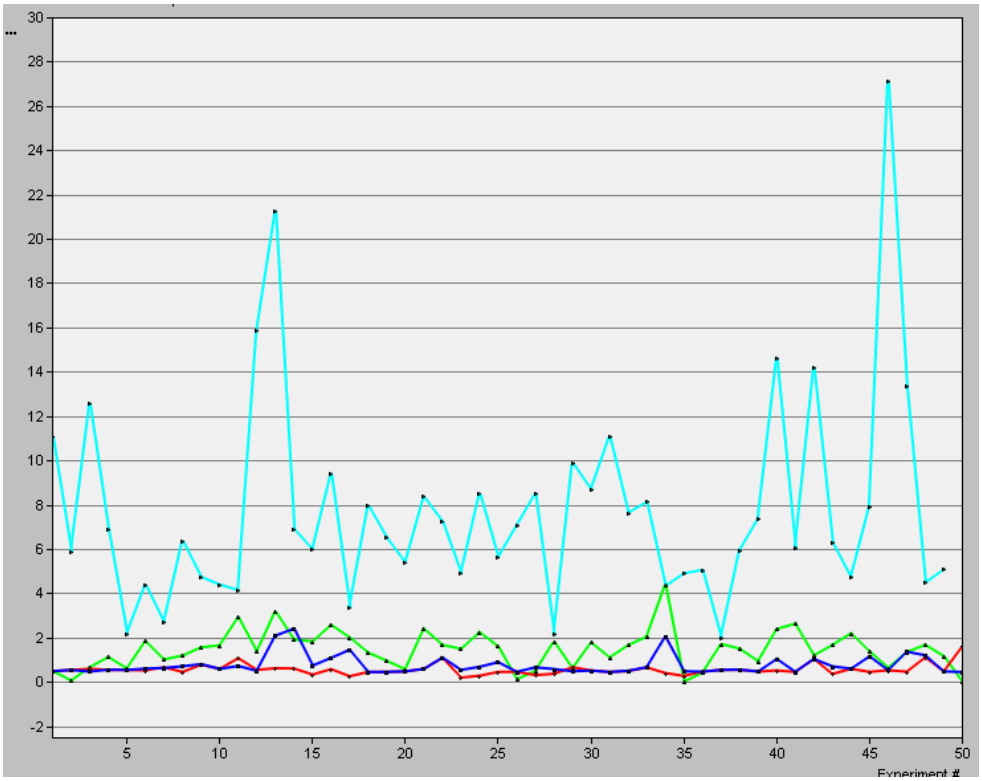


Figura 48 Densidad correspondiente a 110 nodos

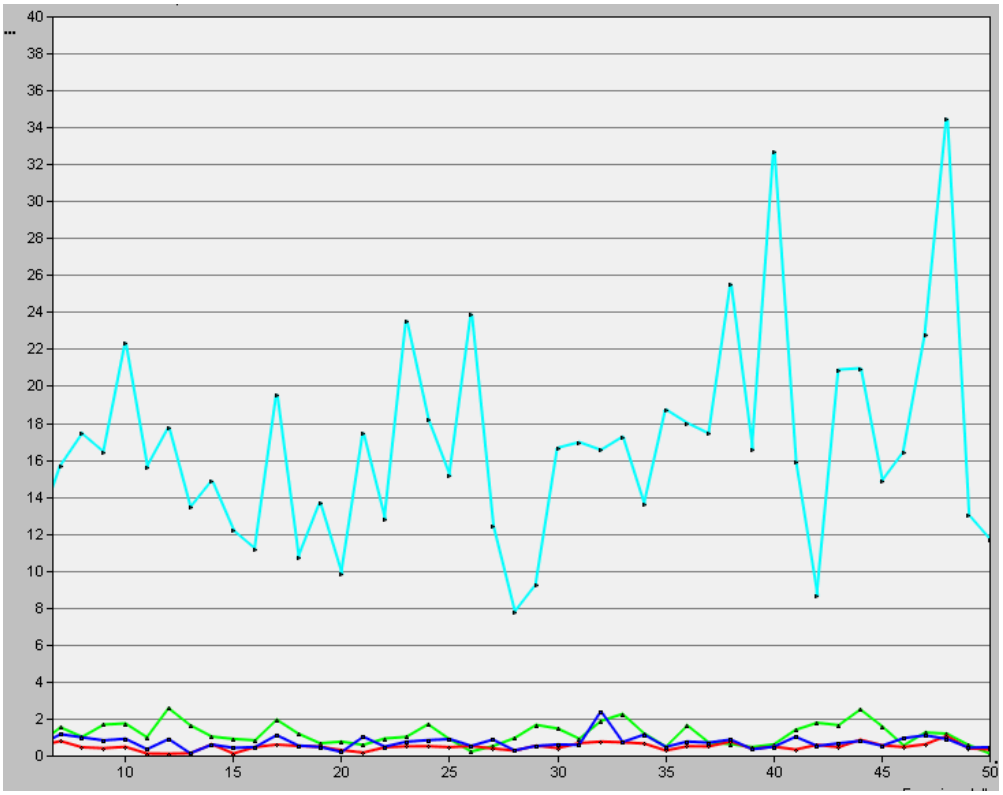


Figura 49 Densidad correspondiente a 200 nodos

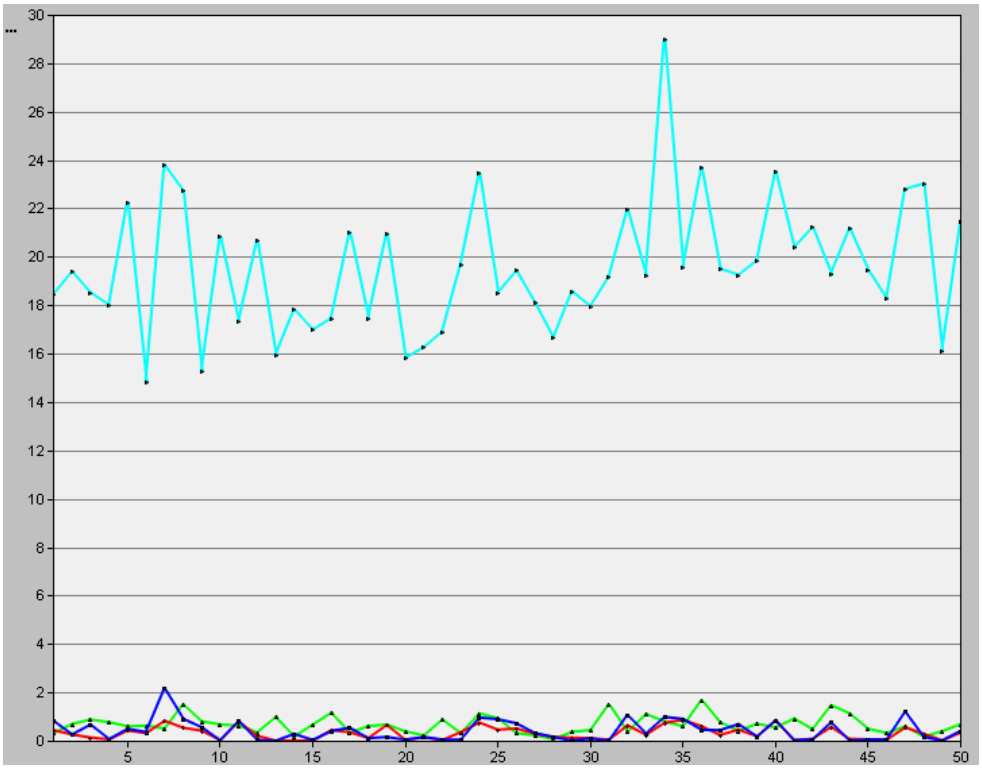


Figura 50 Densidad correspondiente a 280 nodos

Como se puede observar en la gráfica correspondiente a 110 nodos, el método de *flooding* para salir de un nodo cóncavo, aprender de la existencia de este y encontrar una ruta es mejor que el de los otros protocolos debido a que un paquete en GDSTR siempre vuelve a un nodo cóncavo existente en la ruta del nodo fuente al nodo destino y que en GEAR siempre se generaran nodos cóncavos por su definición dada por el protocolo haciendo critico ajustar los valores respectivos para el cálculo de los costos de los nodos. Cuando la densidad aumenta, el número de veces que un paquete entra en un nodo cóncavo disminuye ya que la existencia de estos se reduce de igual manera y si entra en un nodo cóncavo, al tener los nodos gran conectividad, encuentran rápidamente una nueva ruta con poca probabilidad de la existencia de un nodo cóncavo.

5.2.6. Energía restante

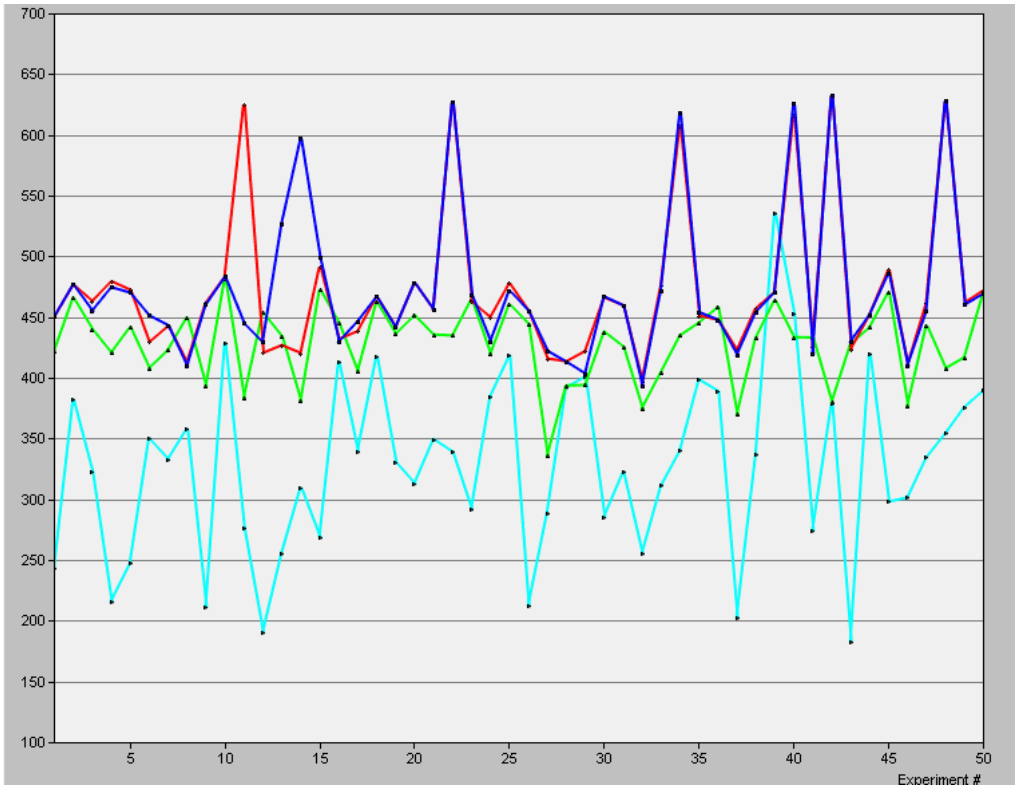


Figura 51 Densidad correspondiente a 110 nodos

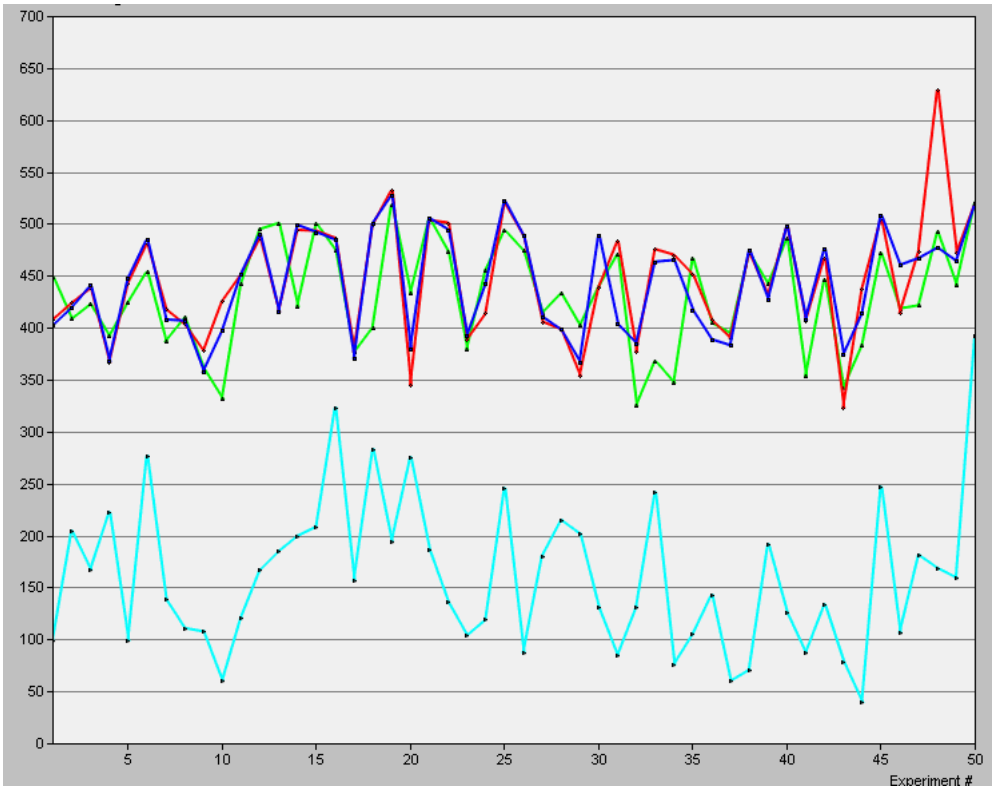


Figura 52 densidad correspondiente a 200 nodos

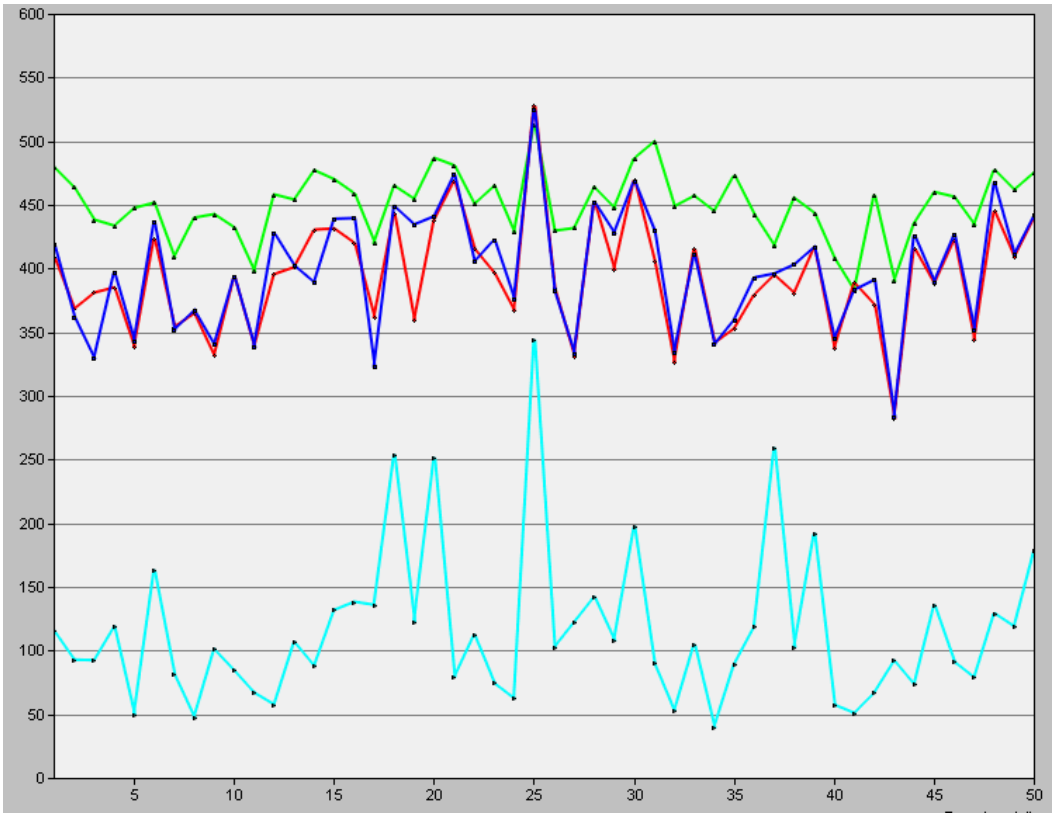


Figura 53 Densidad correspondiente a 280 nodos

Observando las gráficas de energía restante promedio de la red, los protocolos 2-f-GEDIR y 2-f-MFR en el primer escenario quedan con mayor energía dadas dos situaciones: cuando los nodos identifican cuales son los nodos cóncavos de la red y no los vuelven a usar en el enrutamiento, por lo cual no desperdician energía; y al hacer el *flooding* afectando una cantidad reducida de nodos por lo cual no existen muchas replicas en la red que consuman la energía de los estos. La situación cambia al aumentar la densidad ya que un *flooding* se vuelve significativo en consumo de energía cuando existe mucha copias de un paquete consumiendo cada una energía de cada nodo en la ruta hacia el destino. La aproximación de GDSTR empieza a mejorar cuando aumenta la densidad, debido a que solo existe un paquete y este paquete intentara recorrer el árbol el menor tiempo posible. Para el protocolo GEAR se observa una situación adversa ya que al ser un protocolo creado para hacer un uso eficiente de energía en el proceso de enrutamiento, bajo los 3 escenarios es el protocolo que más consume este recurso. La causa de esto es que GEAR es un protocolo que necesita ser ajustado en sus cálculos de costos, es decir, deben buscarse variaciones comparables en sus funciones de costo de transmisión y de cálculo de costos estimados, y tener en consideración situaciones específicas donde el funcionamiento del protocolo no es el más indicado. Finalmente es necesario ver la relación directa entre la tasa de

entrega y el consumo de energía, donde existe varias posibilidades: que una red pierda conectividad rápidamente y los nodos no consuman energía o donde alcanza a entregar una gran cantidad de paquetes e hizo un consumo de energía balanceado.

5.2.7. Tasa de entrega

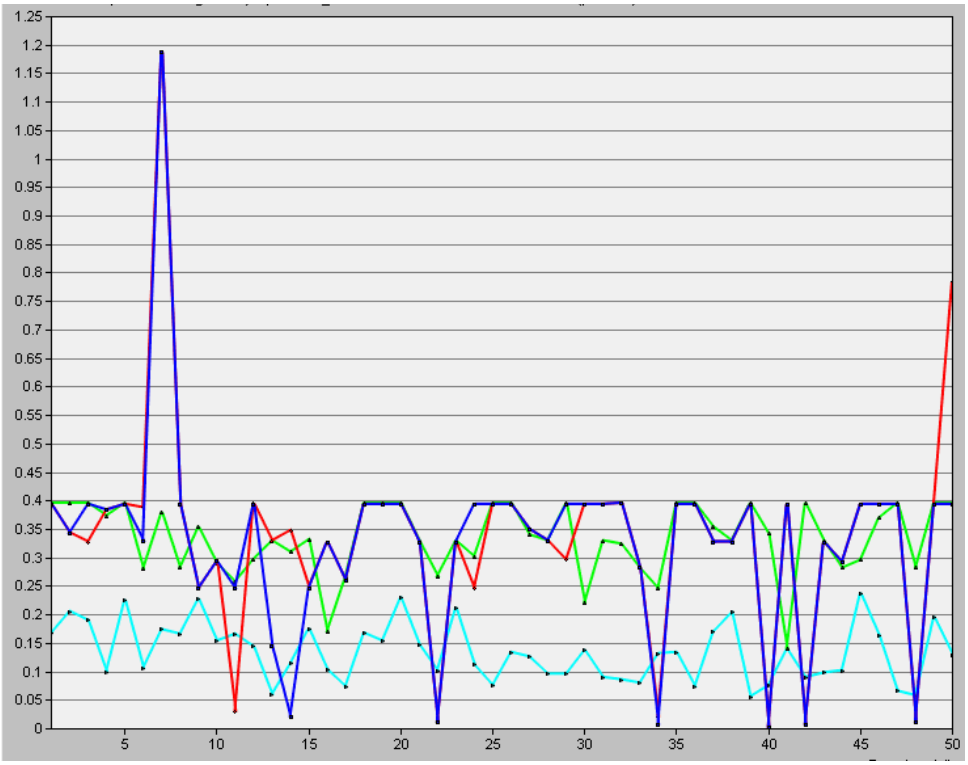


Figura 54 Densidad correspondiente a 110 nodos

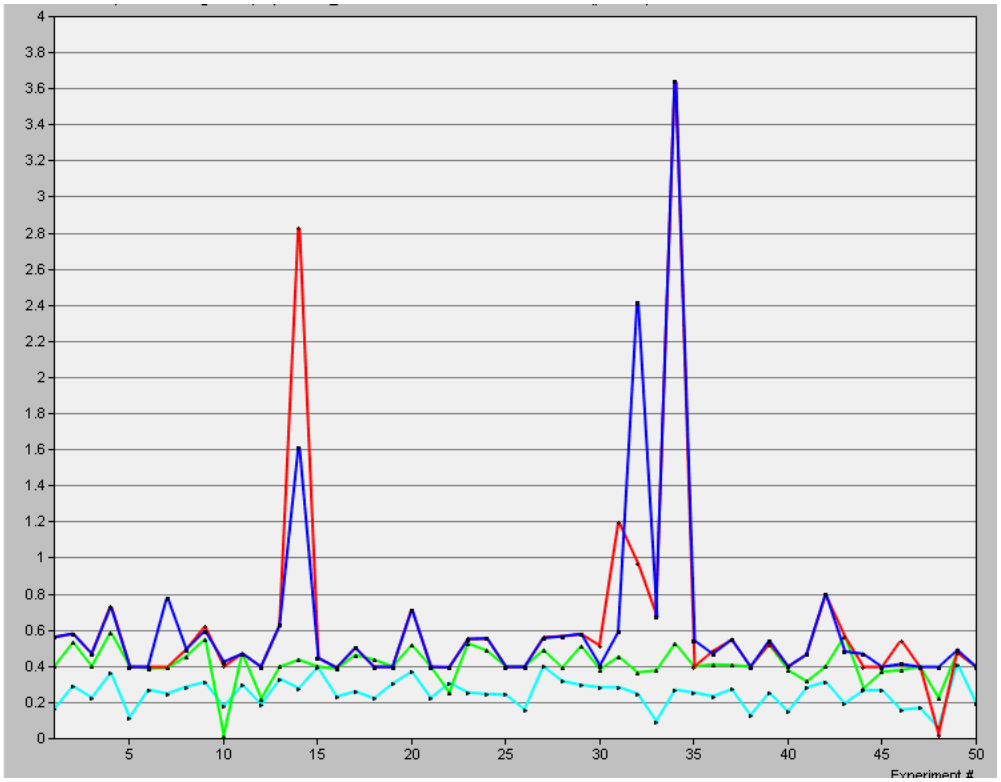


Figura 55 Densidad correspondiente a 200 nodos

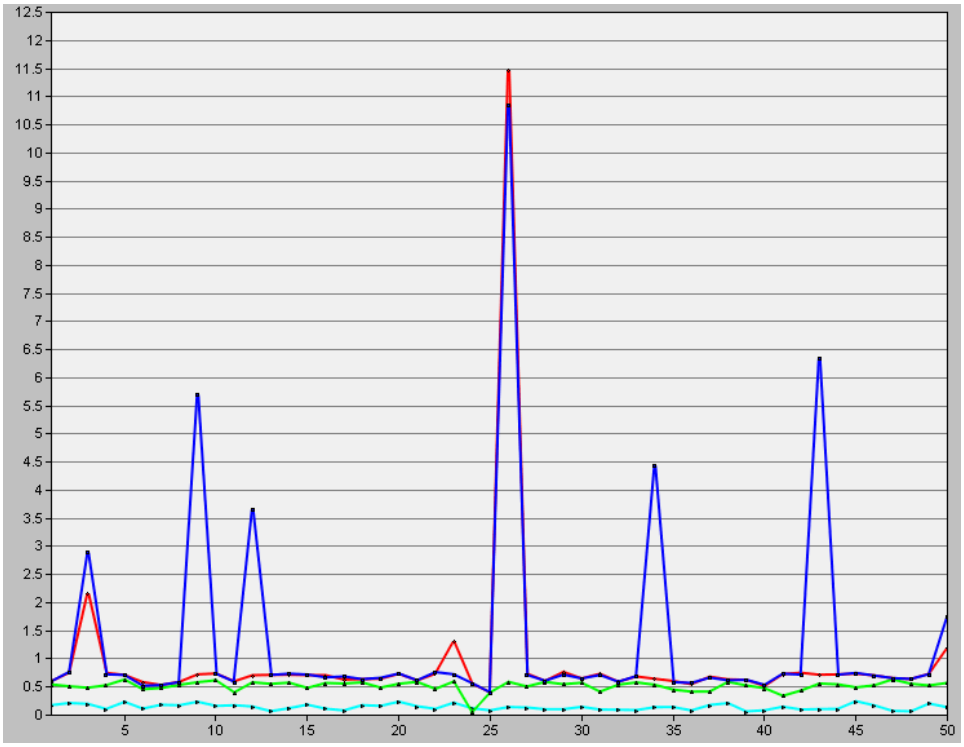


Figura 56 Densidad correspondiente a 280 nodos

El parámetro de la tasa de entrega es un valor de gran importancia en una red, es el parámetro que da una relación entre el protocolo y la confiabilidad de la red. Al observar las graficas se tiene que los protocolos con mayor tasa de entrega son 2-f-MFR y 2-f-GEDIR. Esto son datos bastante aproximados e inexactos ya que este parámetro es necesario verlo en conjunto con las demás graficas.

Es un parámetro que depende mucho de la cantidad de paquetes que llegaron en un instante al nodo destino, es decir se debe verificar si son paquetes de replicas creados por método *flooding*, o por la cantidad de veces que el paquete entra en un nodo cóncavo, o por recorrido hecho sobre las ramas del árbol y por la decisión de re-envío dada por GEAR. Son datos que más que mostrar valores exactos, indican el funcionamiento general del protocolo y reflejan de qué manera fueron implementados.

6. CONCLUSIONES

Con el fin de dar continuidad a los estudios e investigaciones de las redes WSN, se pretende por medio de este trabajo contribuir a la creación inicial de una investigación comparativa sobre protocolos de enrutamiento en redes WSN y de unos algoritmos implementados en lenguaje proto-C sobre el programa de simulación de redes Opnet, para dar una base en el funcionamiento del enrutamiento, en la implementación de protocolos y en el uso de la herramienta para el futuro desarrollo de proyectos.

El protocolo GAF bajo el escenario de prueba creado para comprobar su correcto funcionamiento no presentó problemas algunos, se logró lo propuesto por los autores. En el momento de implementar este protocolo sobre el escenario determinístico se observó que bajo la densidad hecha para este escenario, el protocolo nunca tendría conectividad por lo que fue necesario hacer un tratamiento especial para realizar el análisis de este. De igual manera, en los escenarios aleatorios bajo las 3 cantidades de nodos escogidas, el protocolo nunca tendría un buen funcionamiento a causa de varias razones: la primera, es un protocolo que al apagar nodos periódicamente está creando nodos cóncavos en toda la red, esto implica que el protocolo sobre el cual debe estar acoplado debe ser uno con una gran facilidad para evadirlos o salir de ellos; la segunda, es un protocolo que requiere un alto grado de procesamiento de la cpu y al intentar crear un escenario con la cantidad de nodos suficientes para detallar su funcionamiento, agota los recursos de memoria RAM y procesador antes de obtener los resultados. Esto acoplado a un

enrutamiento *Greedy* básico, si se acopla a un protocolo que evada nodos cóncavos, esta situación será peor; por último si se logra implementar GAF en una red supremamente densa y lograr la recolección de datos, estos datos no serian relevantes en el momento de comparación, ya que al ser tan densa la red y siendo que la diferencia de los protocolos en la mayoría de los casos es su forma de cómo tratar la situación de un nodo cóncavo, este objetivo se perdería ya que los protocolos trabajarían constantemente en modo *Greedy*.

Con respecto a la evaluación comparativa entre protocolos, no es posible determinar un protocolo que sea mejor en todos los parámetros analizados, ya que el comportamiento de cada uno no es constante bajo todos los escenarios. Esto se da por unos casos inesperados que no son tenidos en cuenta por los autores de cada protocolo, donde fue necesario implementar una solución que puede no ser la más optima a causa de la falta de precisión en la documentación a la hora de ajustar valores de funcionamiento. Aun así, la evaluación comparativa realizada si da un indicio básico de que protocolo puede ser mejor bajo algún parámetro elegido. En general el protocolo con mejor comportamiento a lo largo de varios parámetros es el protocolo GDSTR.

El objetivo inicial de este proyecto de grado era realizar una investigación haciendo uso de información existente, que pudiera aportar al estado del arte que se tiene de las redes WSN pero debido que a lo largo del desarrollo del proyecto se encontraron obstáculos imprevistos, el enfoque del proyecto pasó a de ser un proyecto netamente investigativo a un proyecto que adicionalmente requirió un diseño extra de los protocolos para poder evaluarlos de una manera general por lo cual el alcance del proyecto se extendió implicando más tiempo para lograr solucionar los inconvenientes que se presentaron en cada protocolo.

Finalmente, la creación de código de cada protocolo y su correcta implementación en el software elegido presentó un gran desafío dado que la curva de aprendizaje fue lenta a causa de la falta de información de primera mano. Esto conllevó a tener problemas imprevistos de los cuales algunos fueron resueltos por el soporte técnico, y otros no tuvieron respuesta, por lo cual fue necesario buscar una solución que puede no ser la más eficiente frente a las capacidades de un simulador tan robusto como lo es Opnet.

7. Bibliografía

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci, "A Survey on Sensor Networks", Georgia Institute of Technology, IEEE Magazine, August 2002.

[2] Azzedine Boukerche, "Algorithms and Protocols for Wireless Sensor Networks", University of Ottawa, Copyright © 2009 by John Wiley & Sons Ltd., Capitulo 6.

[3] Jamal N. Al-Karaki, Ahmed E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey", Iowa State University Dept. of Electrical and Computer Engineering.

[4] Edgardo Avilés López, "Redes Inalámbricas de Sensores Estado del Arte", CICESE, Seminario de Investigación, Junio 2005.

[5] Holger Karl, Andreas Willig, "Protocols and Architectures for Wireless Sensor Networks", Copyright© 2005 John Wiley & Sons Ltd., Capitulo 11.

[6] Dorothea Wagner, Roger Wattenhofer, "Algorithms for Sensor and Ad Hoc Networks", Springer-Verlag Berlin Heidelberg 2007, Capitulo 9.

[7] Hac, Anna, "Wireless Sensor Network Designs", University of Hawaii at Manoa, Honolulu, USA, Copyright © 2003 by John Wiley & Sons, Ltd., Capitulo 4.

[8] Raúl Aquino-Santos, Luis A. Villaseñor-González, Víctor Rangel Licea, Omar Álvarez Cárdenas, Edwards Block, "Análisis del funcionamiento de estrategias de enrutamiento para redes de sensores inalámbricos", Revisita Facultad de Ingeniería Universidad de Antioquia N.° 52 pp. 185-195. Marzo, 2010

[9] Ya Xu, John Heidemann, Deborah Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing", ACM Mobicom, July 16-21, 2001, Rome, Italy

[10] Ivan Stojmenovic, Xu Lin, "Loop-Free Hybrid Single-Path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks", 5 June 2001 IEEECS Log Number 110520.

[11] Ben Leong, Barbara Liskov, and Robert Morris, "Geographic Routing without Planarization", MIT Computer Science and Articial Intelligence Laboratory, 2006.

[12] Yan Yu, Ramesh Govindan, Deborah Estrin, "Geographical and Energy Aware Routing: A recursive data dissemination protocol for wireless sensor networks", August 14, 2001.

[13] Mahtab Seddigh, Ivan Stojmenovic, Jie Wu, “Location and internal nodes based routing algorithms in wireless networks”

[14] Sheldon M. Ross, “Simulación”, Prentice Hall, México, 1999, Capitulo 7

[15] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, “Wireless sensor network survey”, Computer Networks 52 (2008) 2292–2330, Elsevier B.V.

[16] Referencia [3] y http://www.geoan.com/analitica/vectores/producto_punto.html

[17] XBee™/XBee-PRO™ OEM RF Modules, Product Manual v1.xAx - 802.15.4 Protocol, IEEE® 802.15.4 OEM RF Modules by MaxStream, Inc., Página 4.

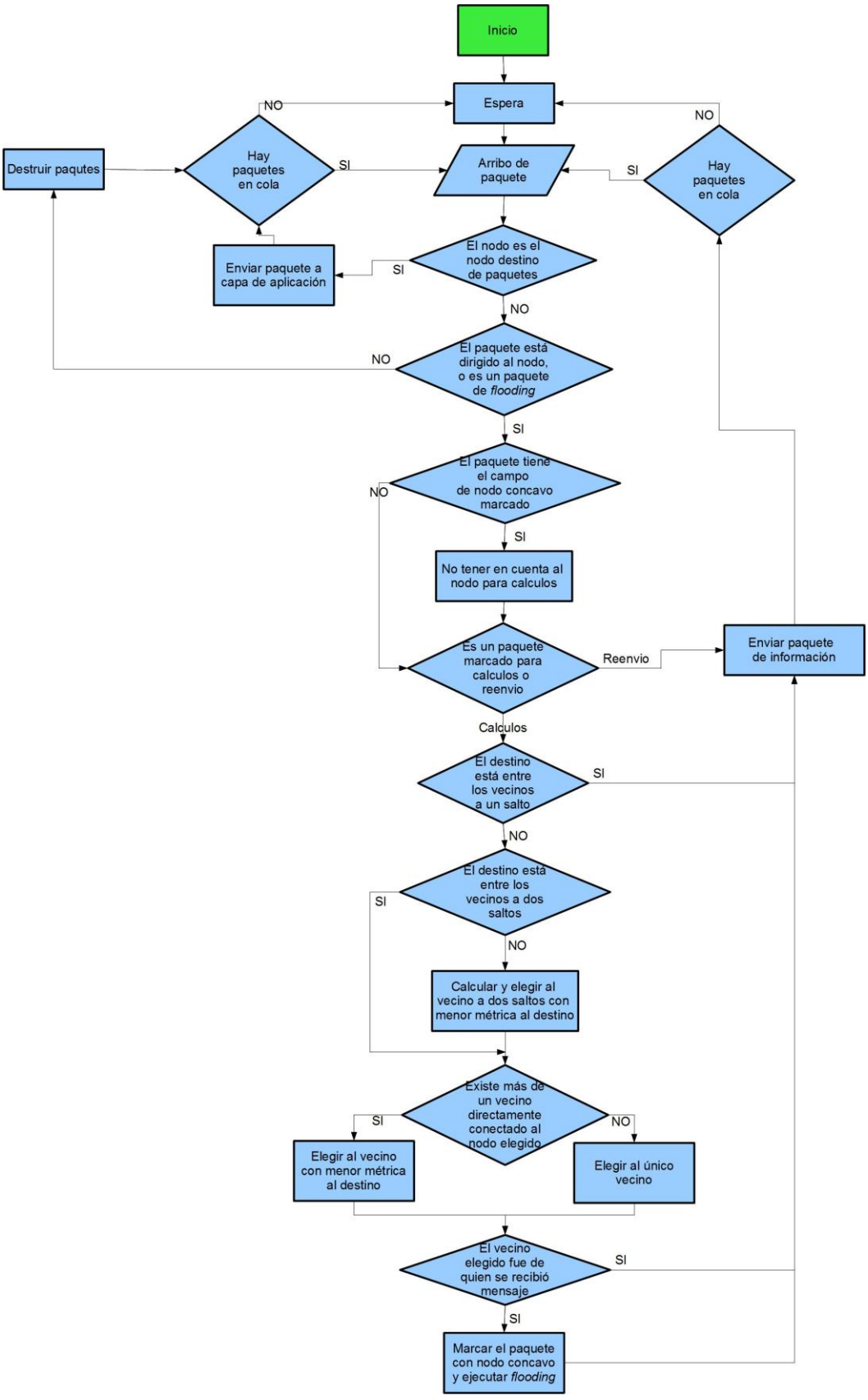
[18] Imagen tomada de http://www.vancouver.wsu.edu/fac/song/images/wsn.jpg

[19] Información tomada de http://www-staff.it.uts.edu.au/~simmonds/jitter.htm

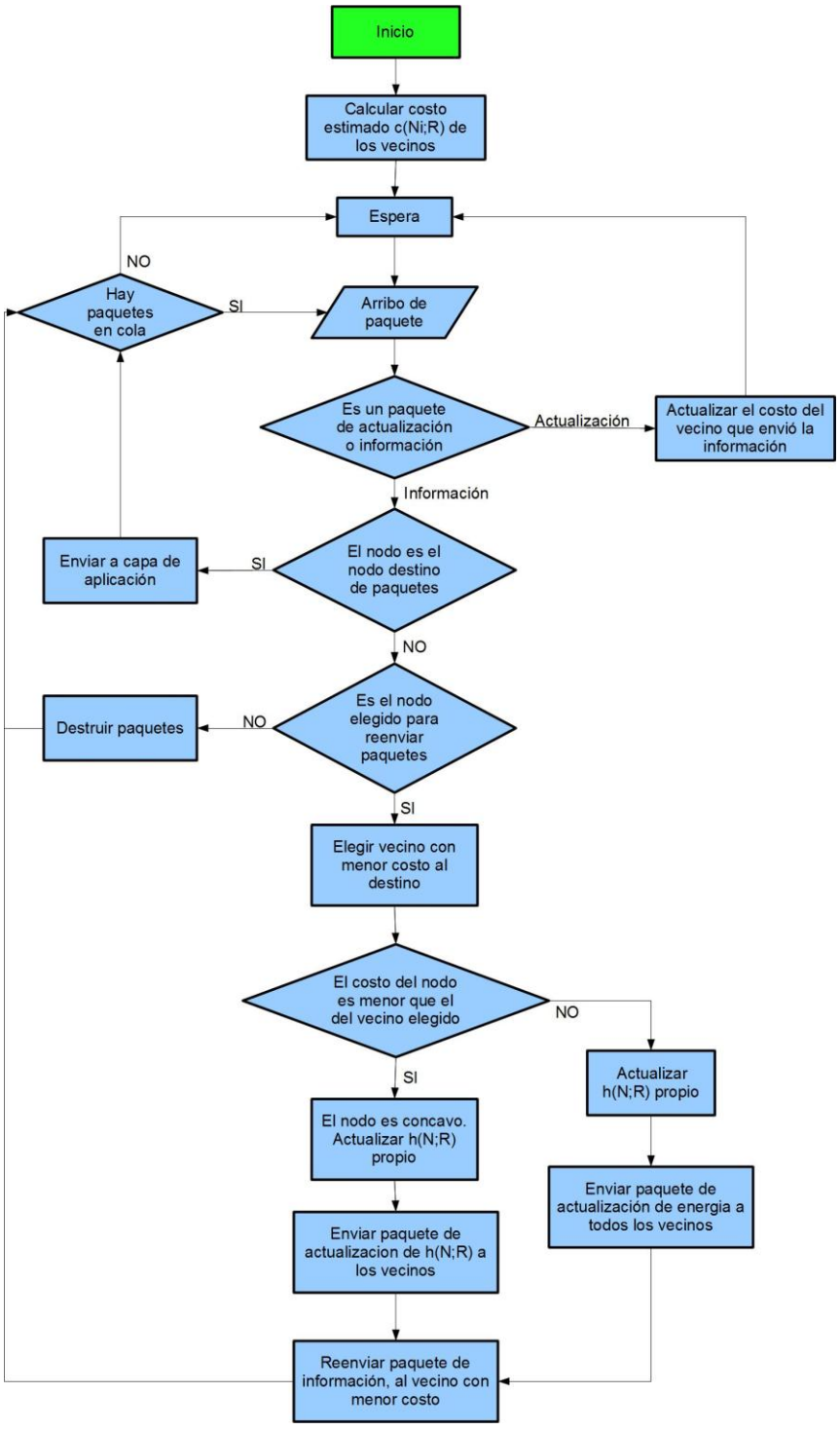
[20] Opnet Modeler Documentation > Modeler Reference > Modeling Concepts > Modeling overview

[21] Opnet Modeler Documentation > Basic Tutorials > Modeler Only Tutorials > Basic Processes

Anexo 1 Diagrama de flujo 2-f-GEDIR y 2-f-MFR

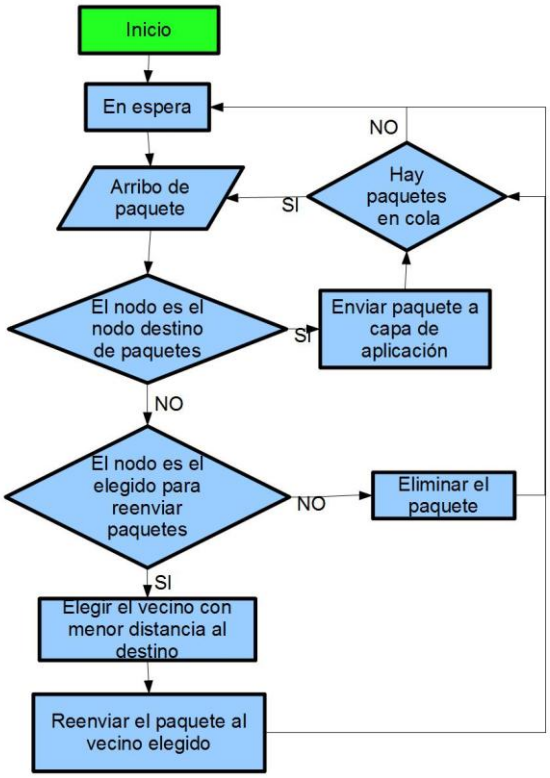


Anexo 2 Diagrama de flujo de GEAR

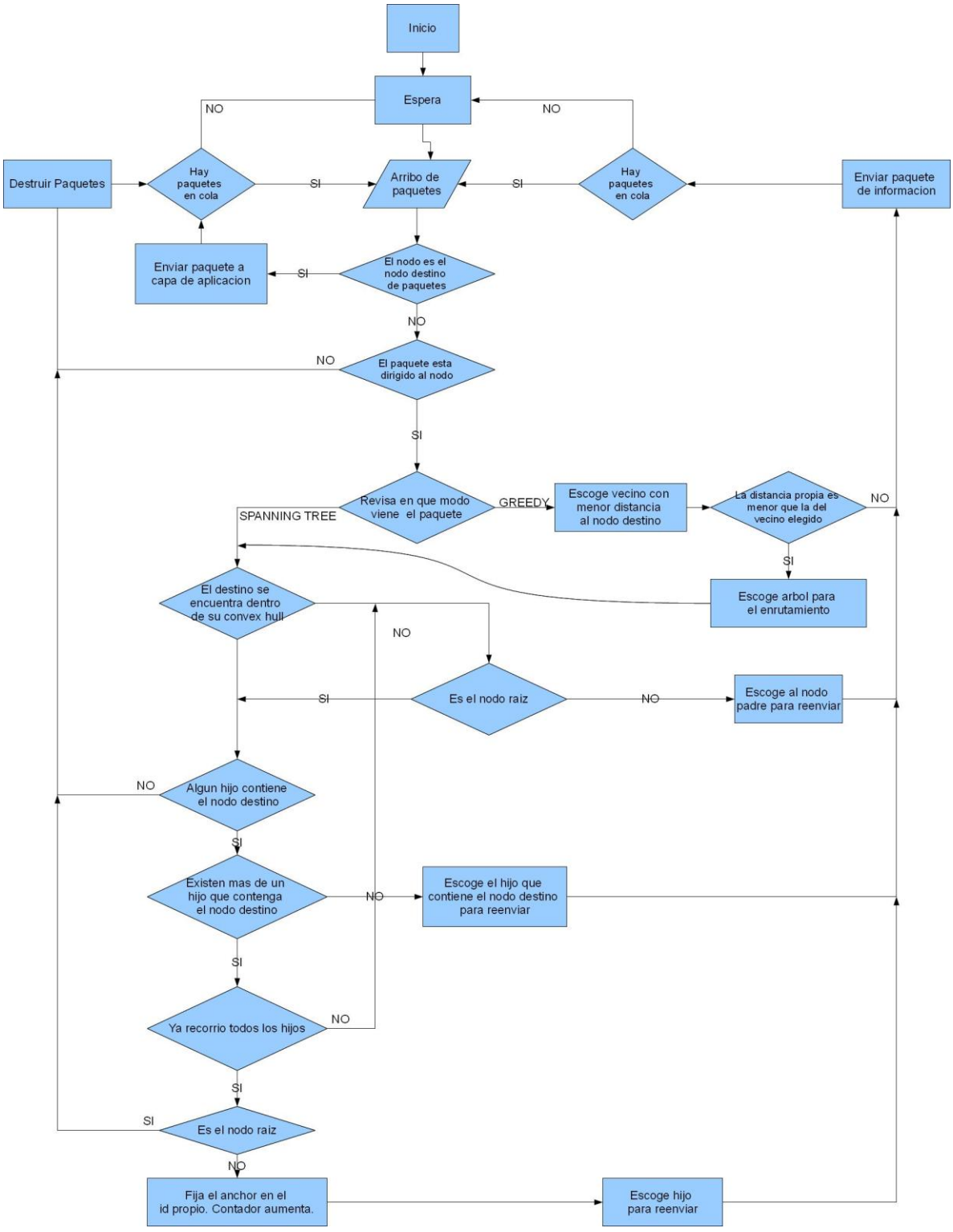


Anexo 3 Diagrama de flujo de GAF

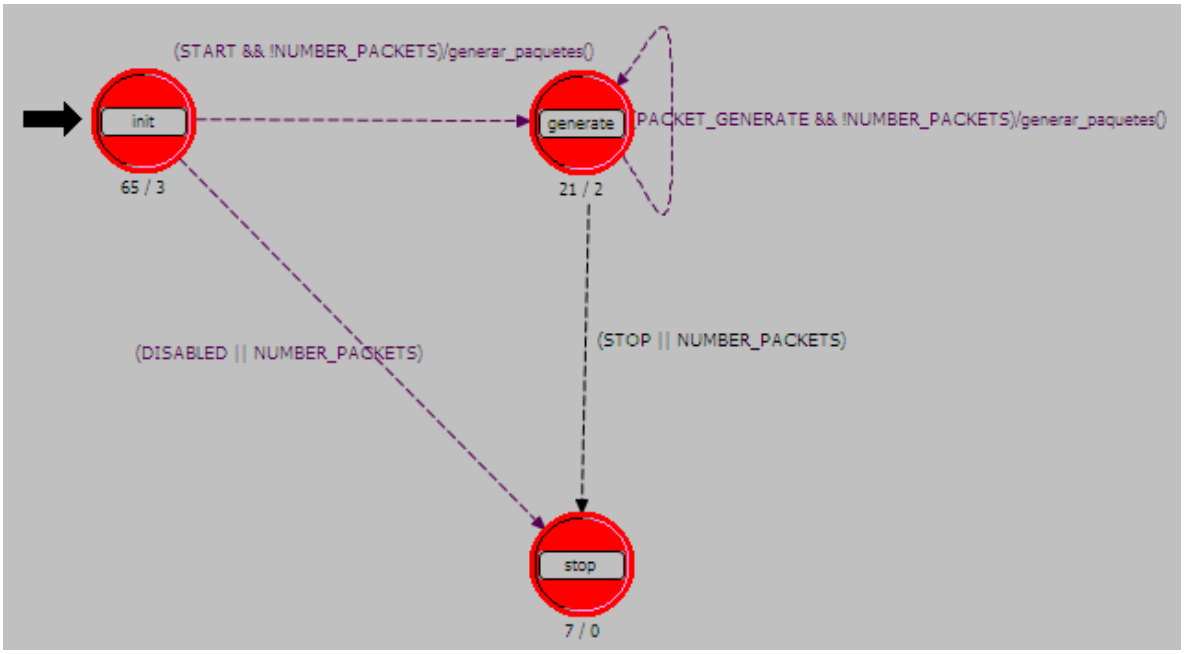
Anexo 4 Protocolo *Greedy*



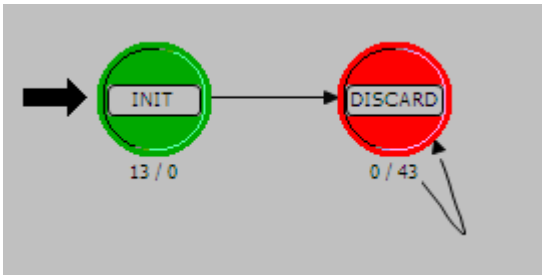
Anexo 5 Enrutamiento tipo Spanning Tree



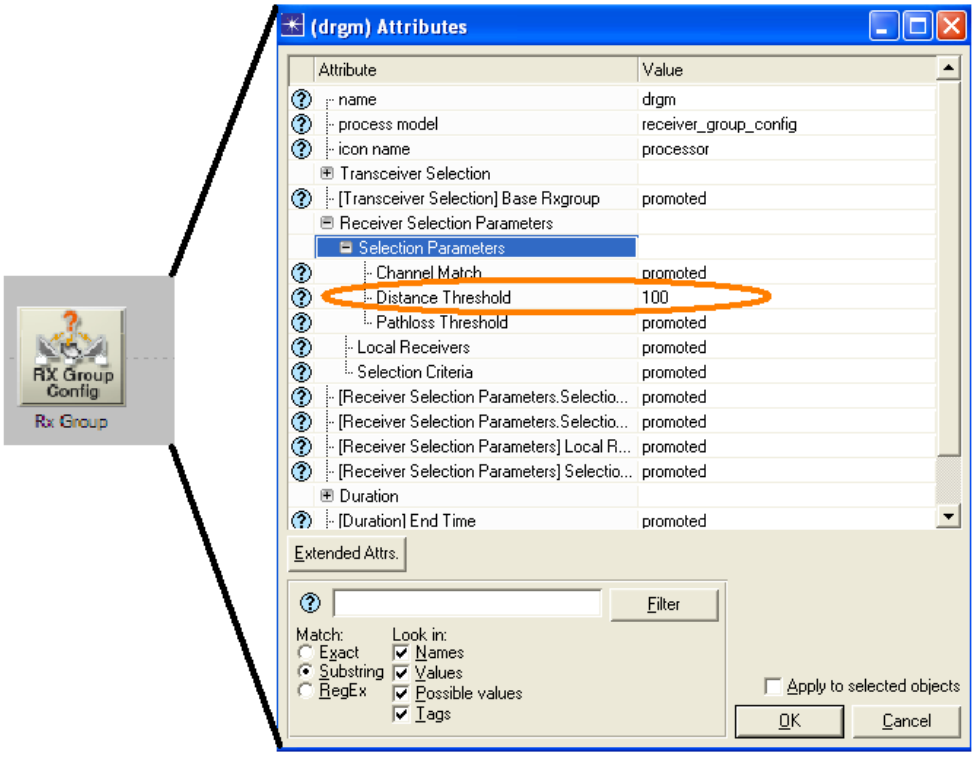
Anexo 6 Modelo de *"Simple source"* de Opnet



Anexo 7 Modelo de "sink" de Opnet



Anexo 8 Atributo *"Distance Threshold"* del RX_GROUP



Anexo 9 Pipeline

Ir al archivo Pipeline donde se explican los estados del pipeline que fueron usados en la simulación.

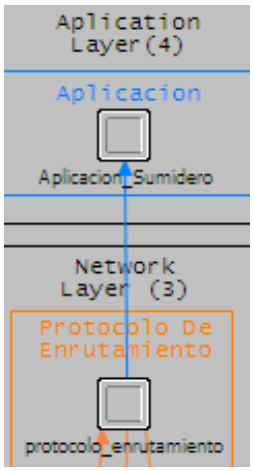
Anexo 10 Modelos y funciones de Opnet usados

En el archivo Especificaciones se explican los modelos que utiliza Opnet, el uso de listas y de las funciones principales de estas, el uso de paquetes y sus funciones, la manera de recolectar datos, entre otras funciones del simulador.

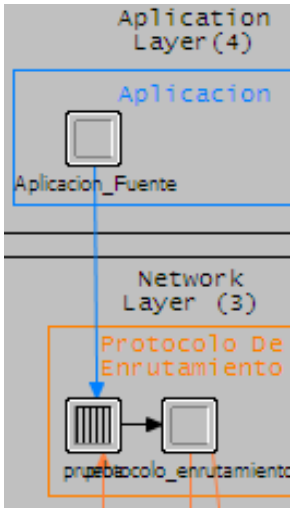
Anexo 11 Línea de tiempo de protocolos

En el archivo se relacionan los diferentes protocolos de tipo geográfico. Archivo Excel en la documentación "línea de tiempo portocolos"

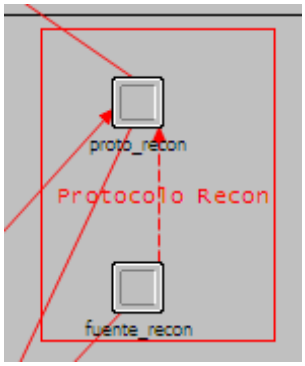
Anexo 12 Módulo *"Aplicacion_sumidero"*



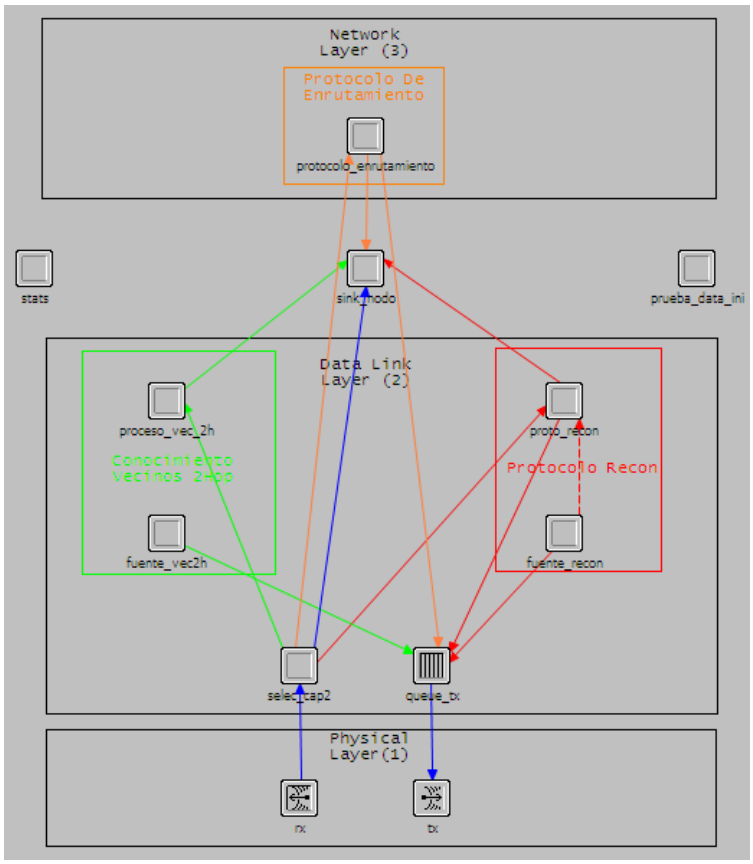
Anexo 13 Módulo "Aplicacion_Fuente"



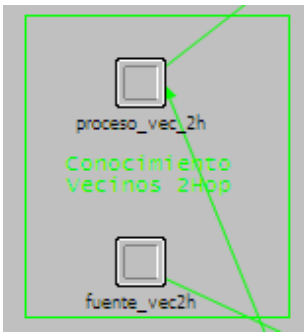
Anexo 14 Módulos del protocolo de reconocimiento



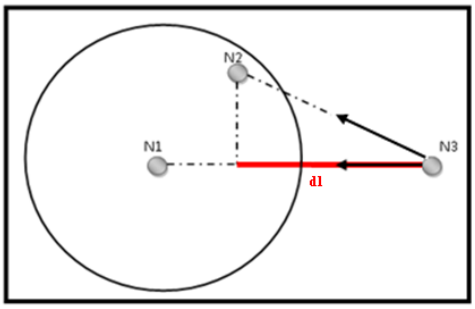
Anexo 15 Modelo de Nodo de los protocolos 2-f-GEDIR y 2-f-MFR



Anexo 16 Reconocimiento vecinos *2-hop*



Anexo 17 Progreso del nodo N2 al nodo N3



Anexo 18 Cálculo del progreso

$$progreso = d1 = \frac{\overline{N_3N_2} \cdot \overline{N_3N_1}}{\left\|\overline{N_3N_1}\right\|} \qquad (10)$$

$$\overline{N_3N_2} = (pos\_x_{N_3} - pos\_x_{N_2}, pos\_y_{N_3} - pos\_y_{N_2})$$

$$\overline{N_3N_2} = (pos\_x_{N_3N_2}, pos\_y_{N_3N_2}) \qquad (11)$$

$$\overline{N_3N_1} = (pos\_x_{N_3} - pos\_x_{N_1}, pos\_y_{N_3} - pos\_y_{N_1})$$

$$\overline{N_3N_1} = (pos\_x_{N_3N_1}, pos\_y_{N_3N_1}) \qquad (12)$$

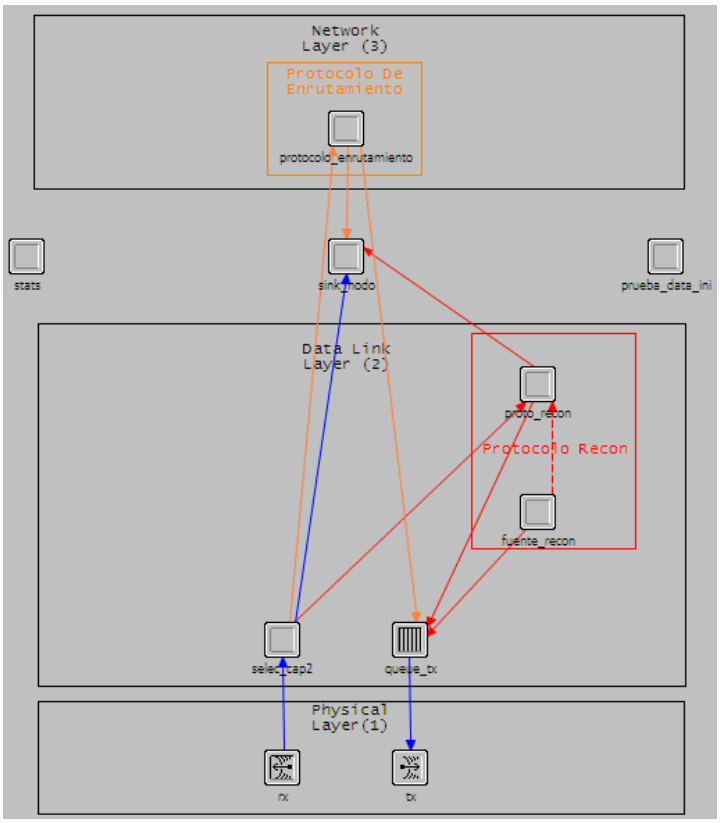
$$\overline{N_3N_2} \cdot \overline{N_3N_1} = (pos\_x_{N_3N_2} \cdot pos\_x_{N_3N_1} + pos\_y_{N_3N_2} \cdot pos\_y_{N_3N_1}) \qquad (13)$$

$$\left\|\overline{N_3N_1}\right\| = \sqrt{(pos\_x_{N_3} - pos\_x_{N_1})^2 + (pos\_y_{N_3} - pos\_y_{N_1})^2} \qquad (14)$$

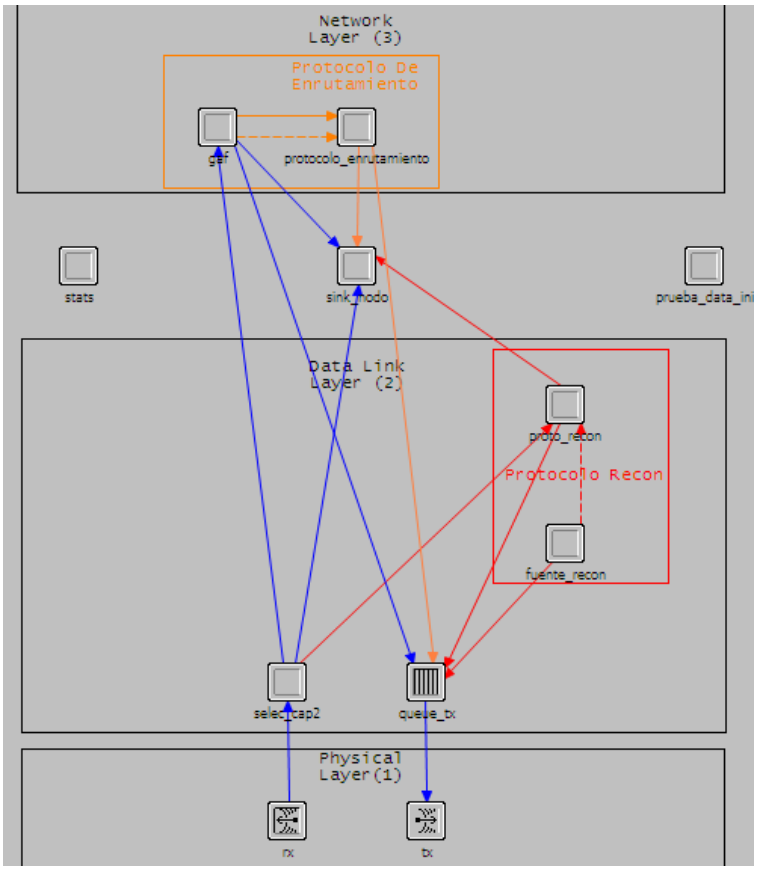
de (13) *y* (14) $progreso = d1 = \frac{\overline{N_3N_2} \cdot \overline{N_3N_1}}{\left\|\overline{N_3N_1}\right\|}$

$$= \frac{(pos\_x_{N_3N_2} \cdot pos\_x_{N_3N_1} + pos\_y_{N_3N_2} \cdot pos\_y_{N_3N_1})}{\sqrt{(pos\_x_{N_3} - pos\_x_{N_1})^2 + (pos\_y_{N_3} - pos\_y_{N_1})^2}} \qquad (15)$$

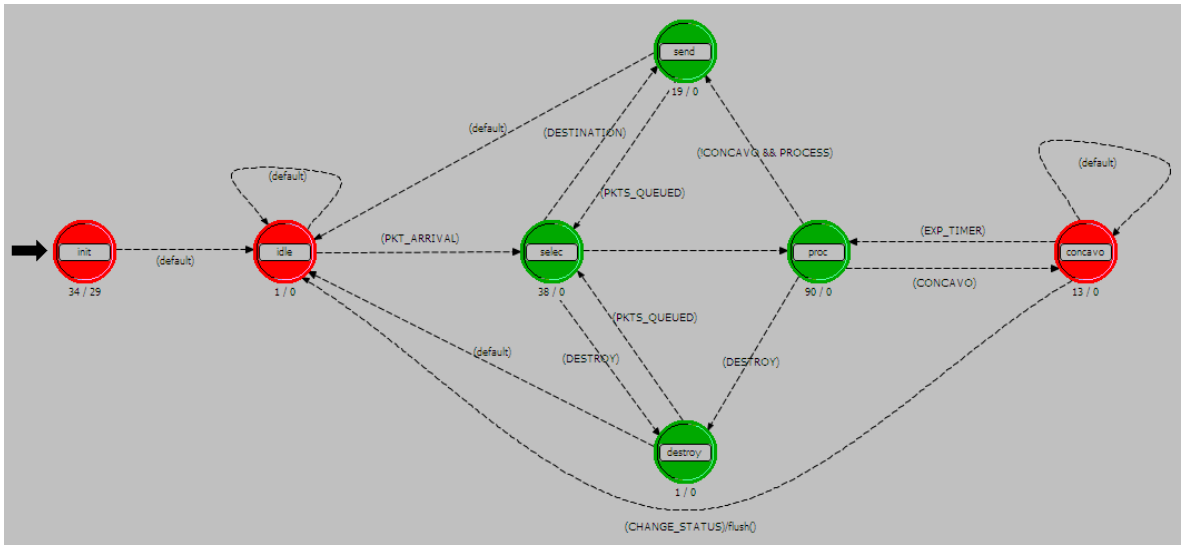
Anexo 19 Modelo de nodo de GEAR



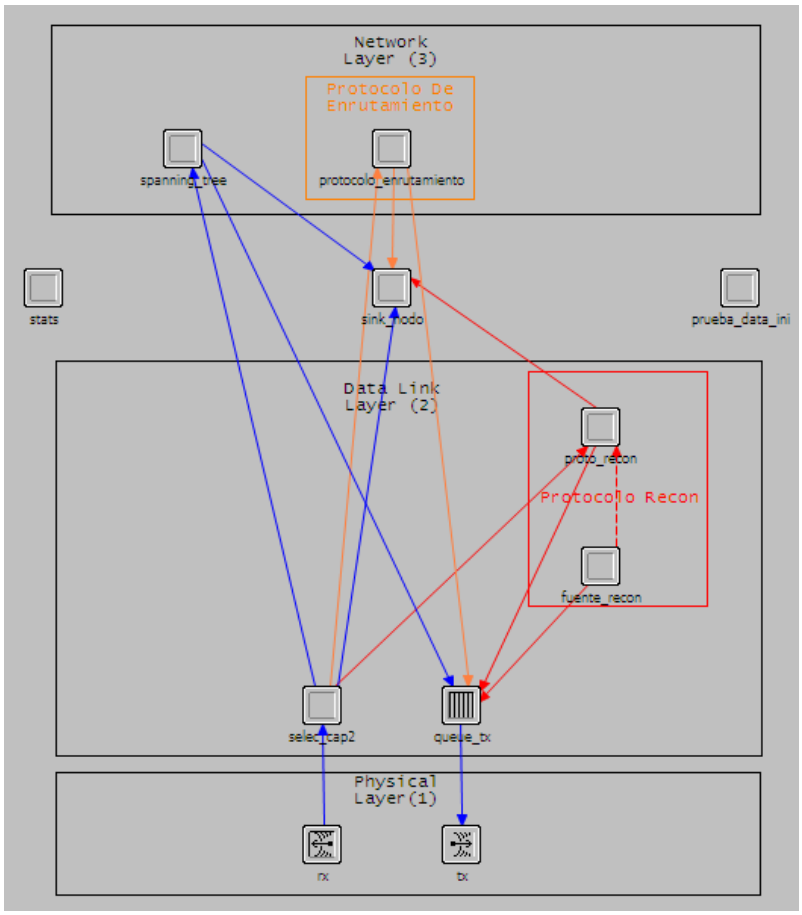
Anexo 20 Modelo de nodo GAF



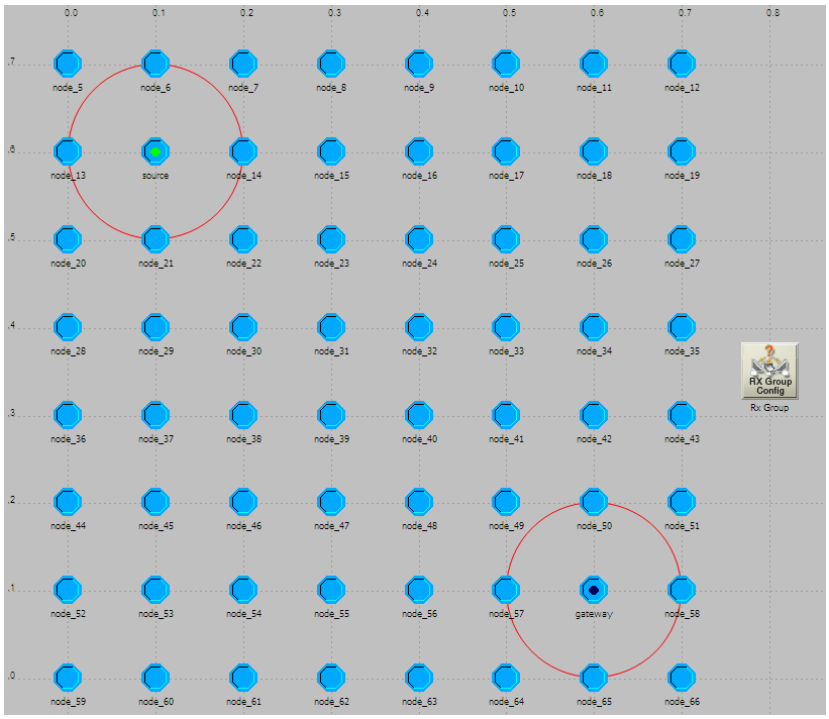
Anexo 21 Modelo de proceso del protocolo de enrutamiento en GAF



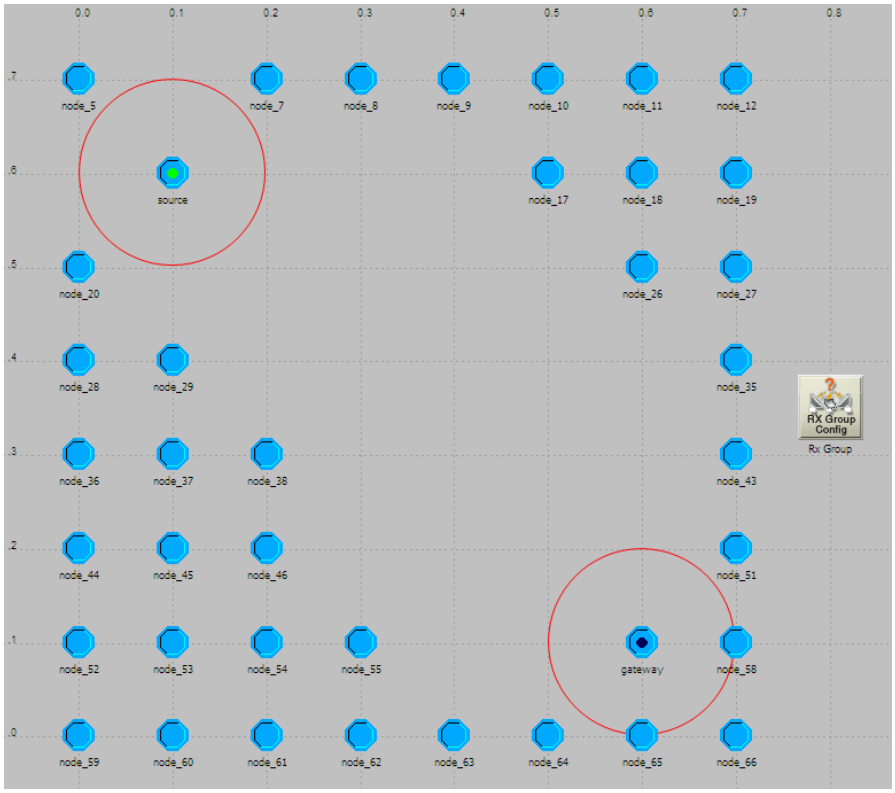
Anexo 22 Modelo de nodo de GDSTR



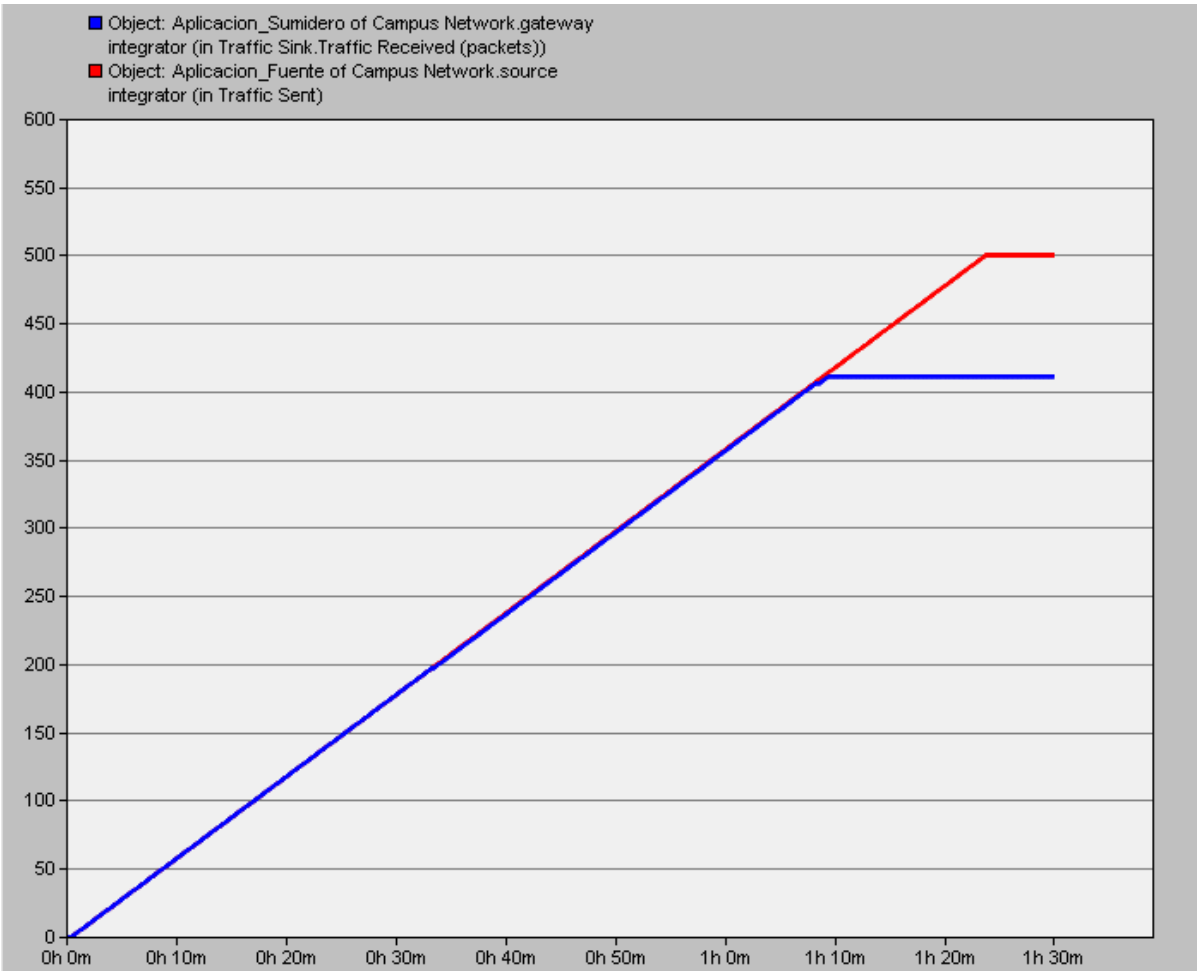
Anexo 23 Escenario determinístico 2-f-GEDIR, 2-f-MFR, GEAR y GDSTR



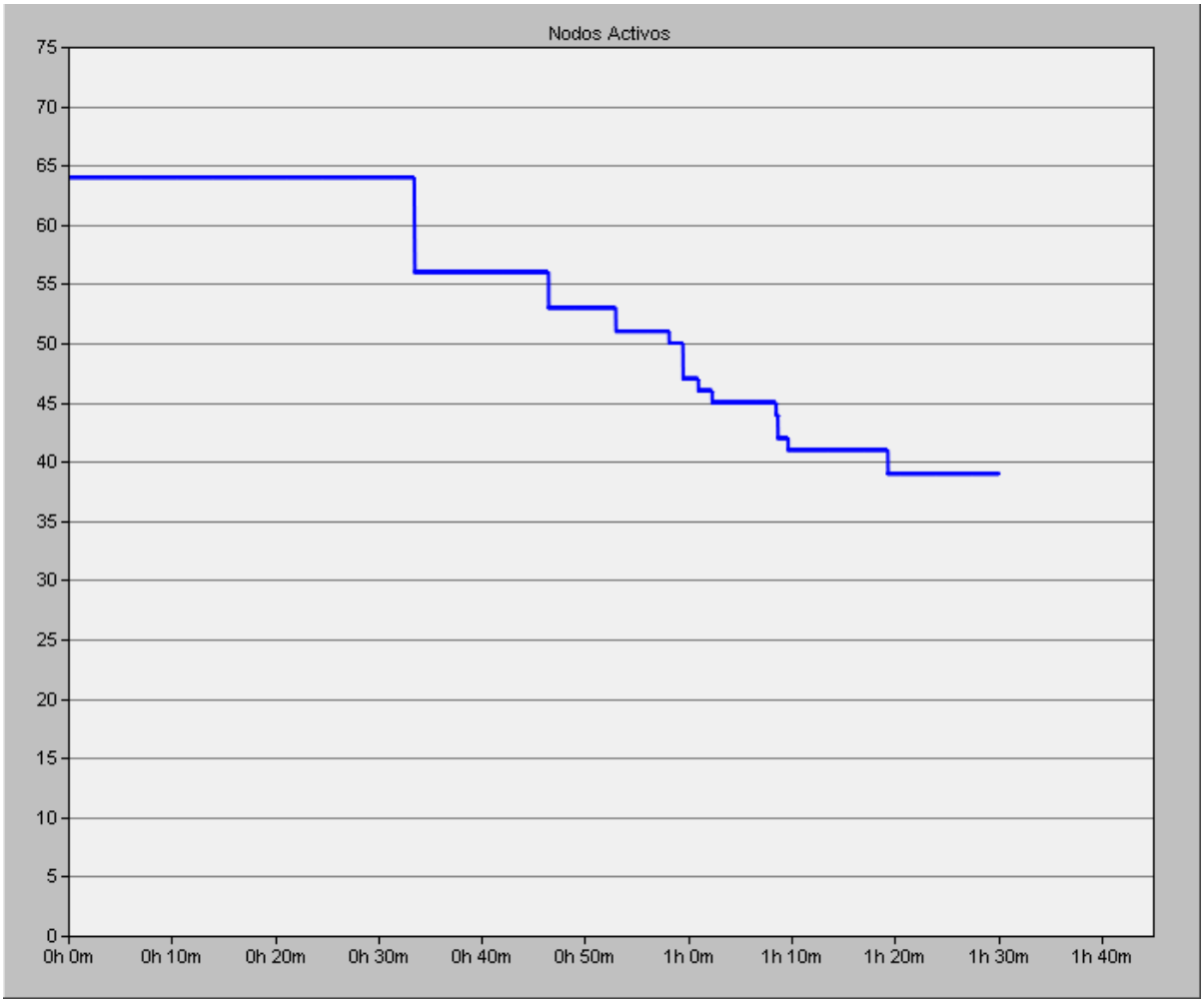
Anexo 24 Escenario 2-f-GEDIR nodos encendidos



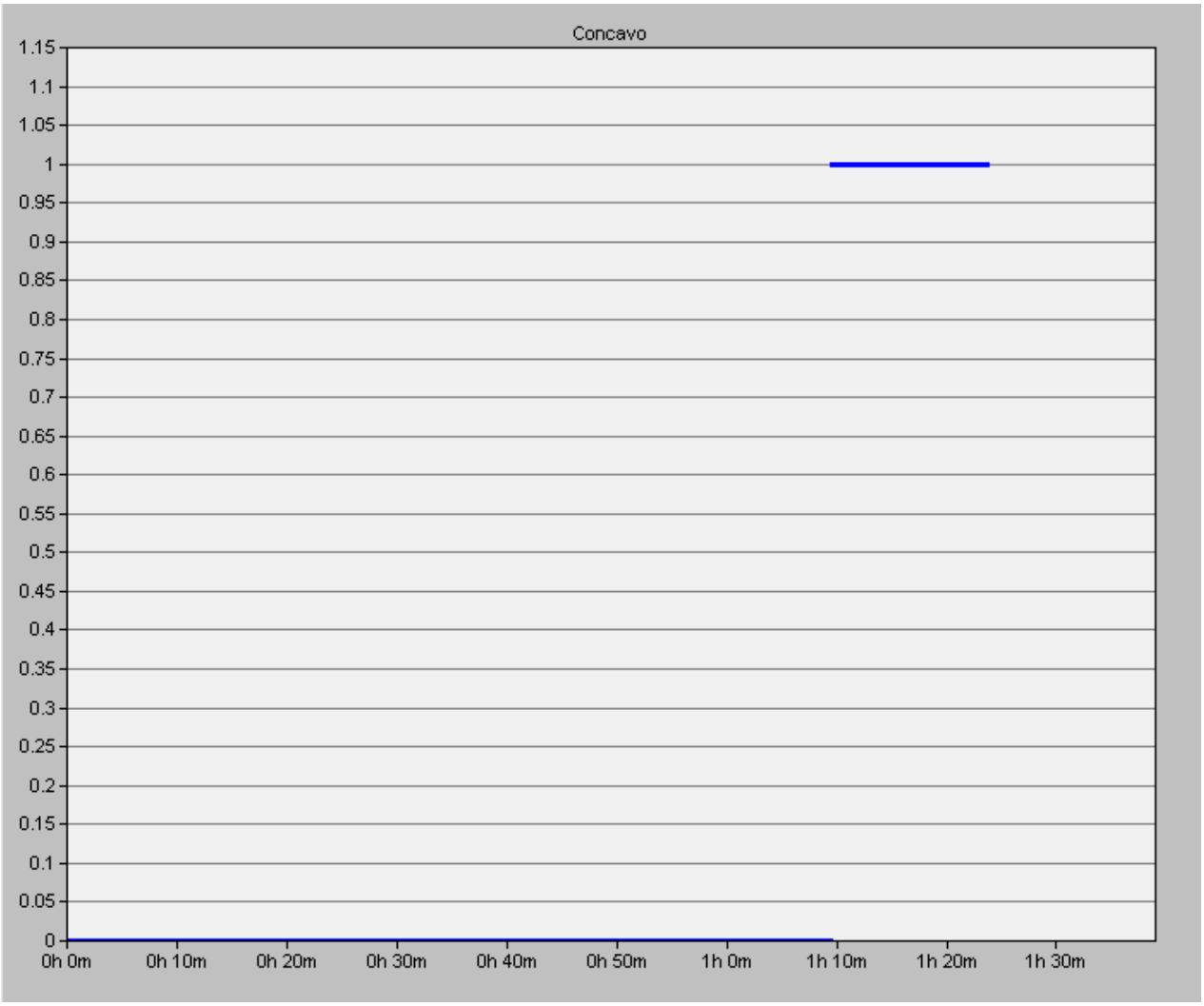
Anexo 25 Paquetes recibidos y paquetes generados 2-f-GEDIR



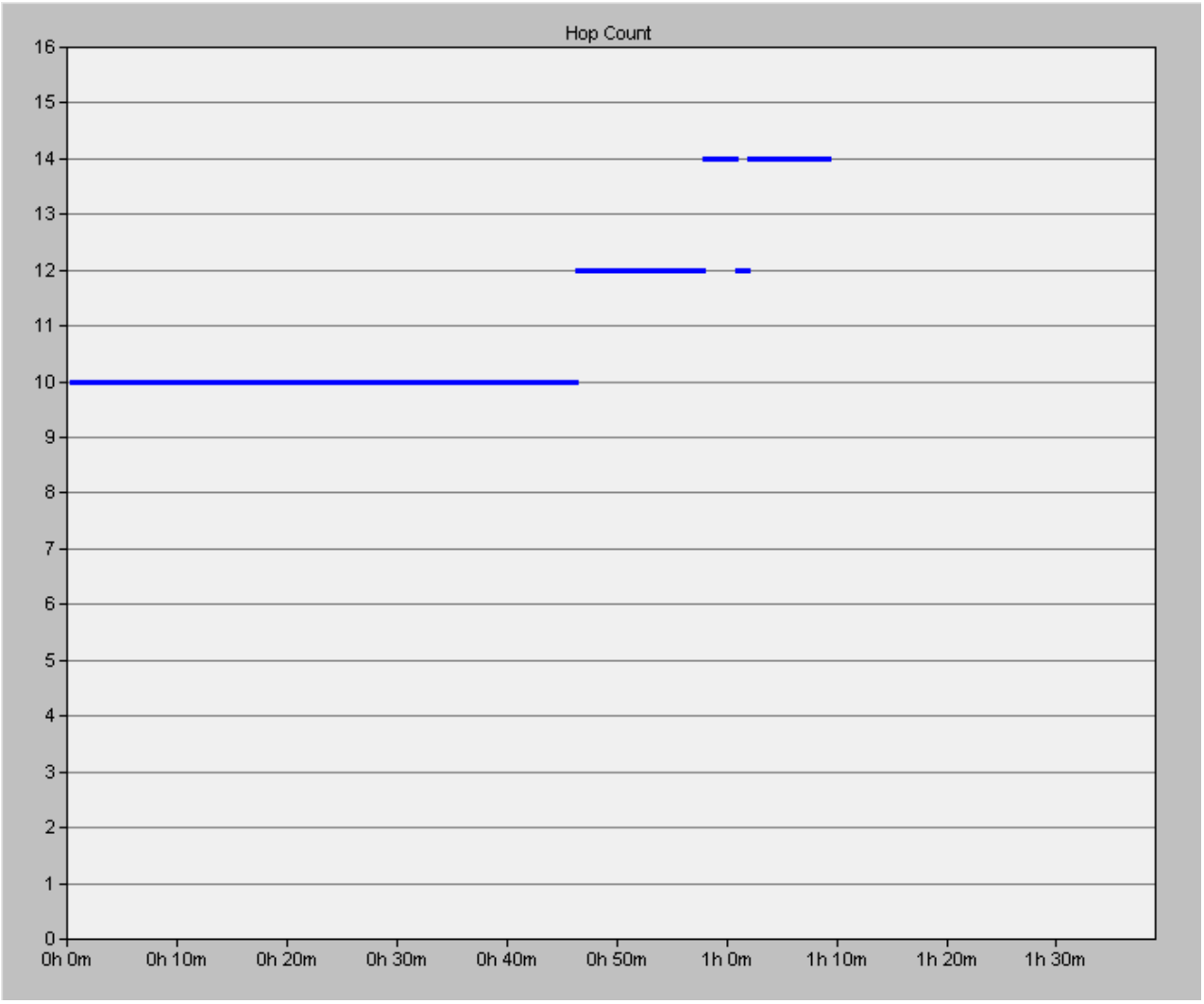
Anexo 26 Nodos Activos 2-f-GEDIR



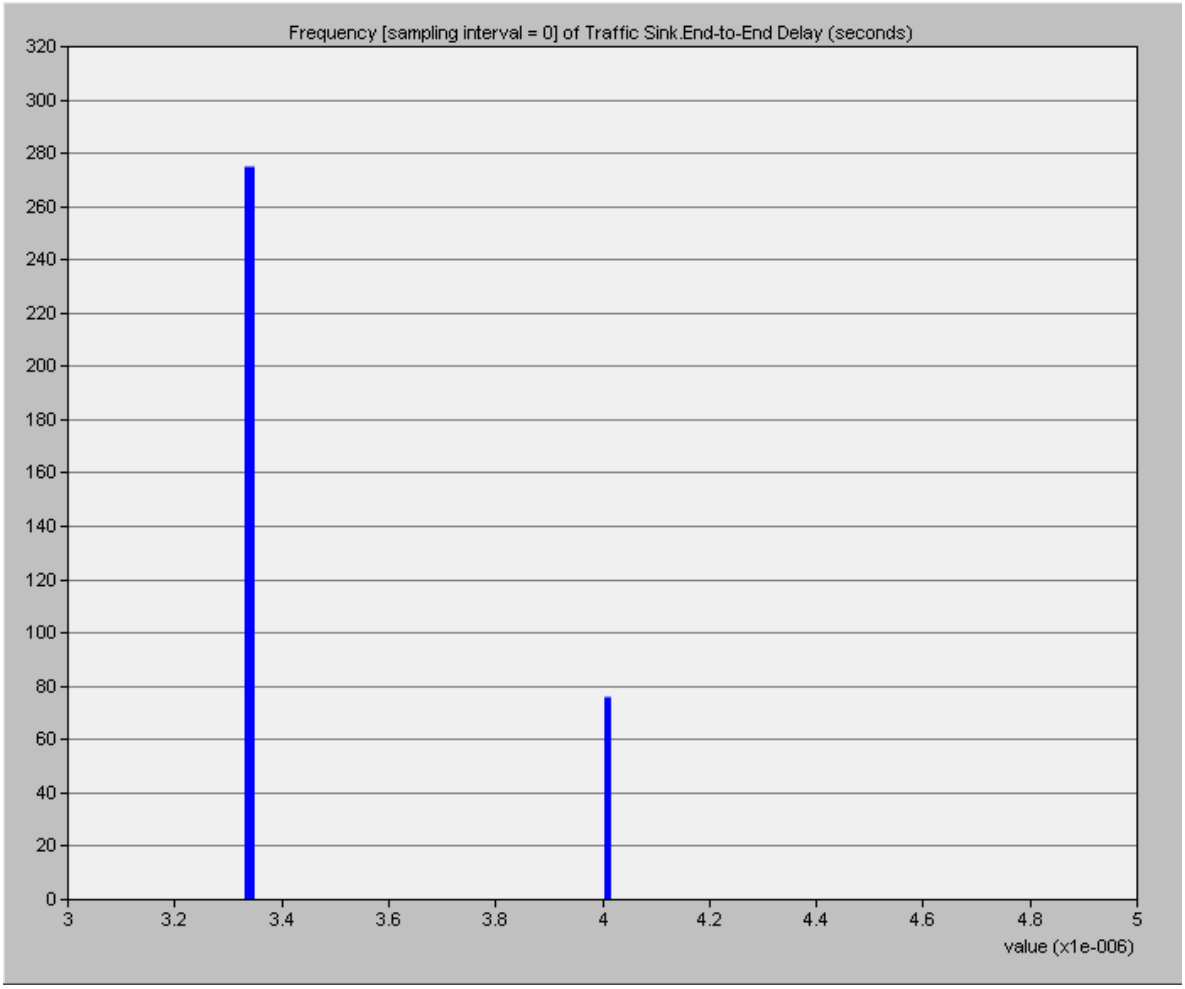
Anexo 27 Paquetes en nodos Cóncavo 2-f-GEDIR



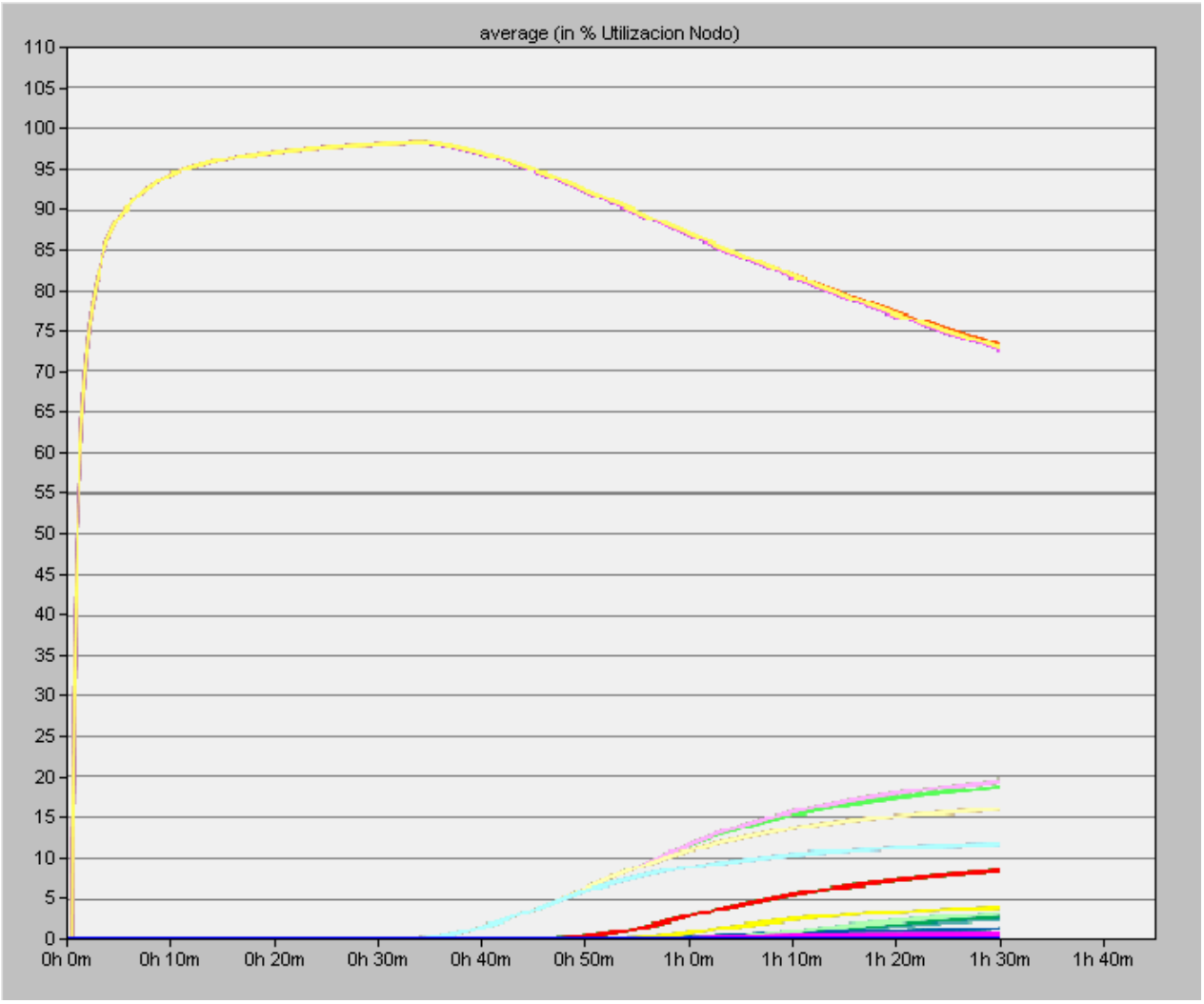
Anexo 28 Número de saltos al destino 2-f-GEDIR



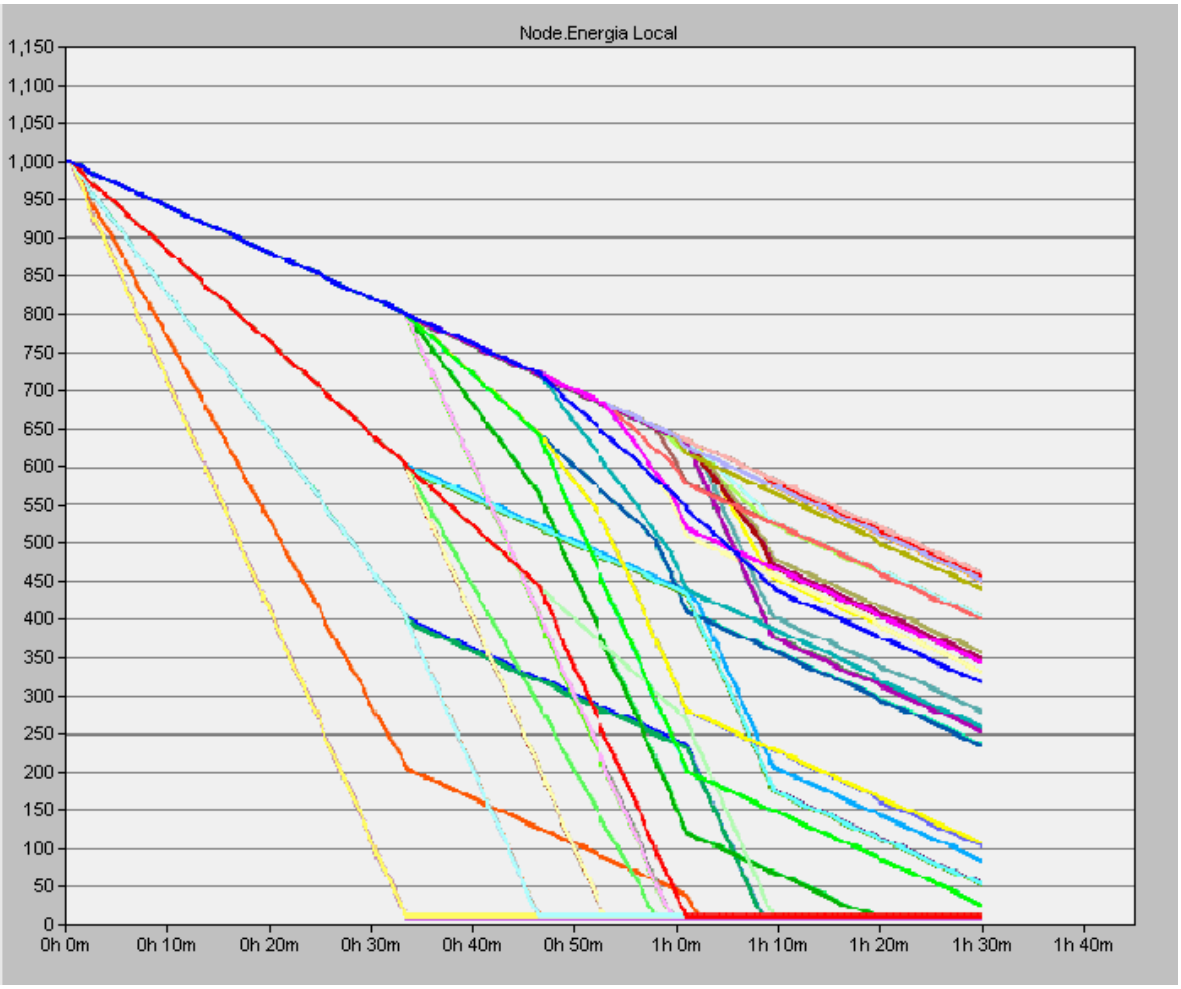
Anexo 29 *End to end delay* 2-f-GEDIR



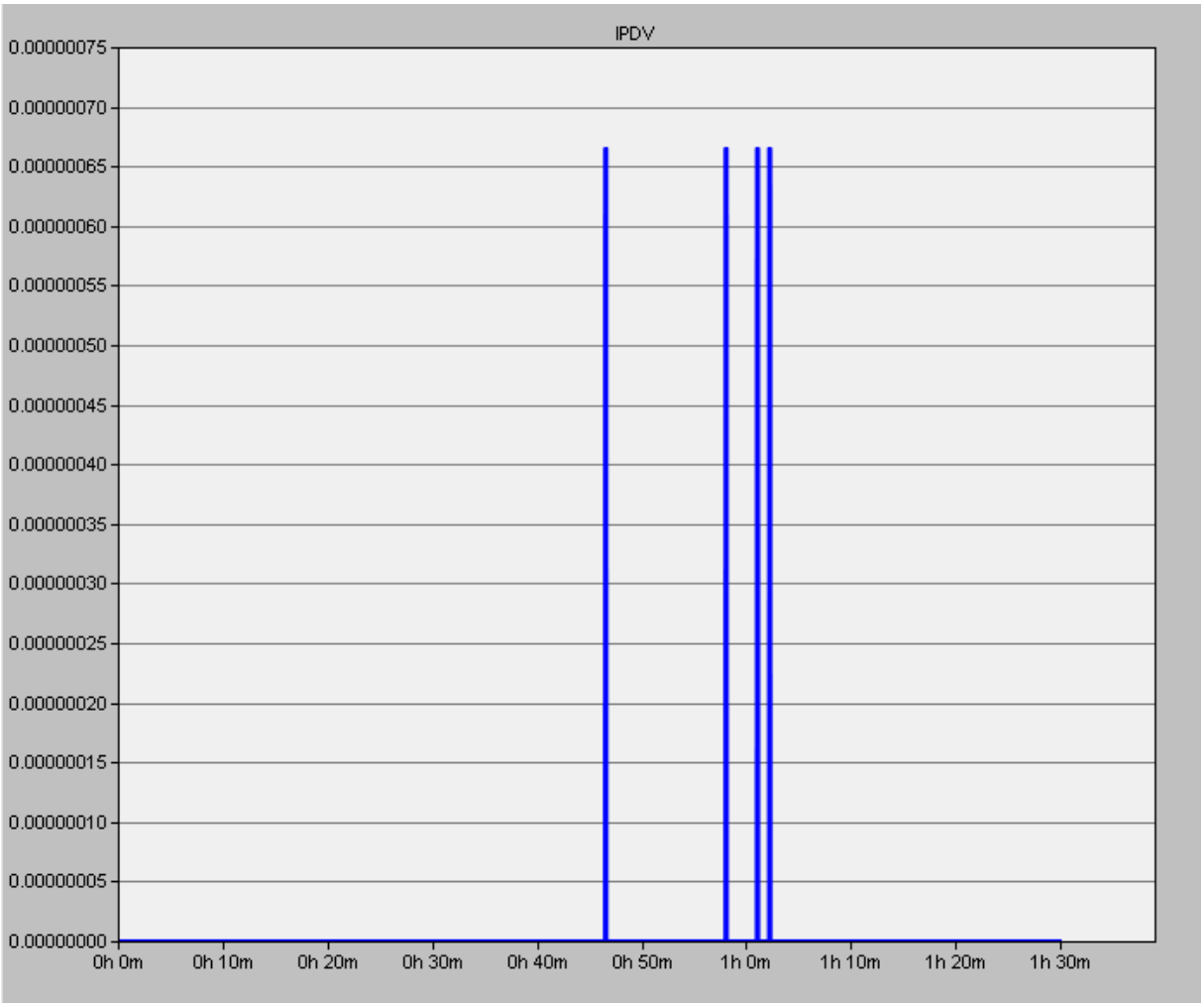
Anexo 30 Porcentaje promedio de utilización de los nodos 2-f-GEDIR



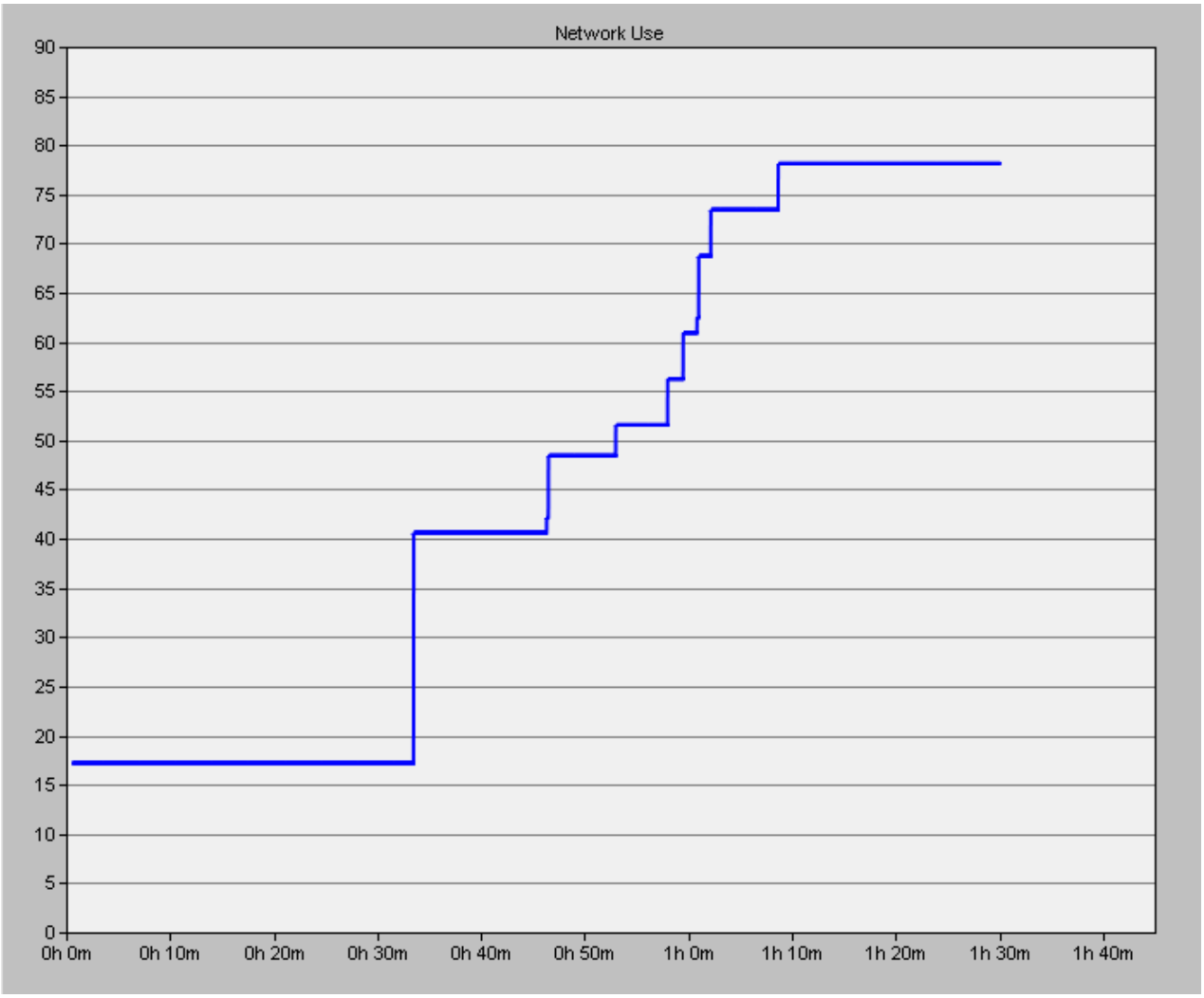
Anexo 31 Energía consumida por nodo 2-f-GEDIR



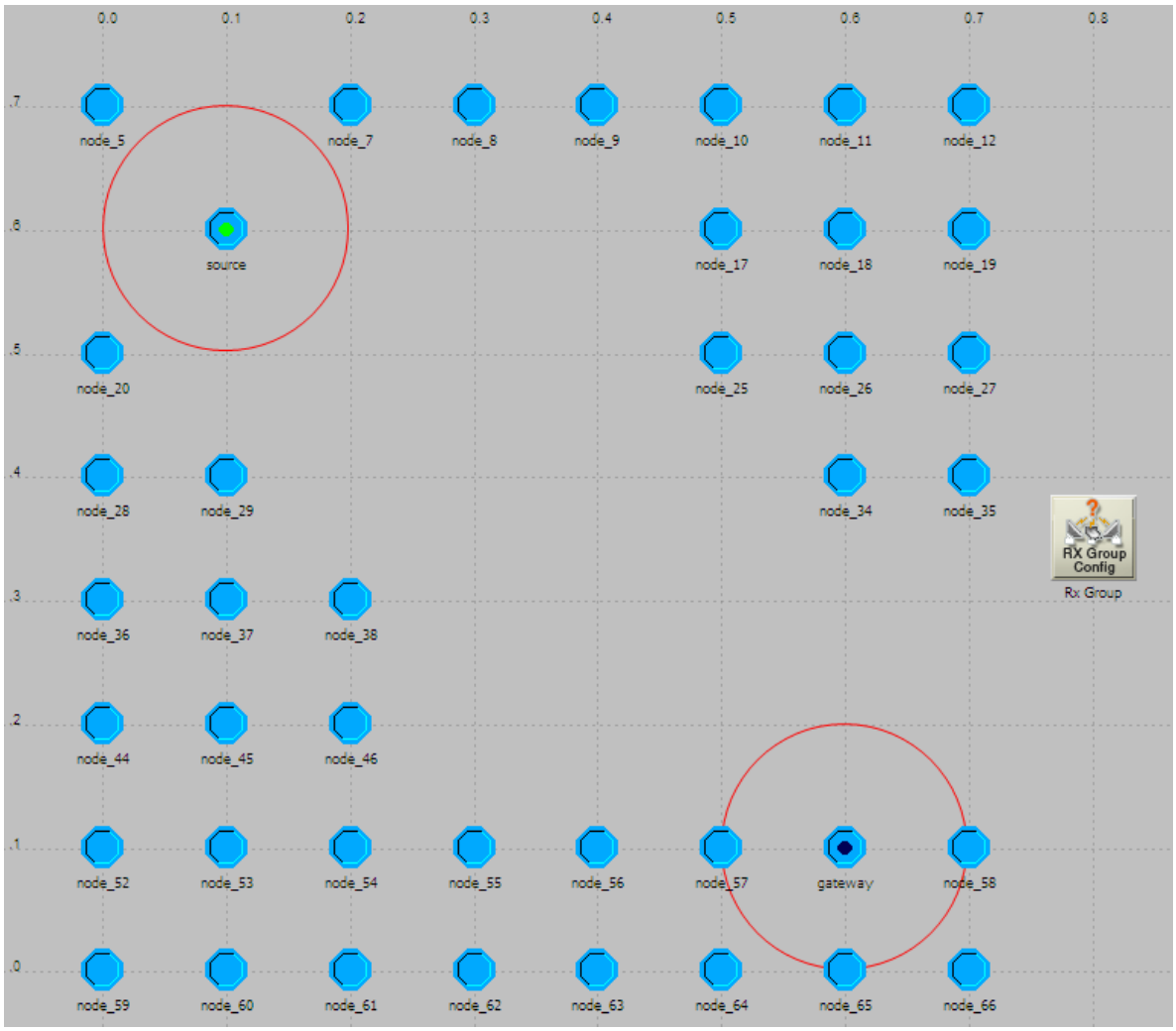
Anexo 32 IPDV 2-f-GEDIR



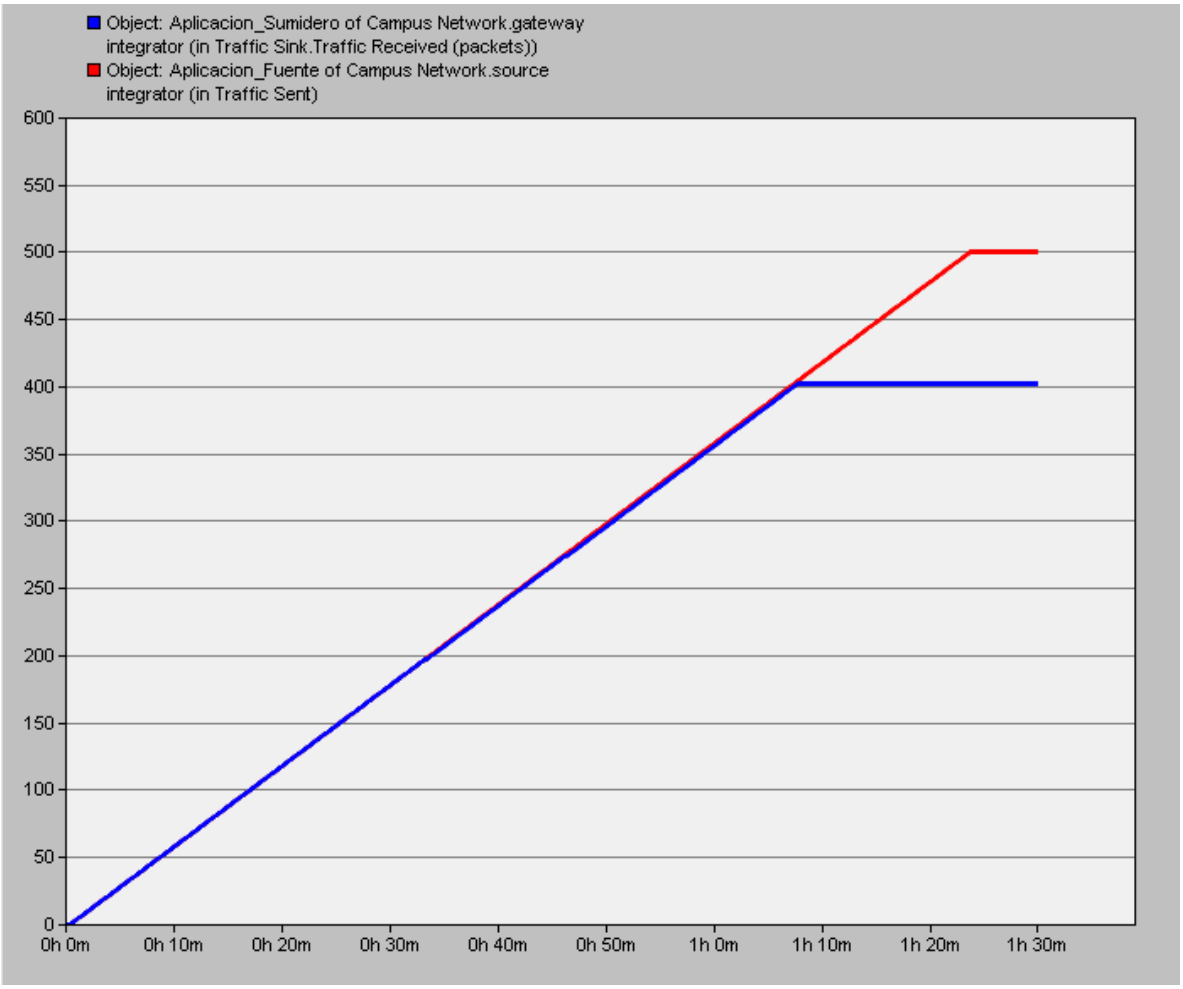
Anexo 33 Porcentaje de utilización de la red 2-f-GEDIR



Anexo 34 Escenario 2-f-MFR nodos encendidos



Anexo 35 Paquetes generados y paquetes recibidos 2-f-MFR



Anexo 36 Razón por la que 2-f-GEDIR y 2-f-MFR eligen diferentes nodos[11]

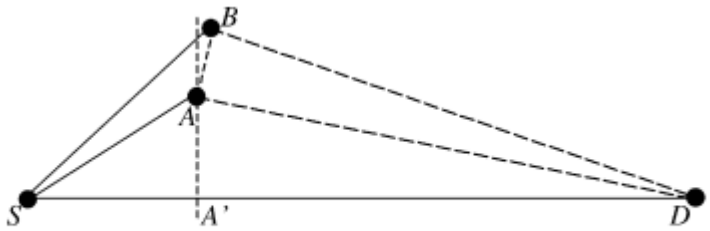
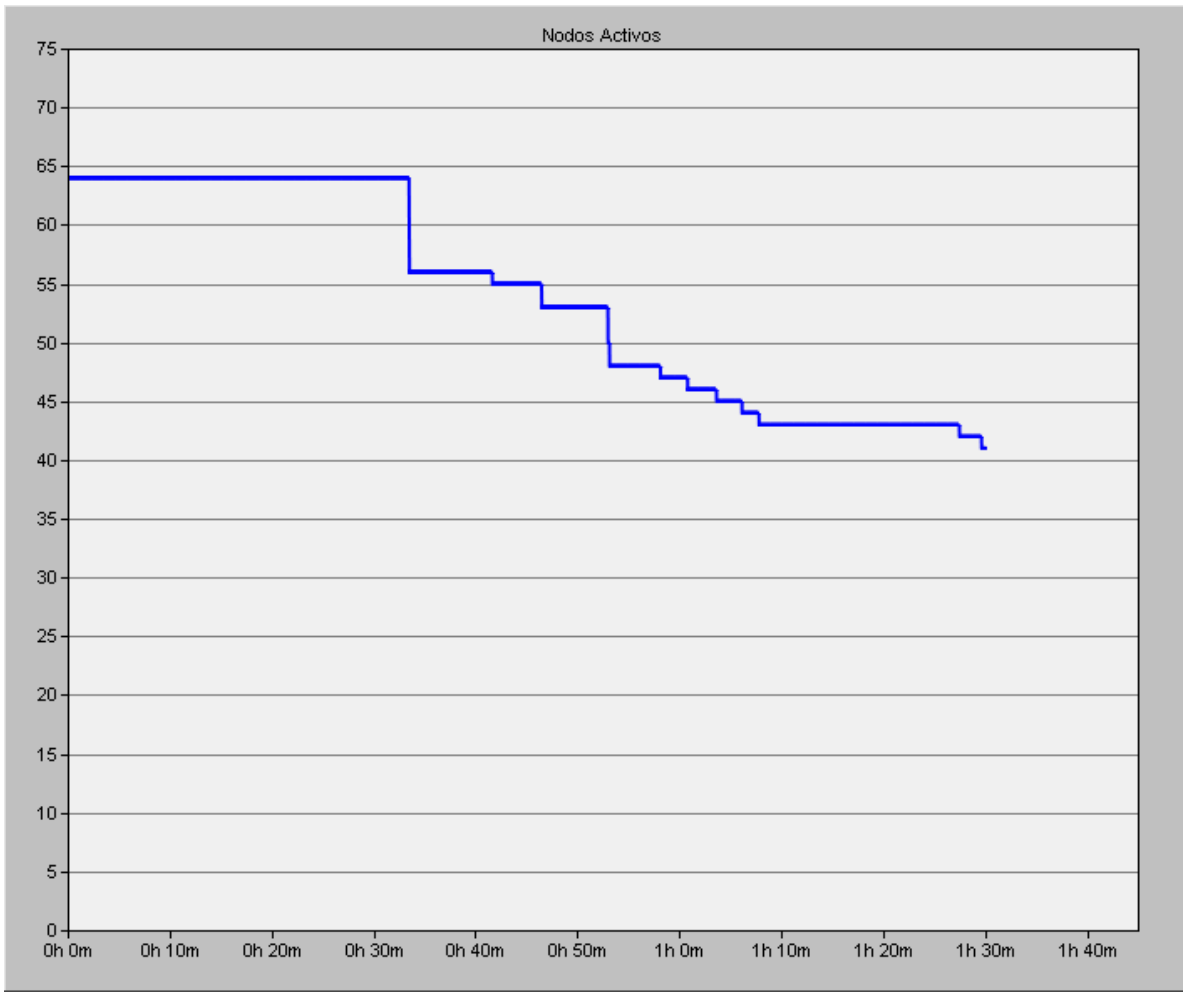


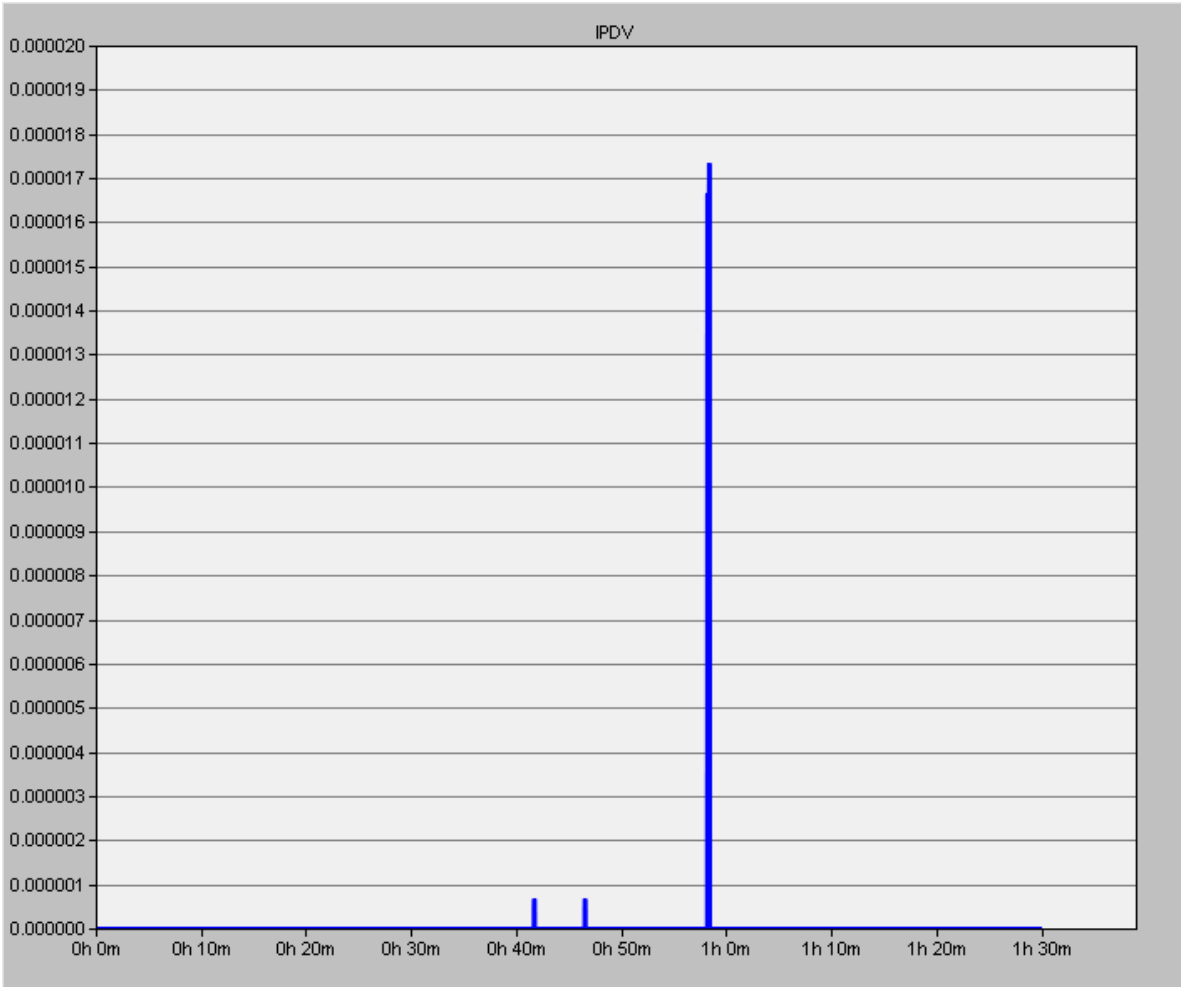
Fig. 5. *GEDIR* and *MFR* may choose different nodes.

[11] Imagen tomada de [10]

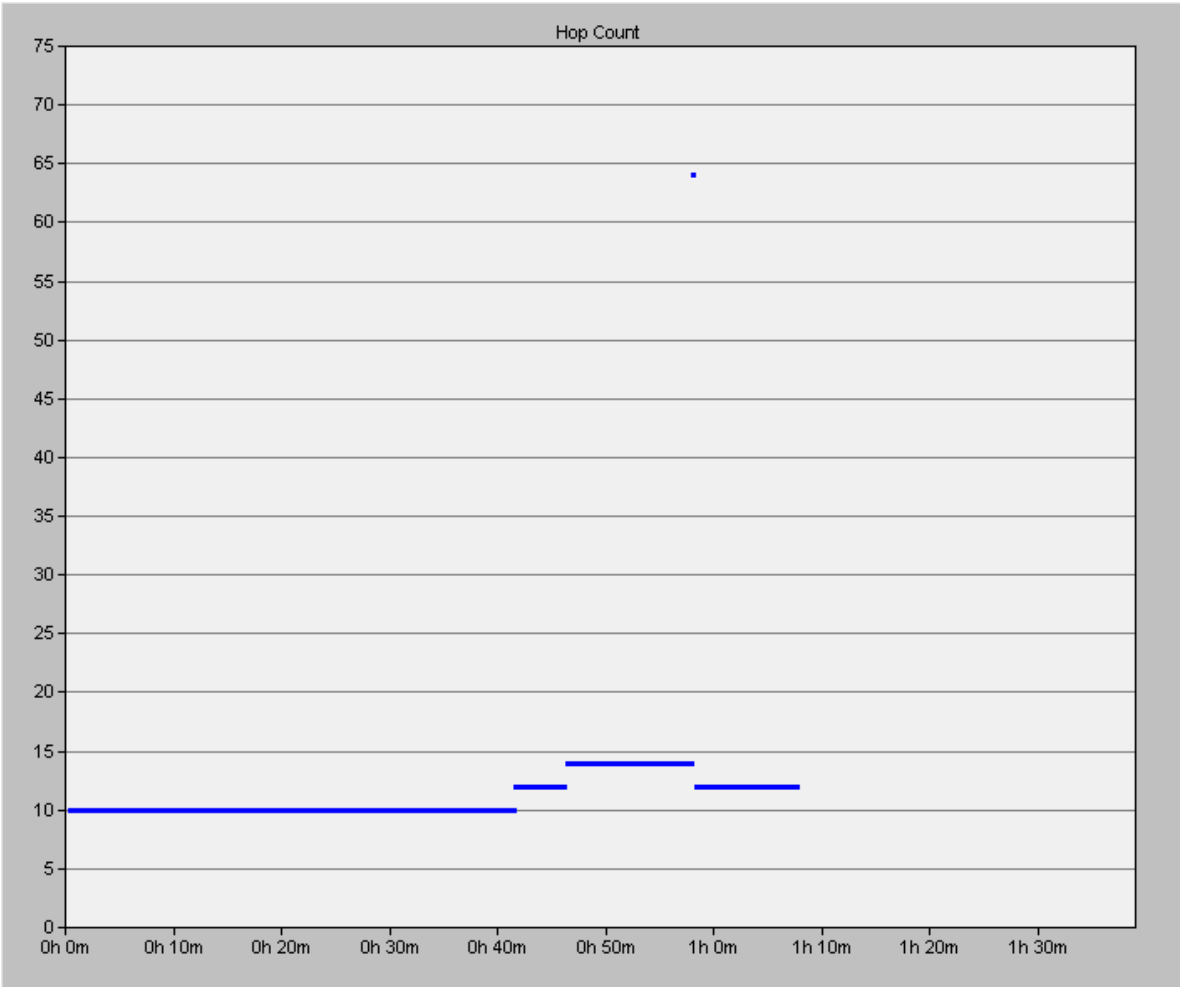
Anexo 37 Nodos Activos 2-f-MFR



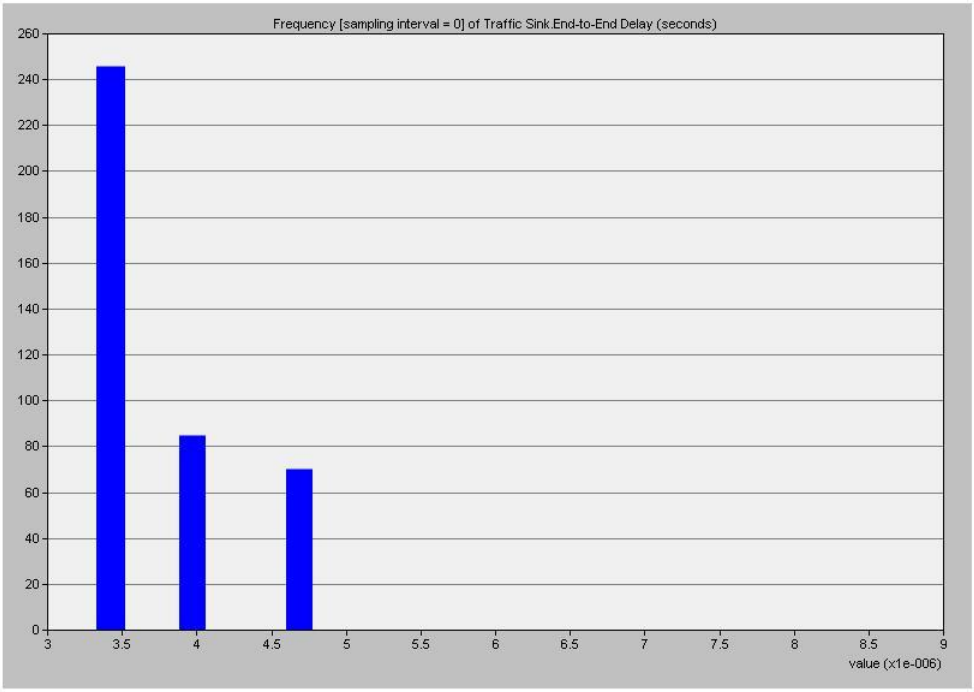
Anexo 38 IPDV 2-f-MFR



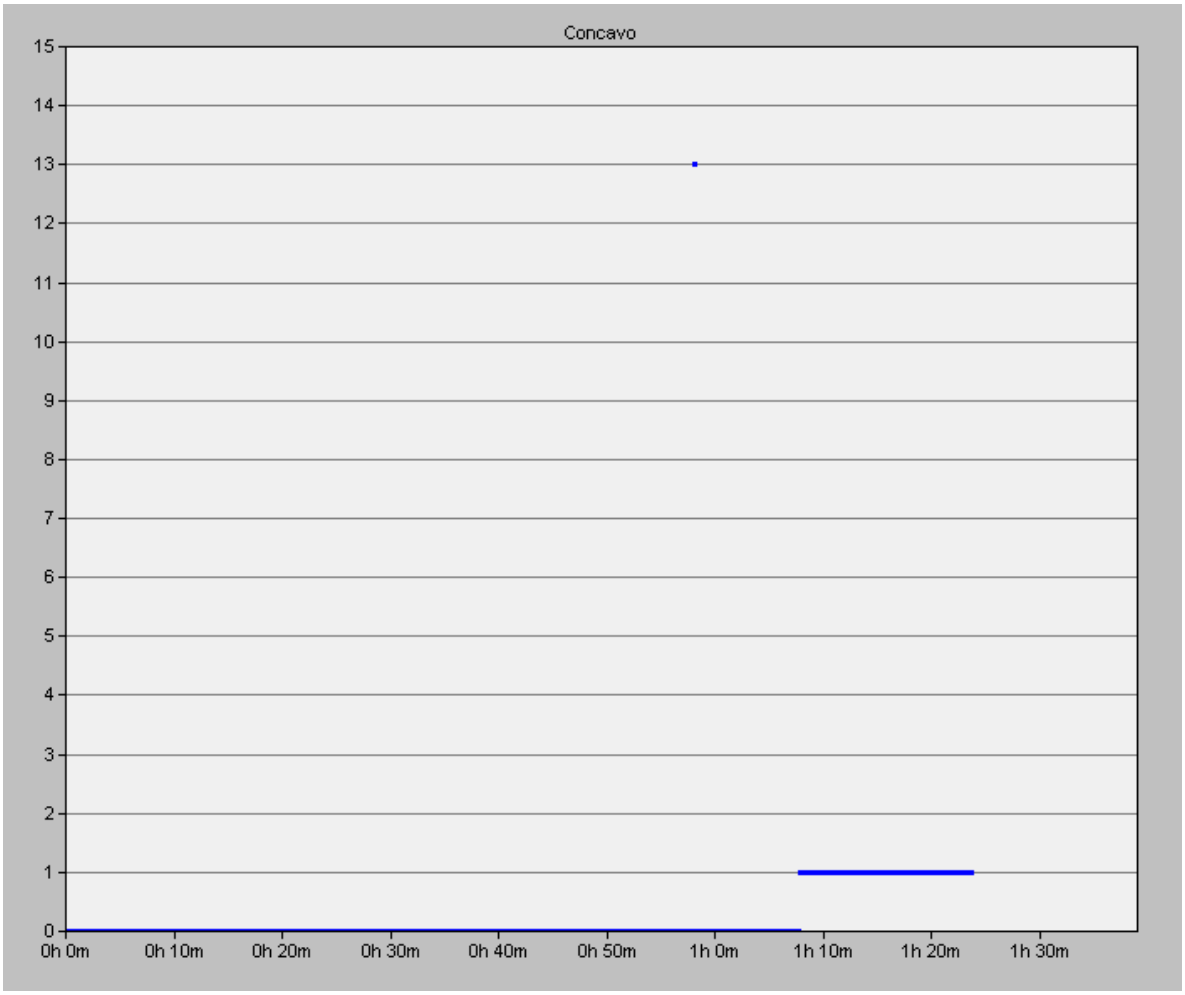
Anexo 39 *Hop count* 2-f-MFR



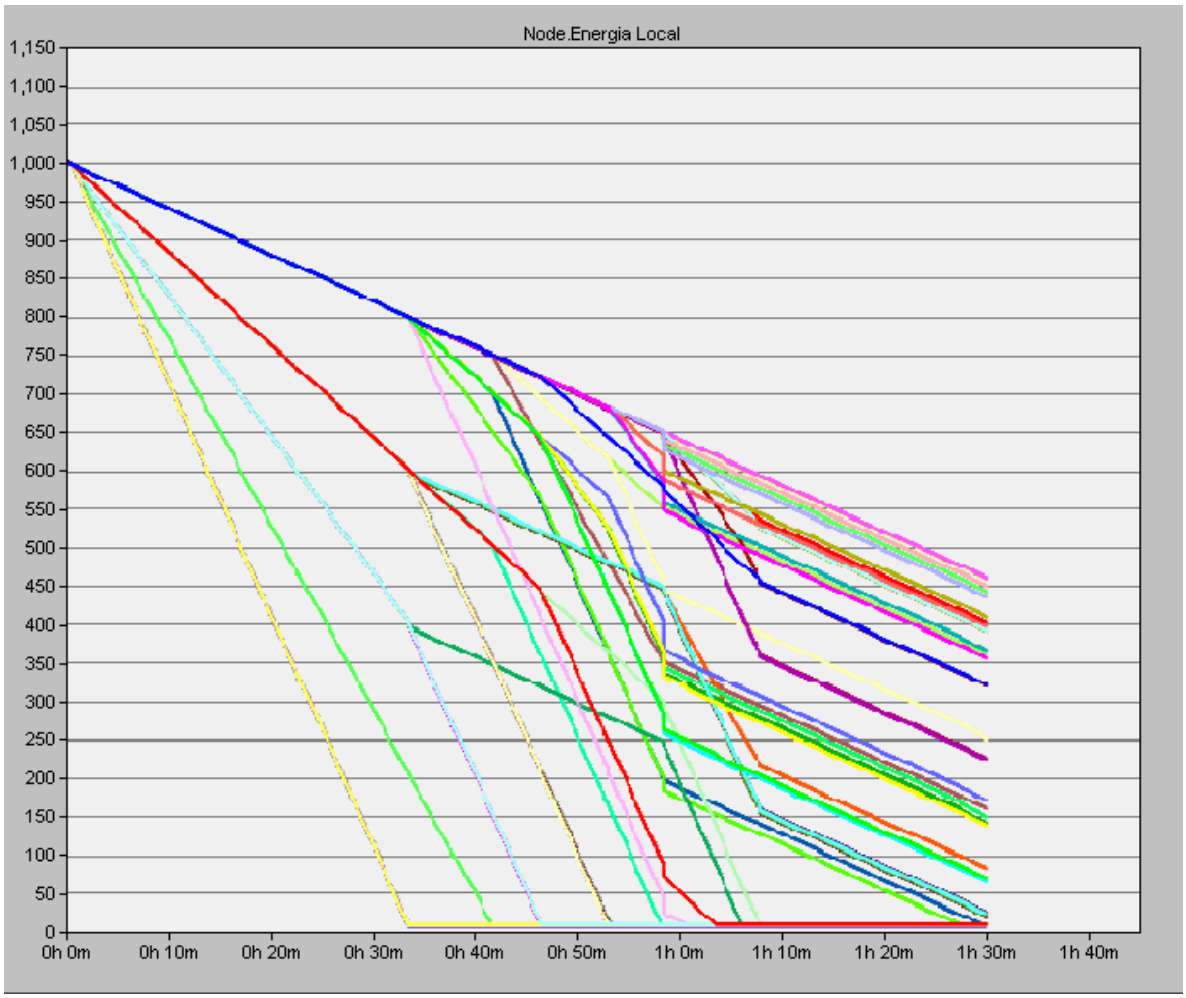
Anexo 40 *End to End Delay* 2-f-MFR



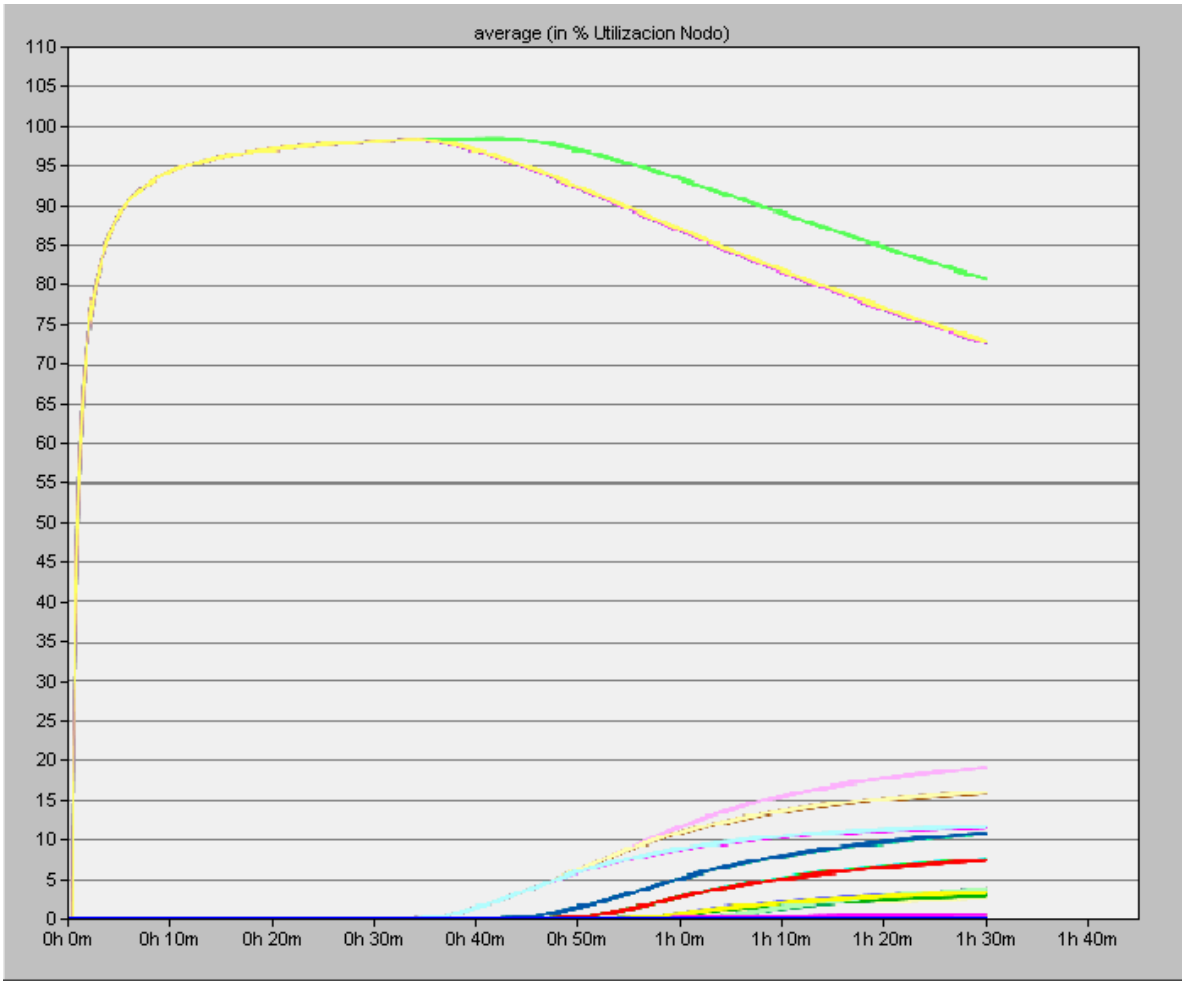
Anexo 41 Paquetes en nodos cóncavos



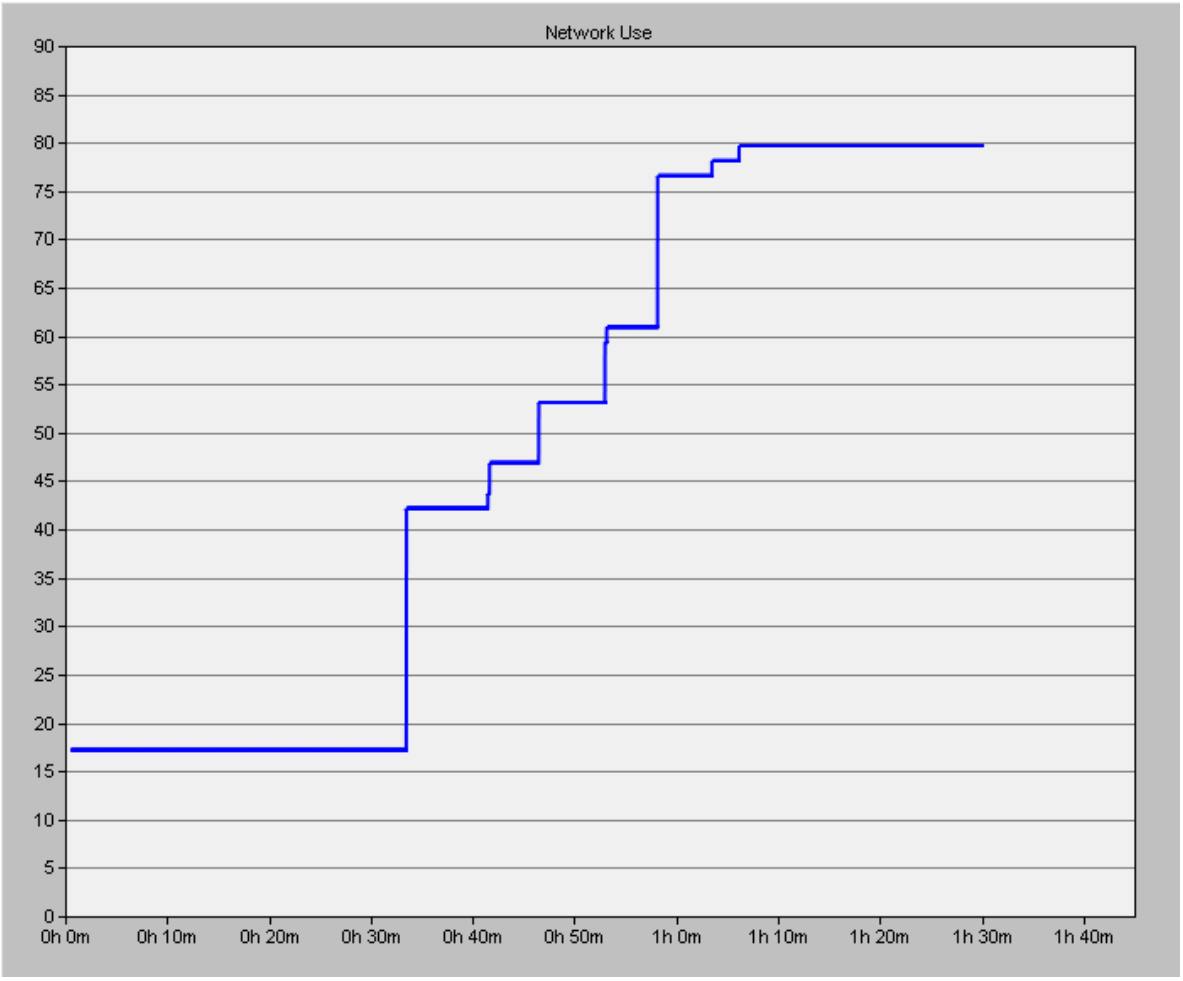
Anexo 42 Energía de los nodos 2-f-MFR



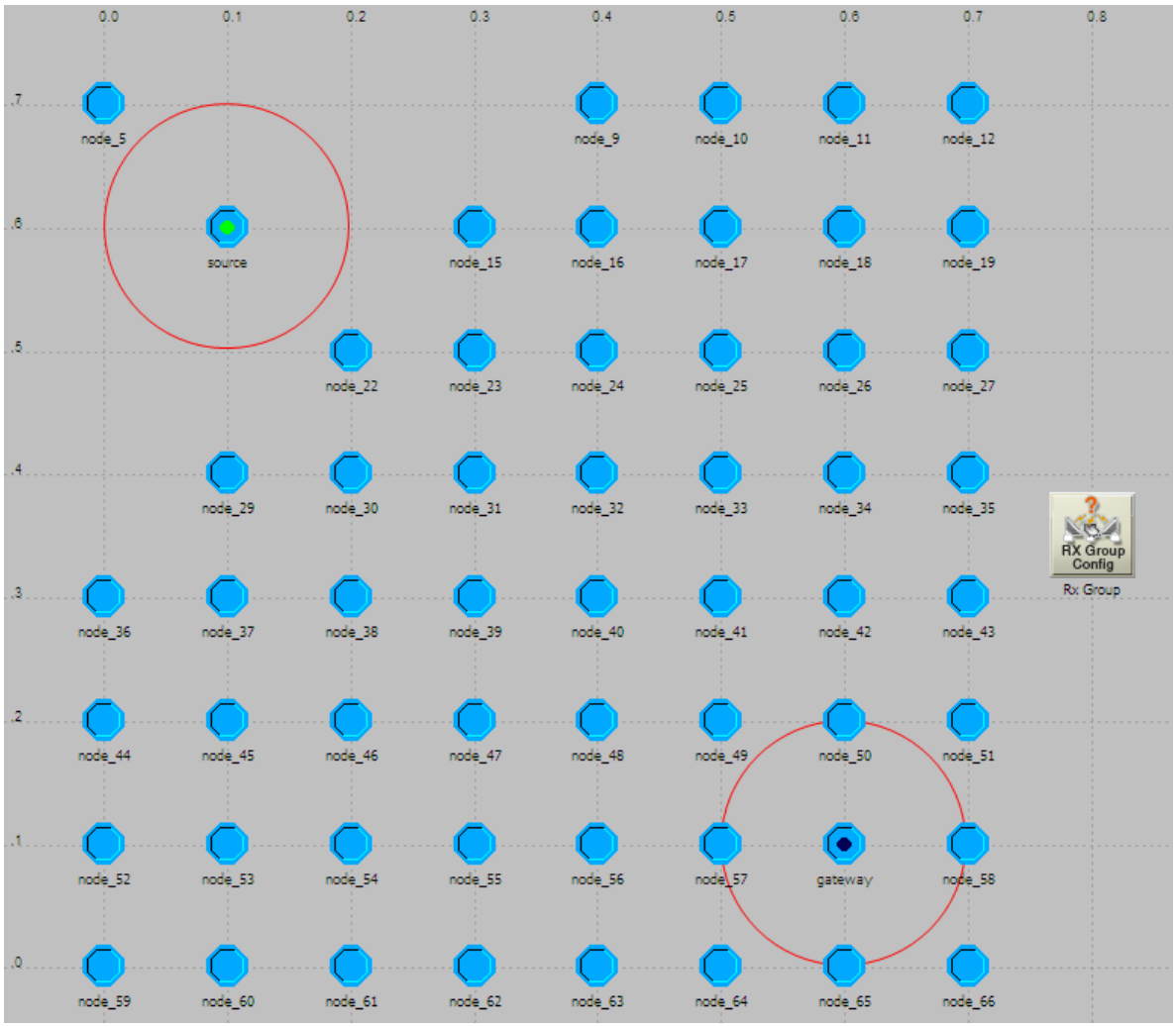
Anexo 43 Porcentaje promedio de utilización de los nodos 2-f-MFR



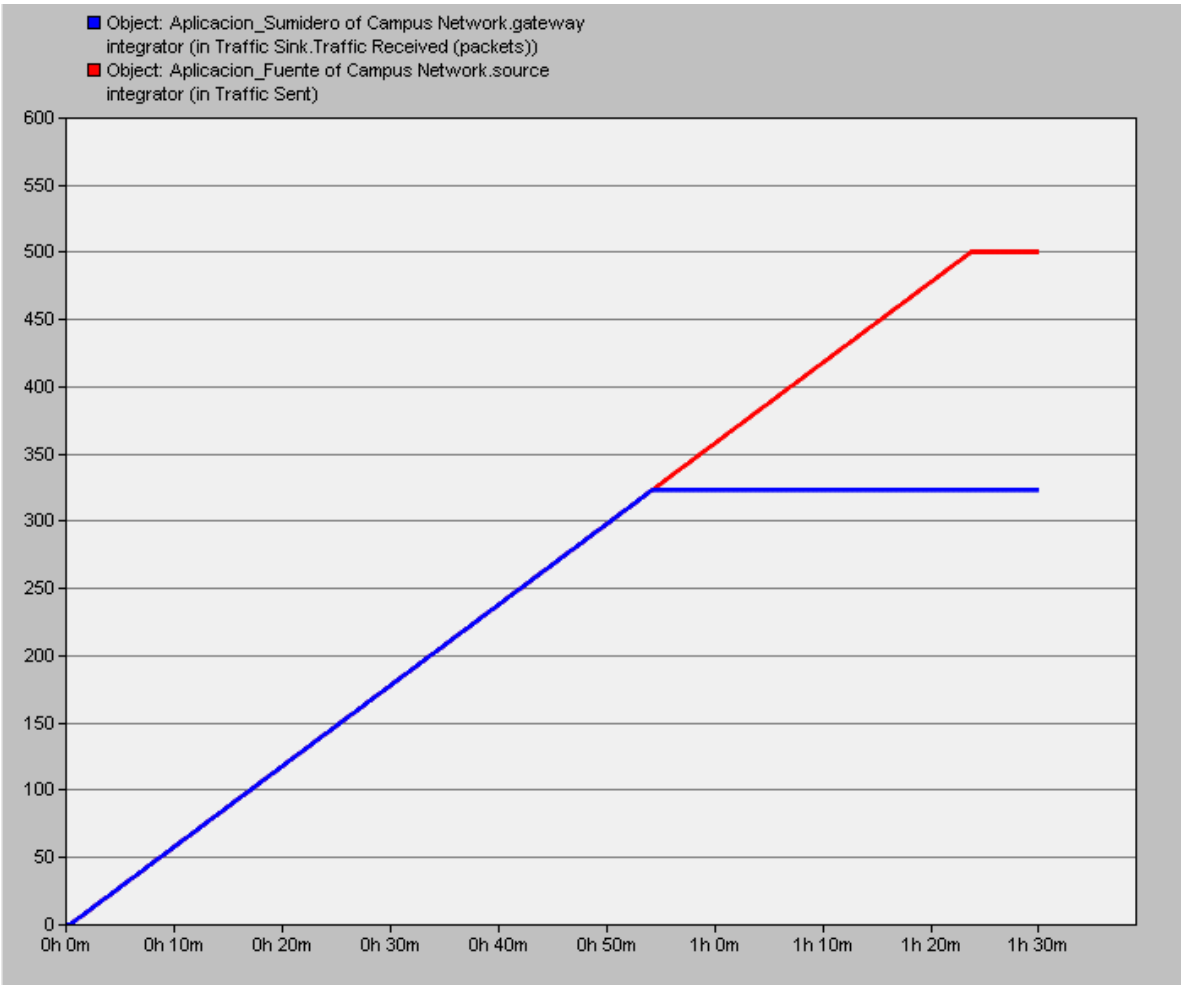
Anexo 44 Porcentaje de utilización de la red 2-f-MFR



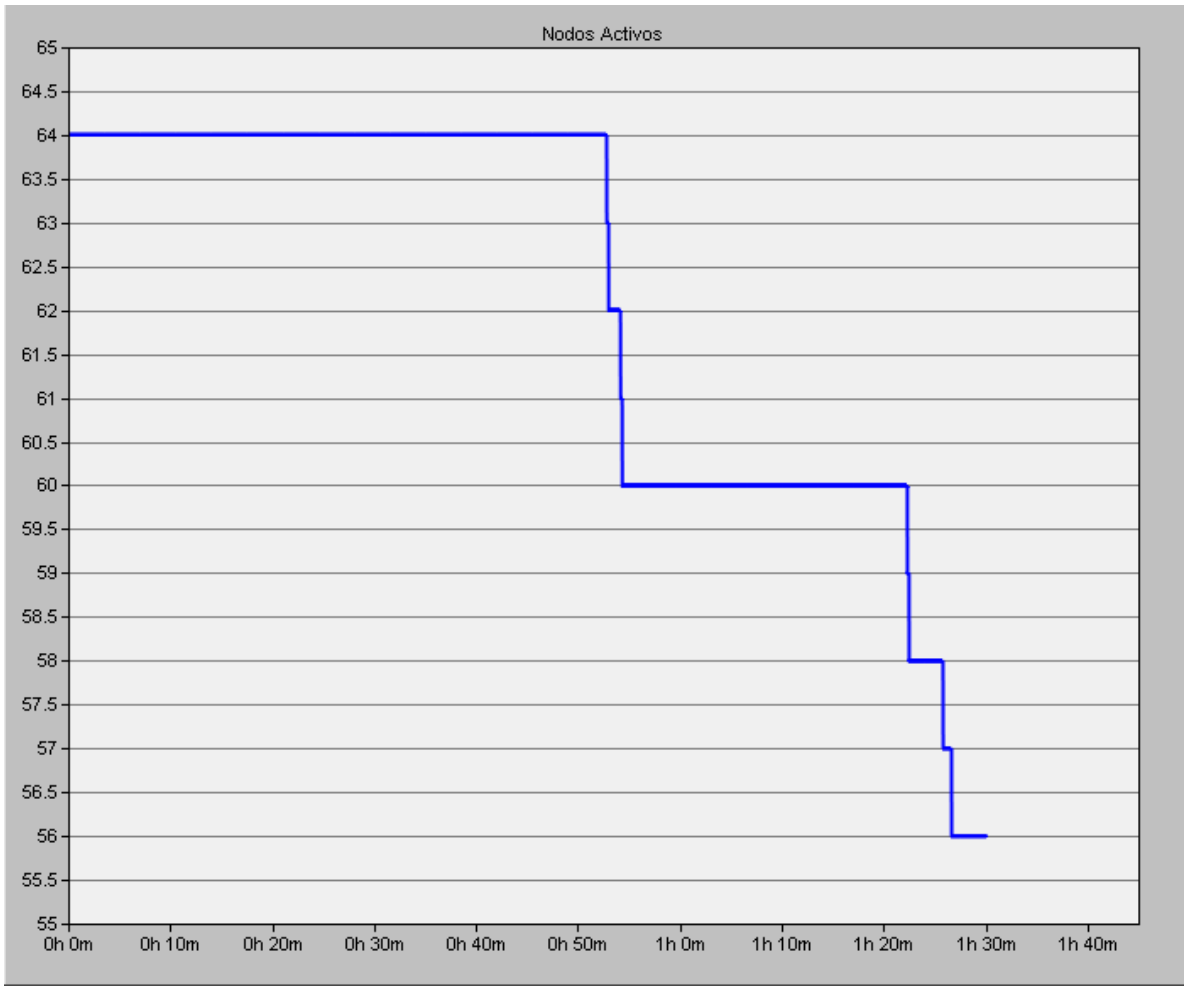
Anexo 45 Escenario GEAR nodos encendidos



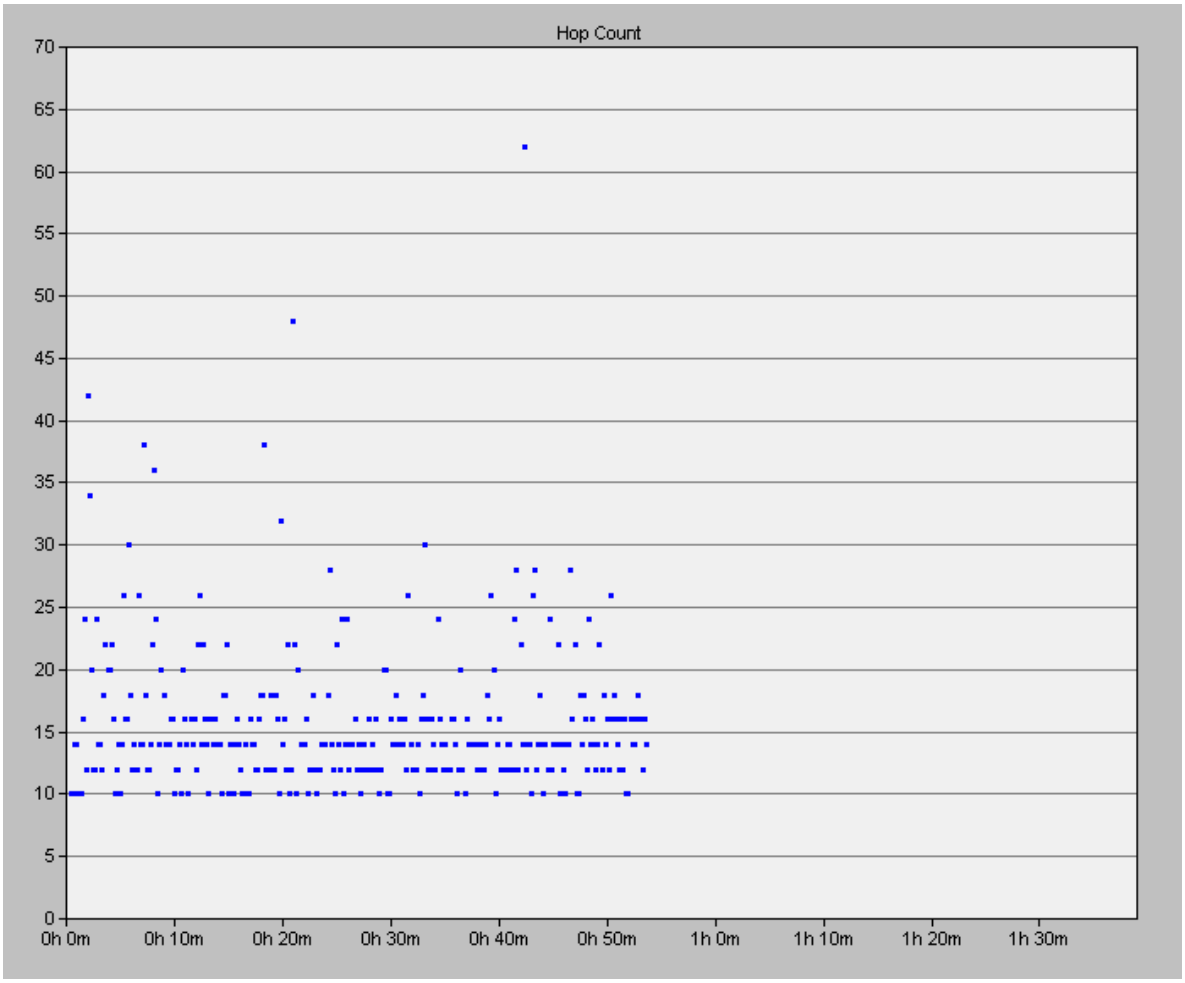
Anexo 46 Paquetes recibidos y paquetes generados GEAR



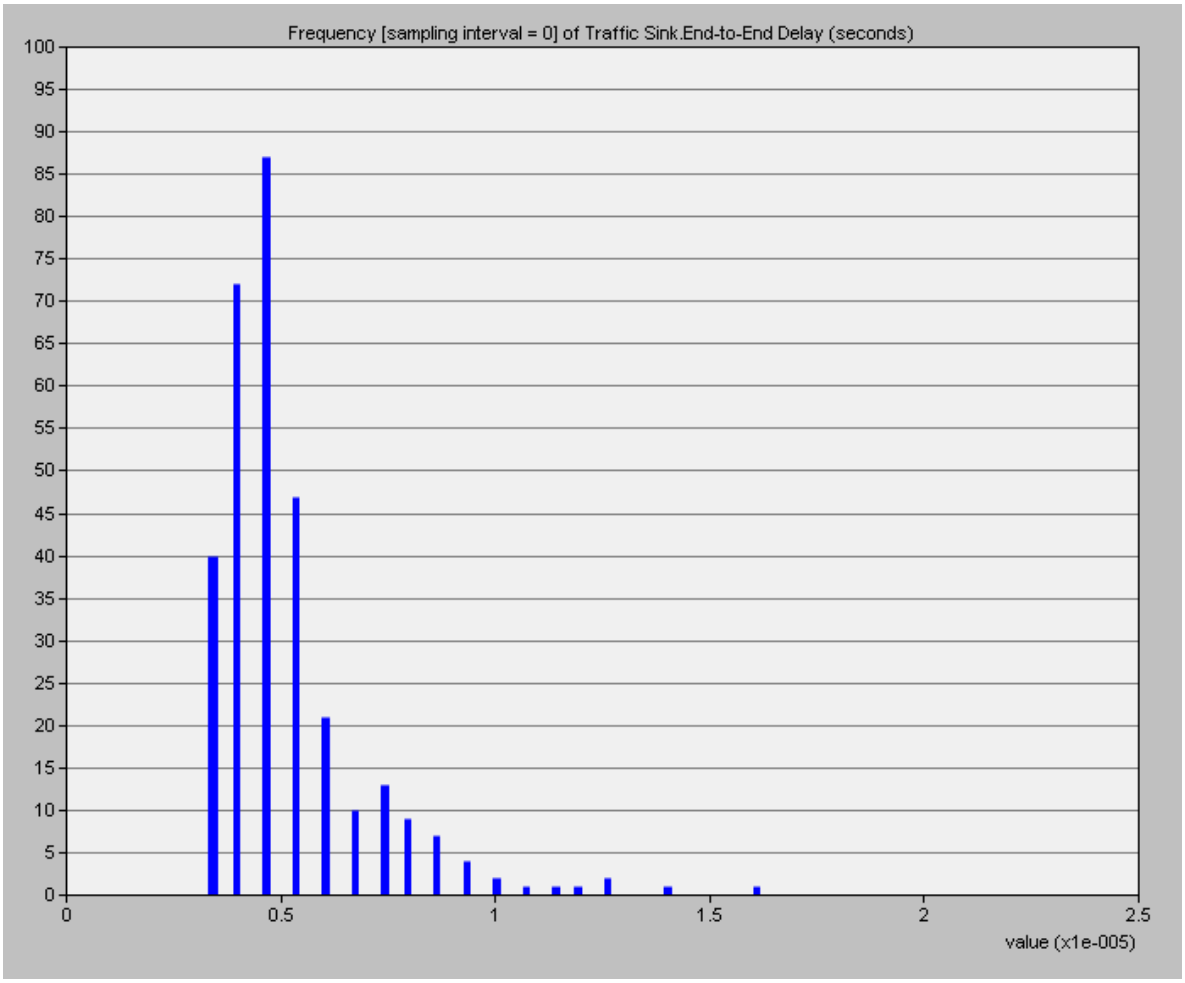
Anexo 47 Nodos activos GEAR



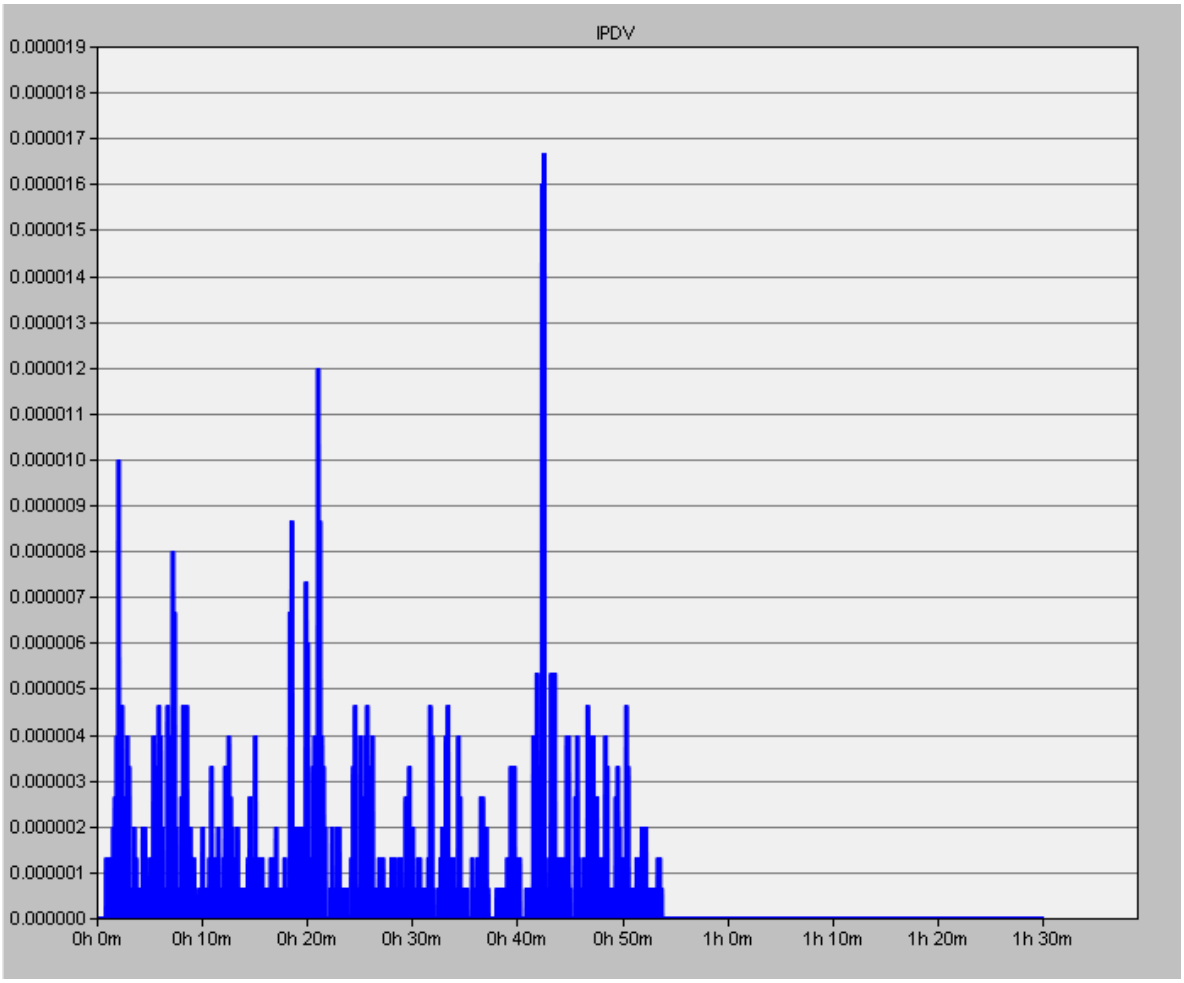
Anexo 48 Número de saltos GEAR



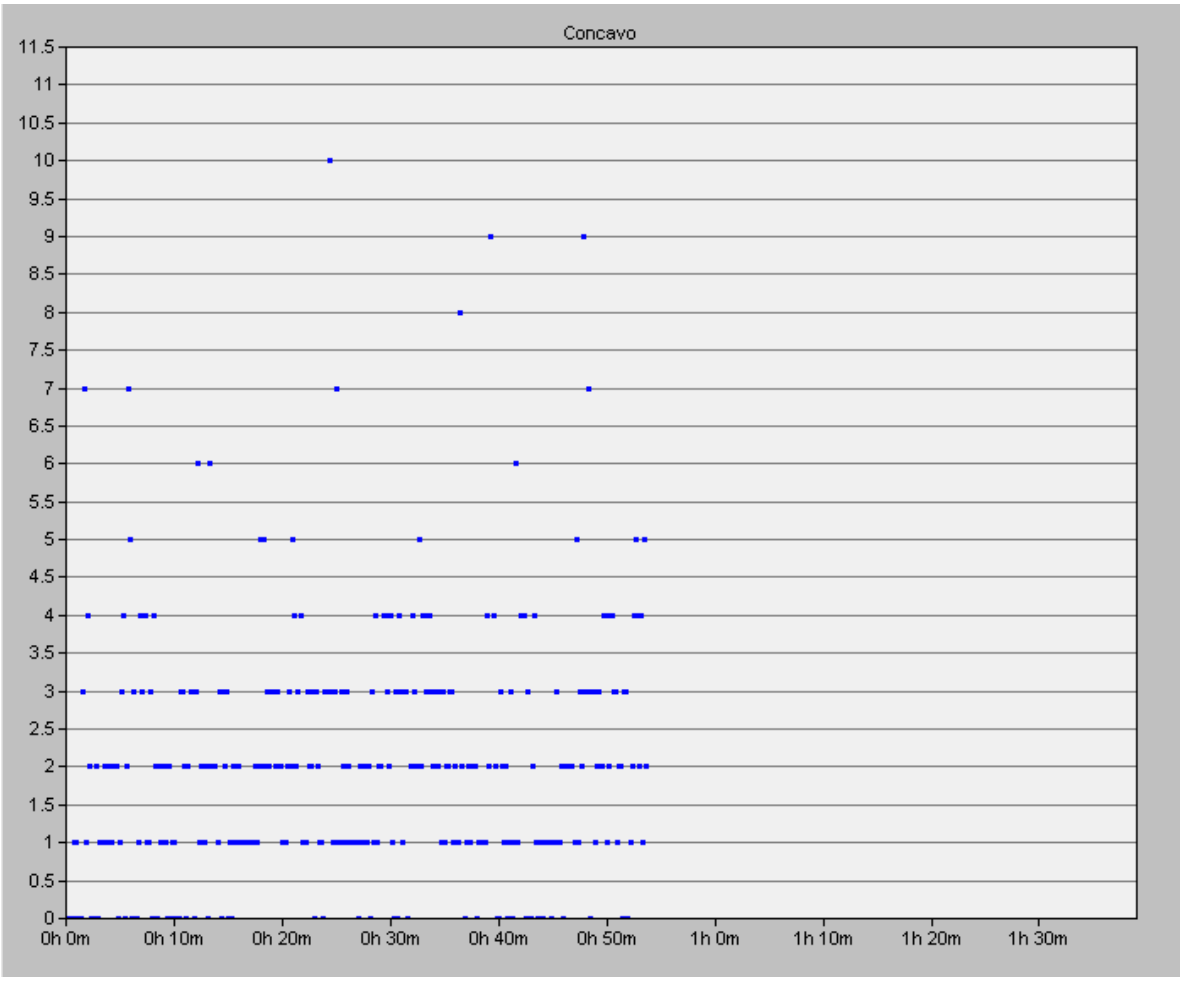
Anexo 49 *End to end delay* GEAR



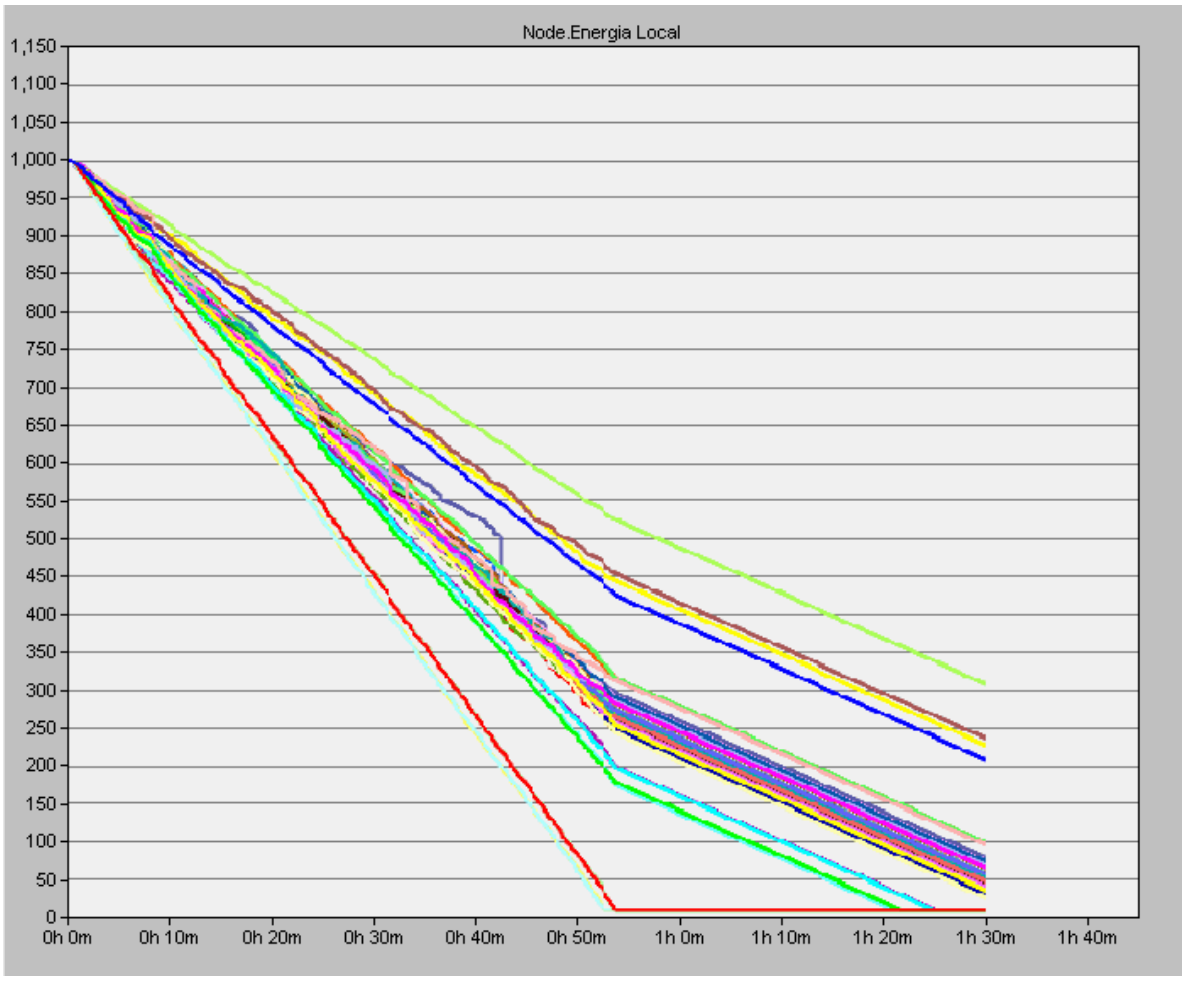
Anexo 50 IPDV GEAR



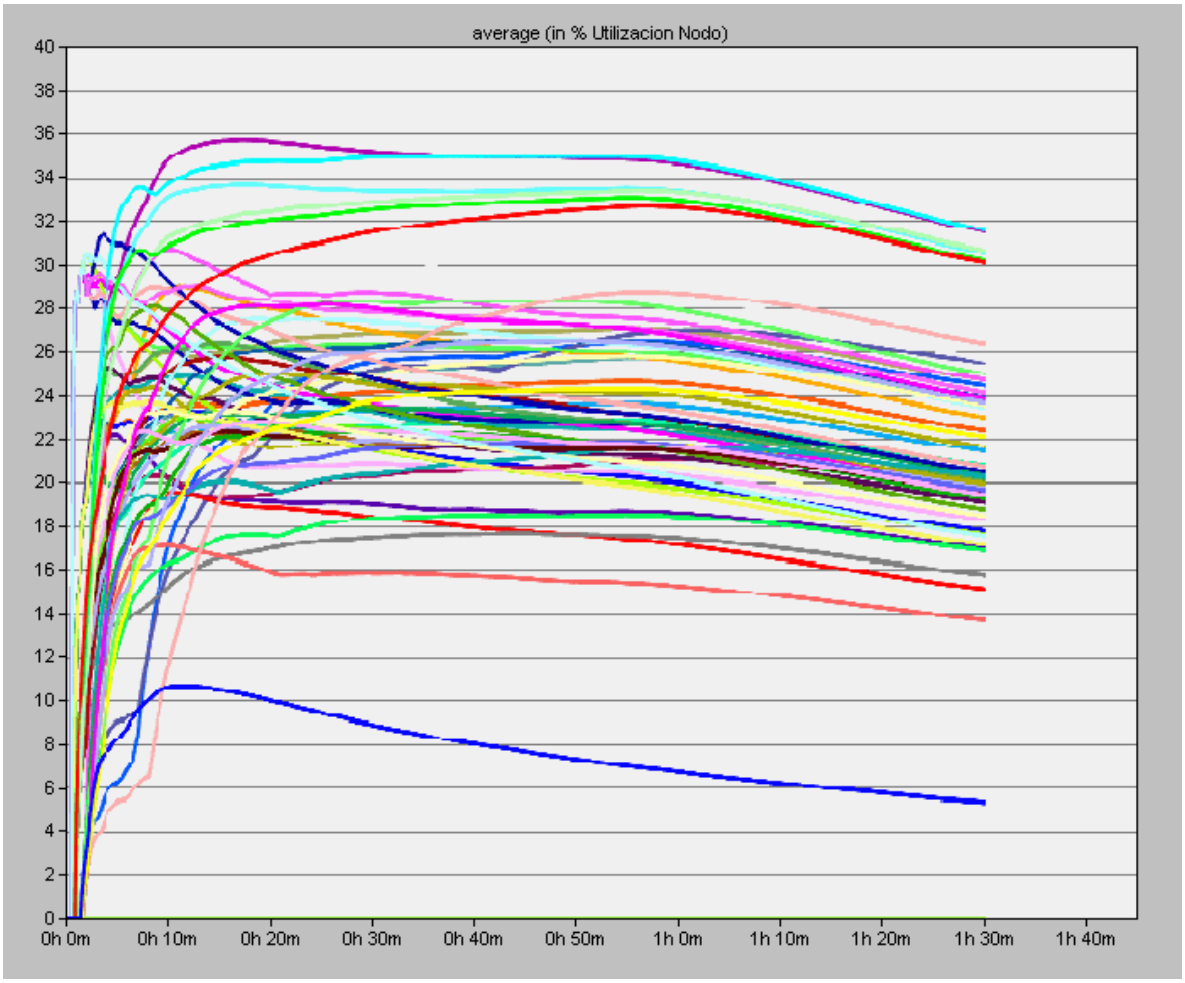
Anexo 51 Paquetes en nodos cóncavos GEAR



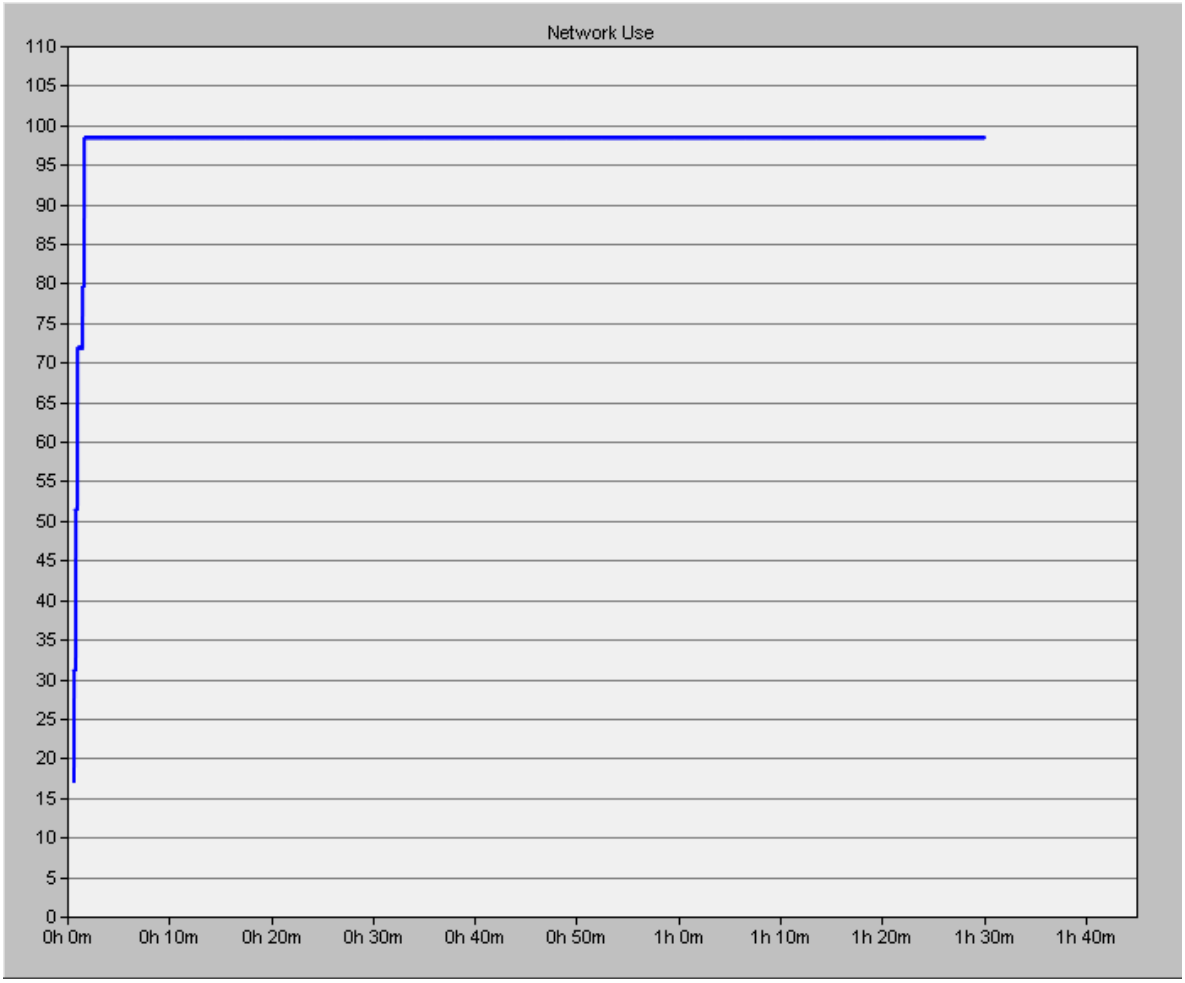
Anexo 52 Energía de los nodos en GEAR



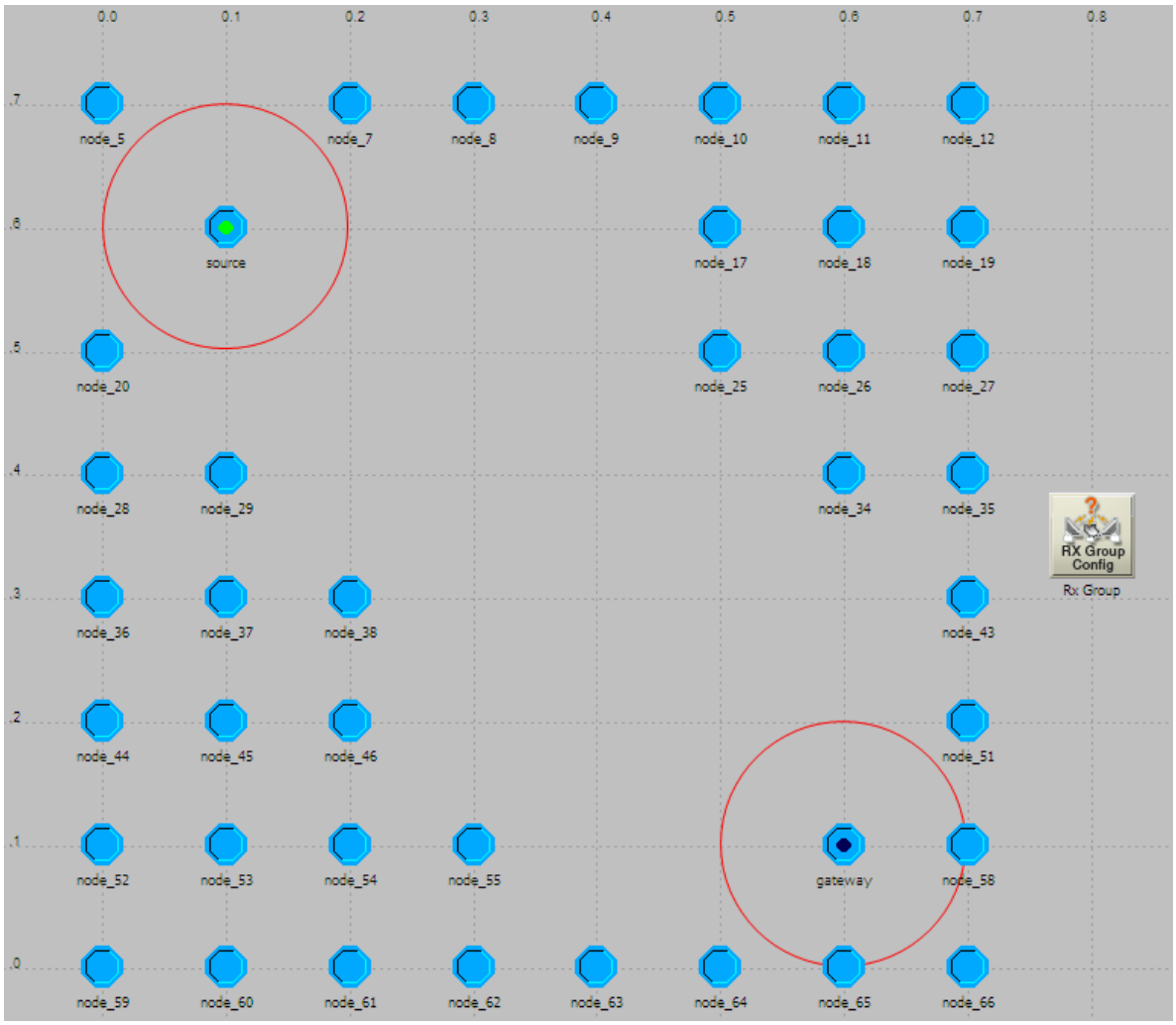
Anexo 53 Porcentaje promedio de la utilización de los nodos GEAR



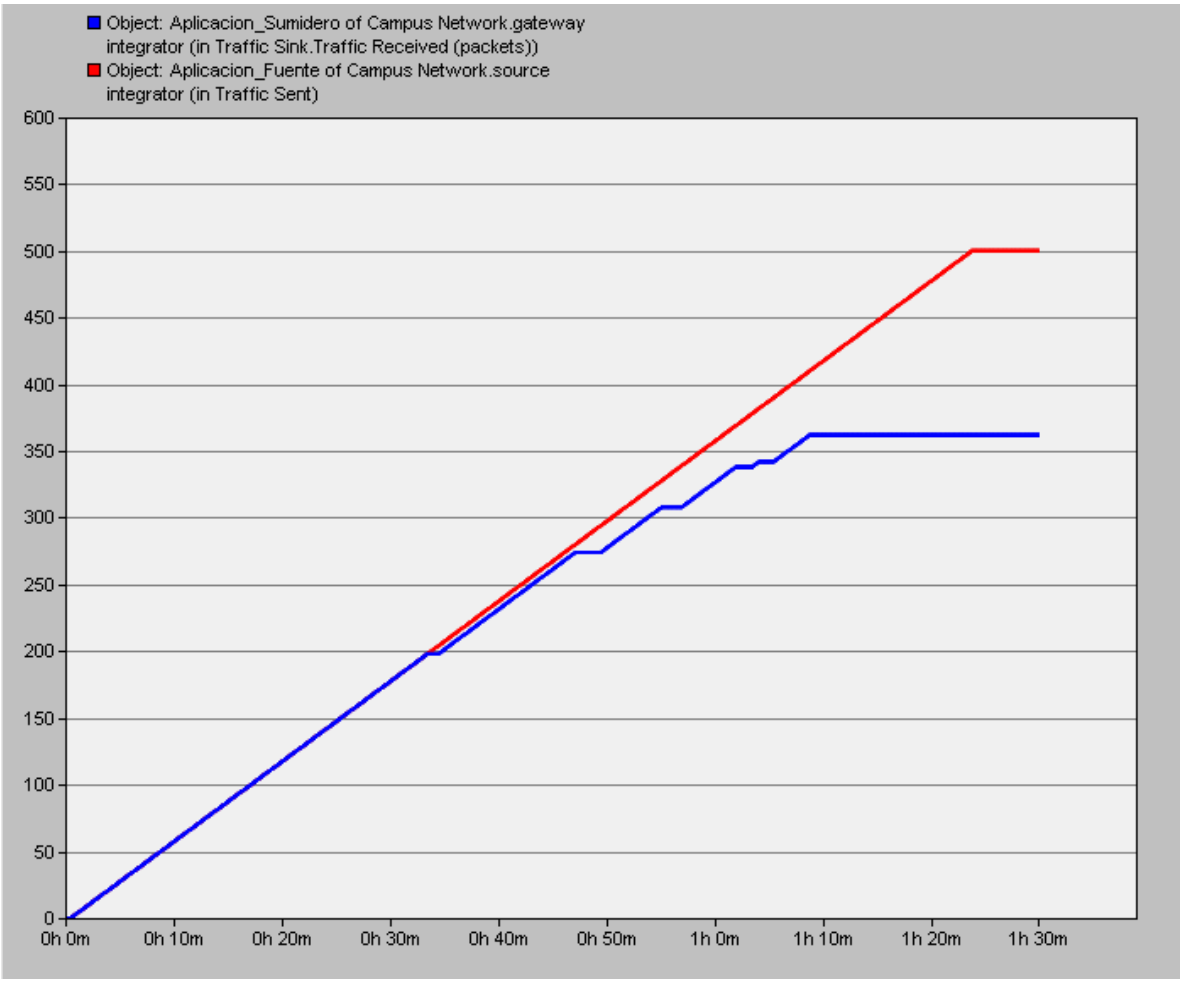
Anexo 54 Porcentaje de de utilización de la red GEAR



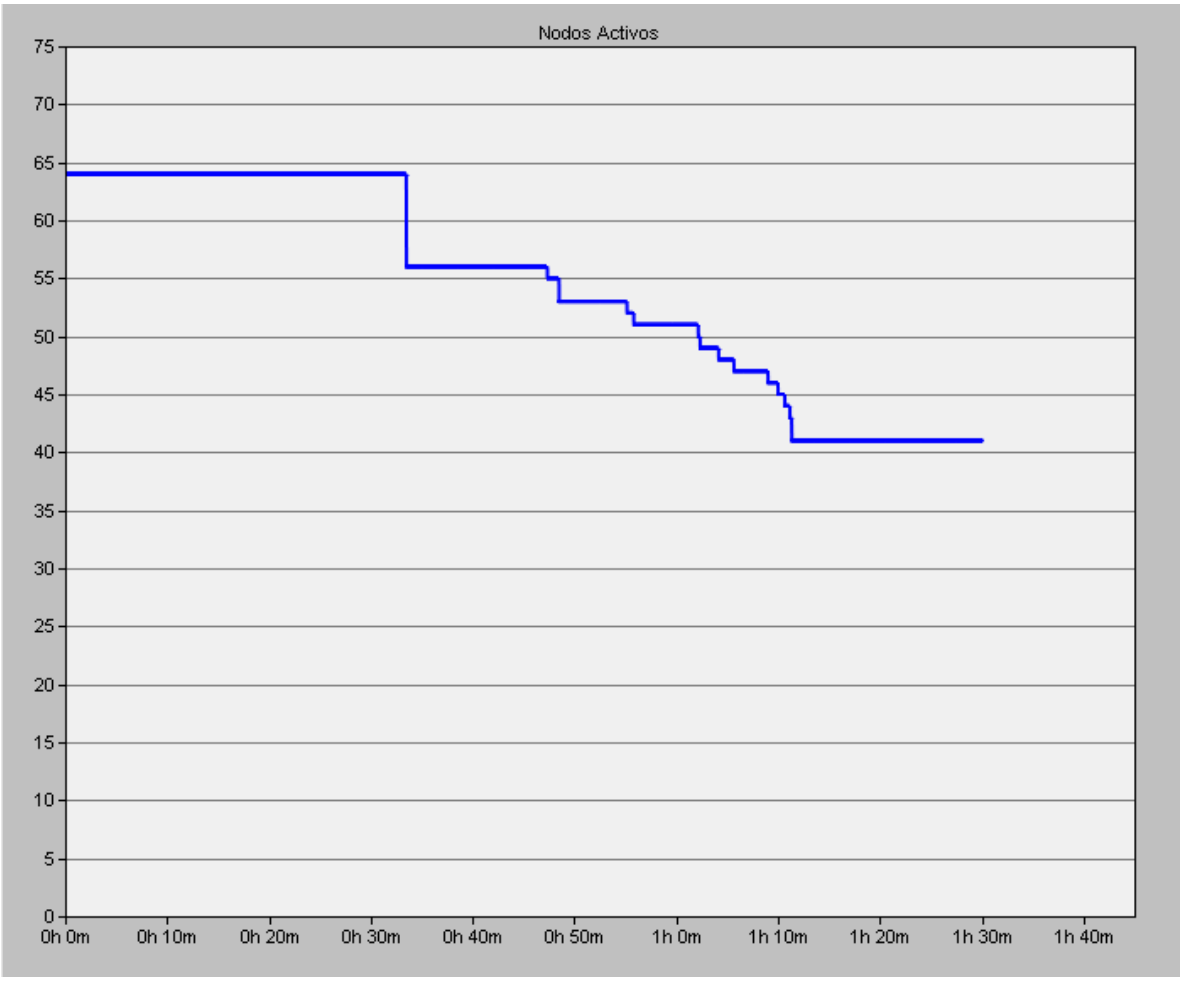
Anexo 55 Escenario GDSTR nodos activos



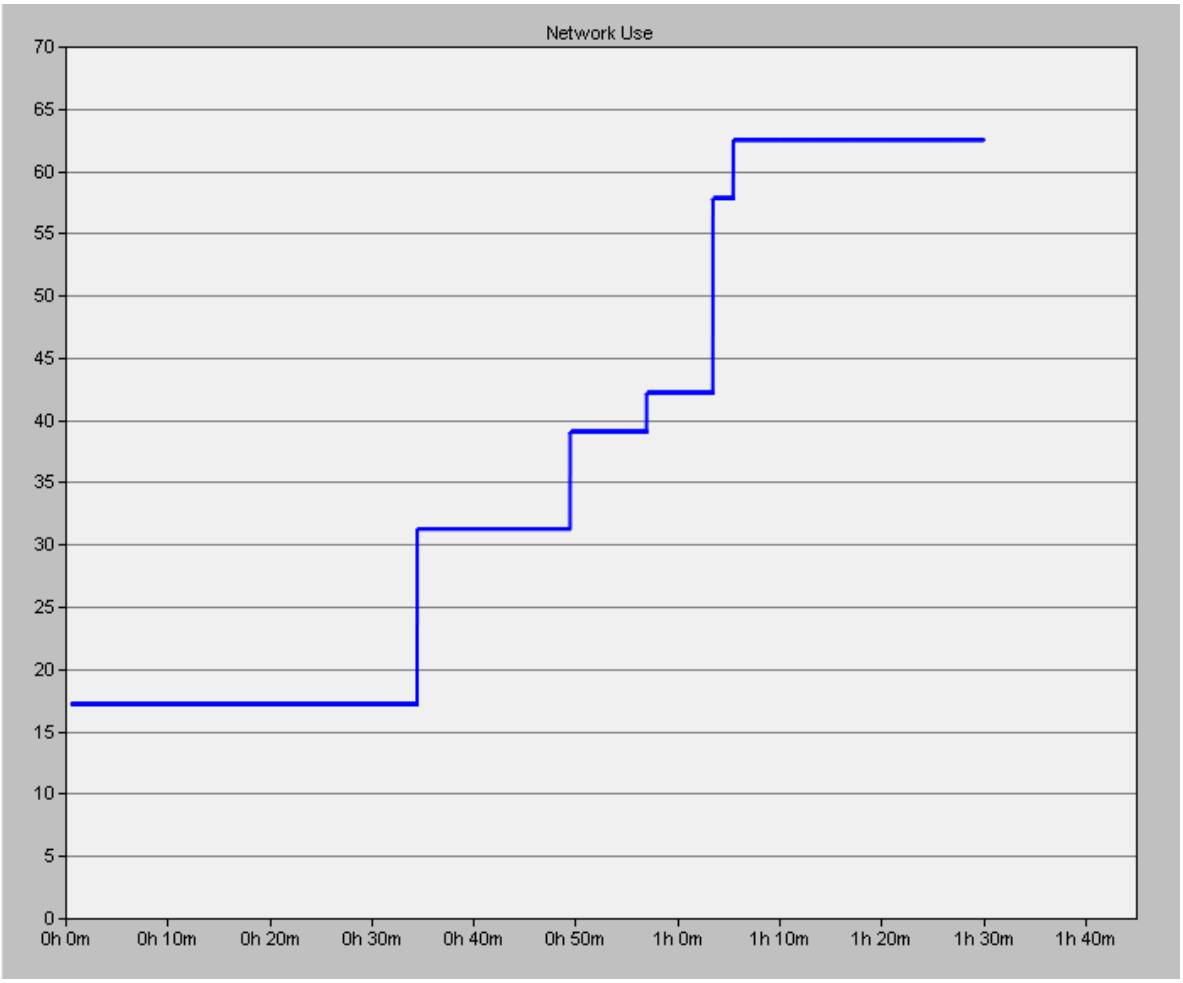
Anexo 56 Paquetes generados y paquetes recibidos GDSTR



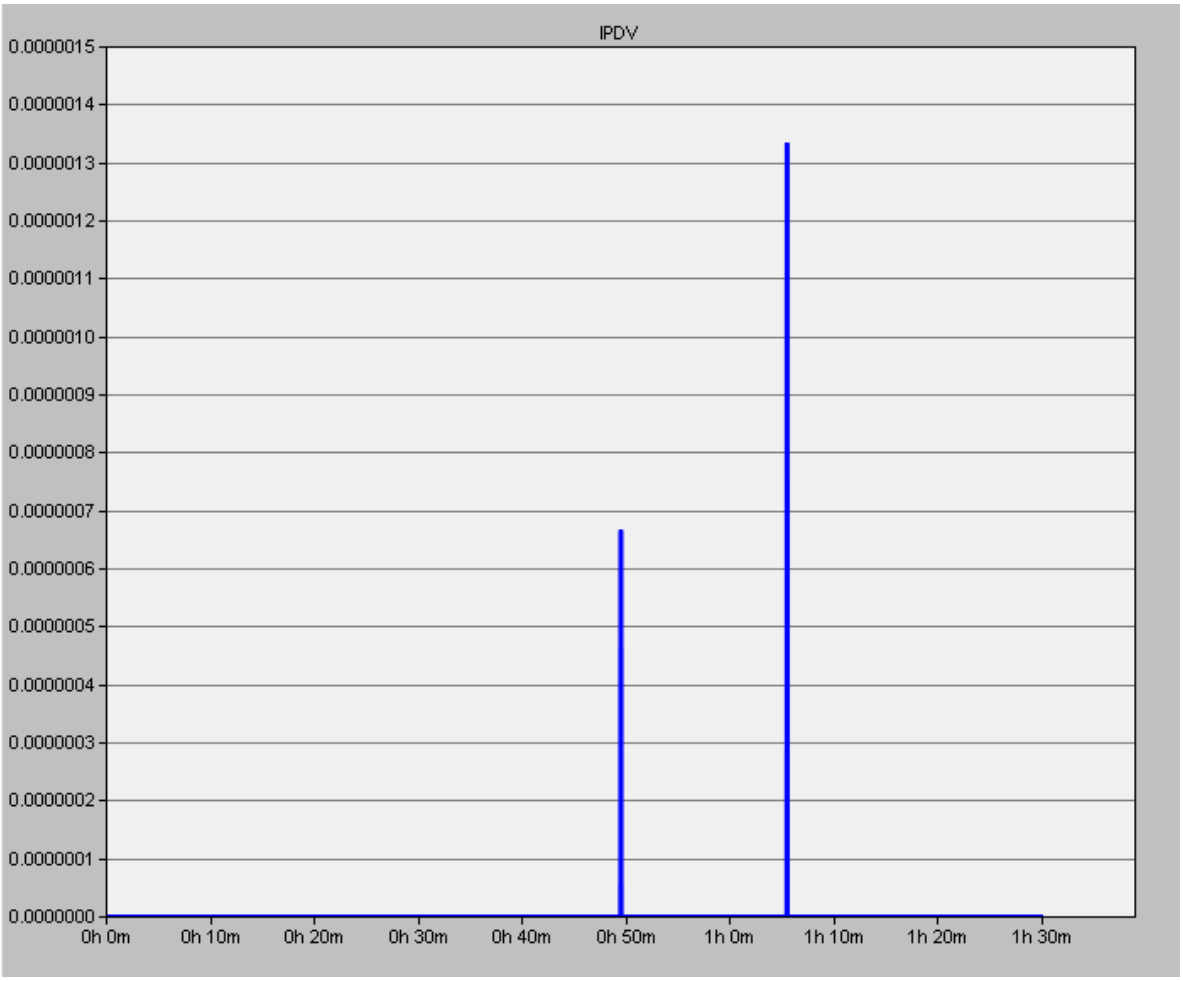
Anexo 57 Nodos activos GDSTR



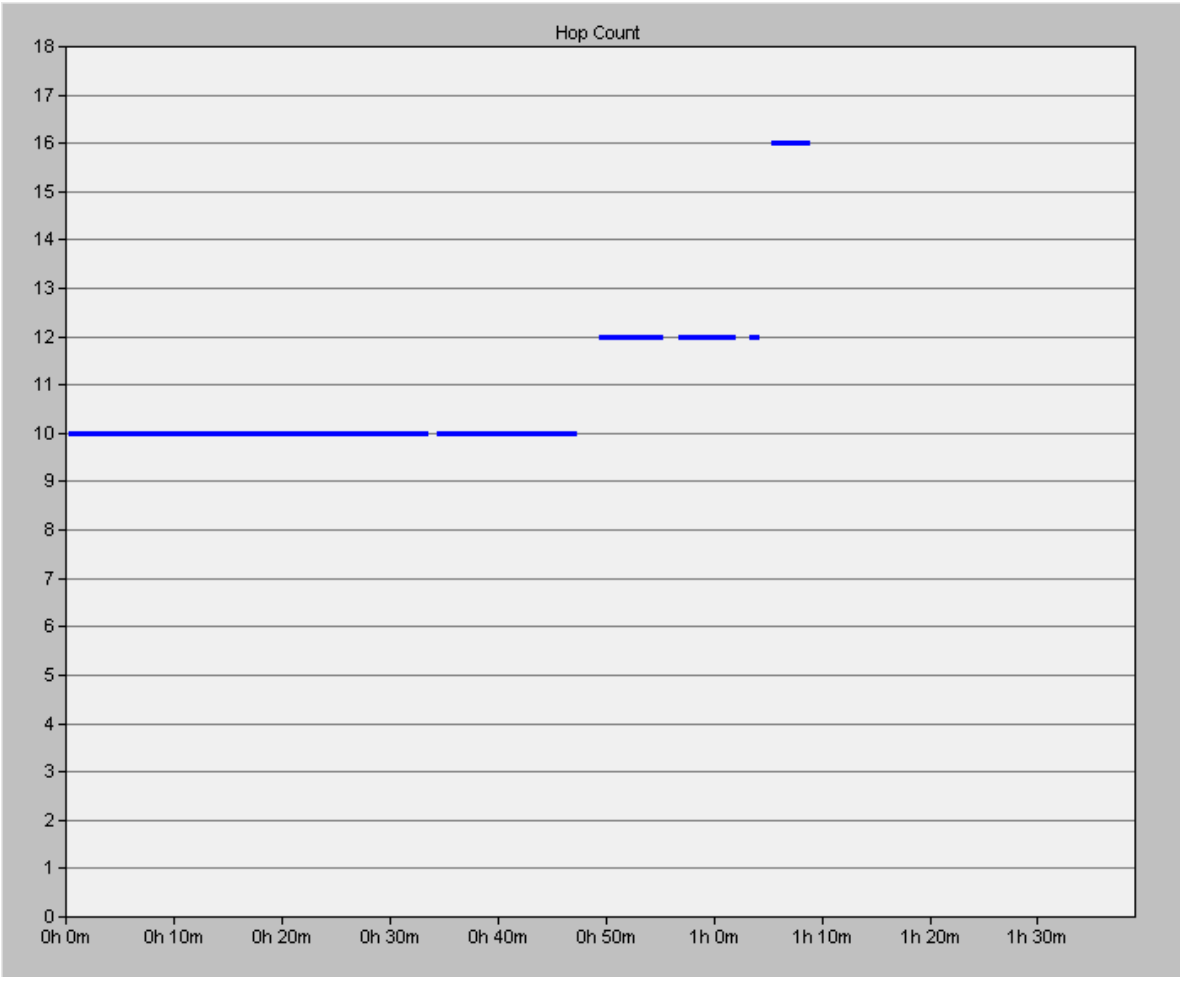
Anexo 58 Porcentaje de utilización de la red GDSTR



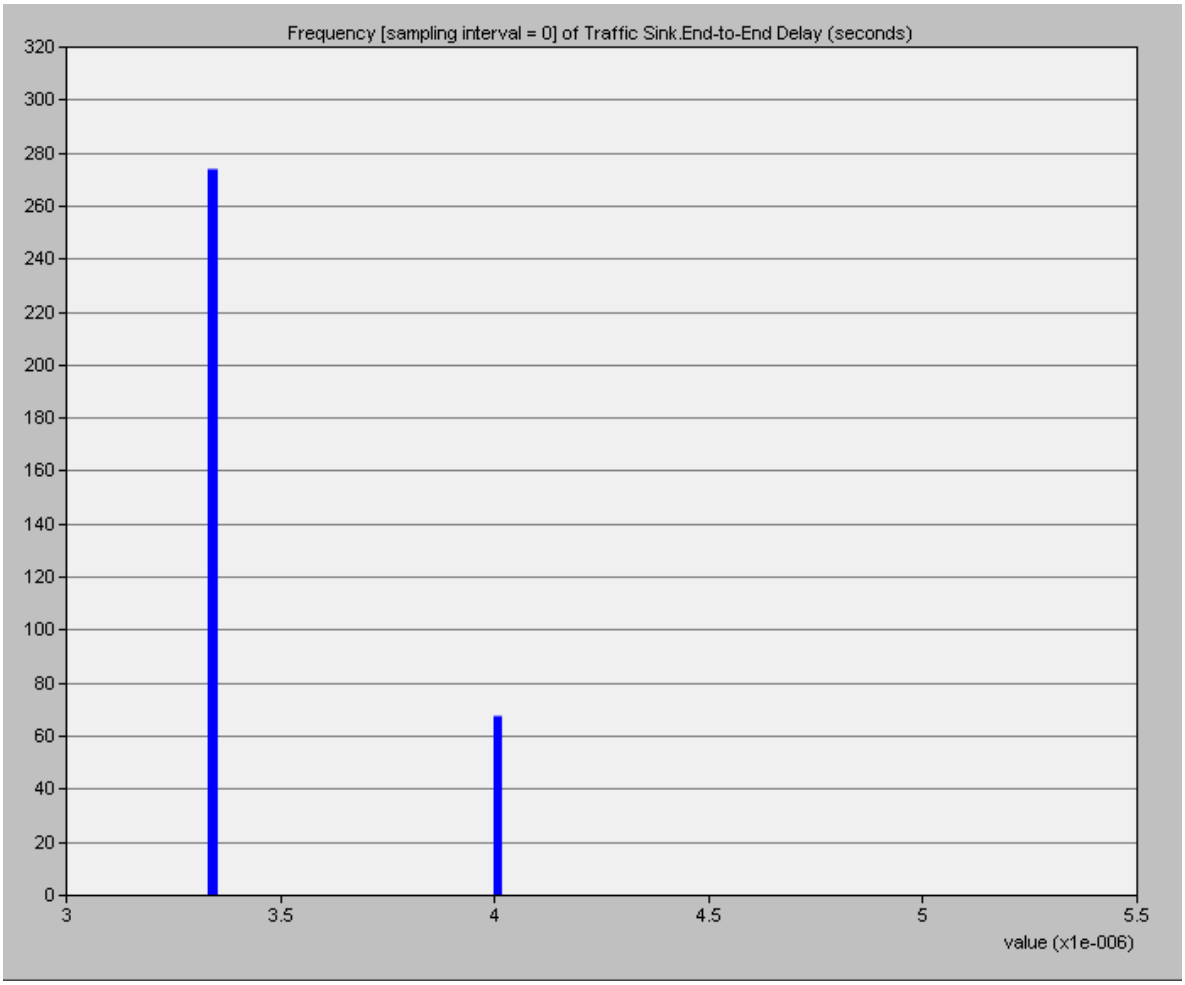
Anexo 59 IPDV GDSTR



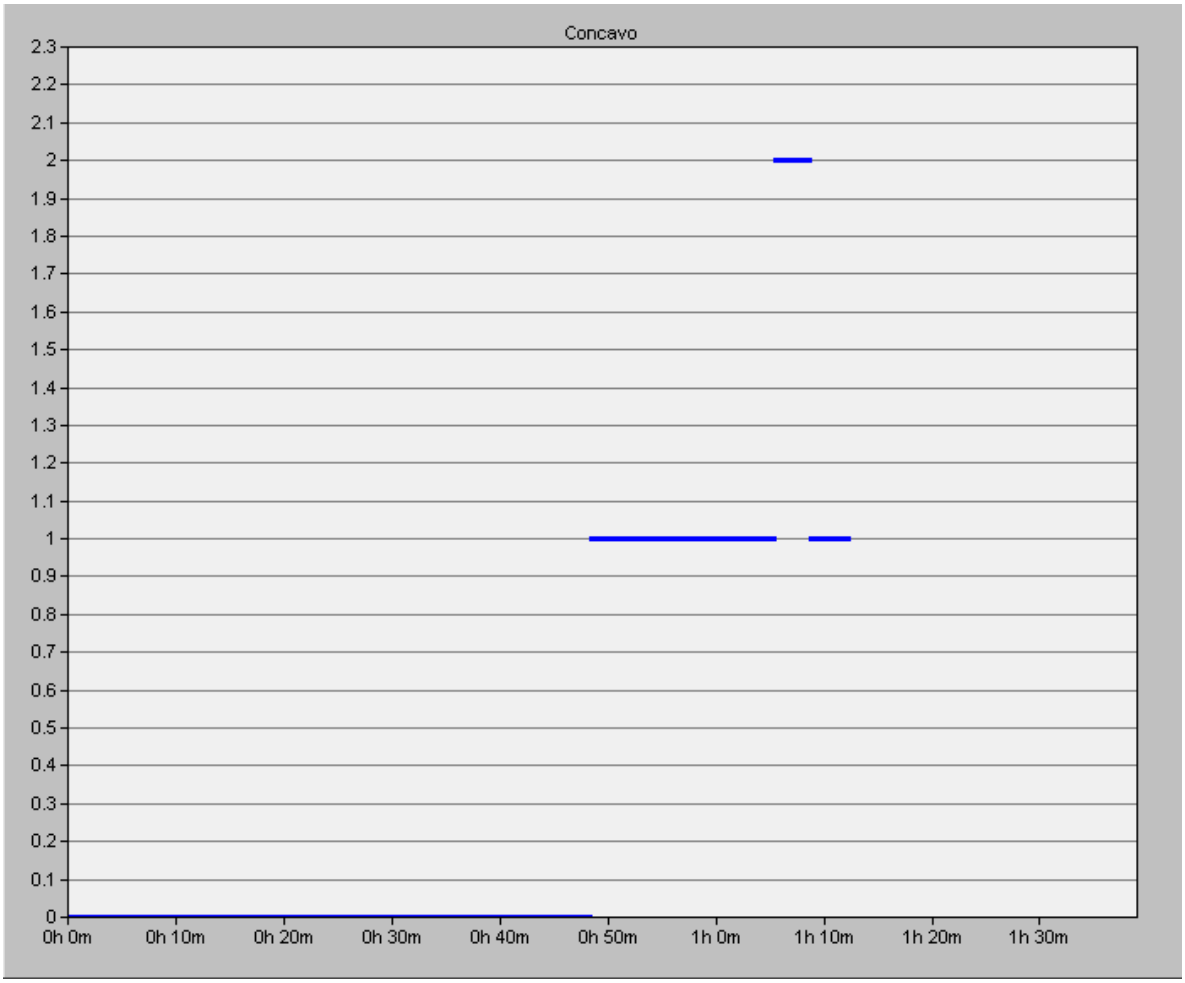
Anexo 60 *Hop count* GDSTR



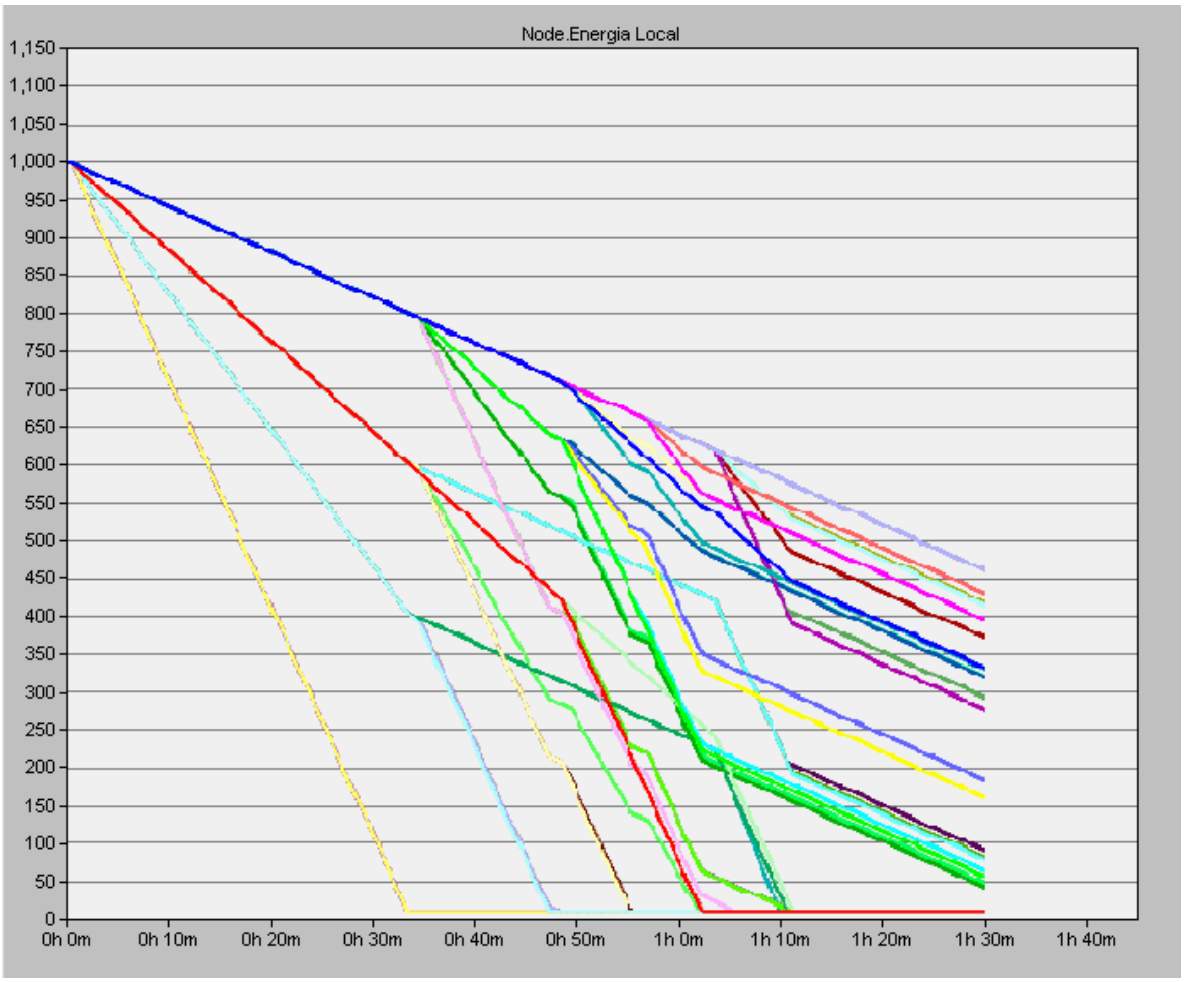
Anexo 61 *End to end delay* GDSTR



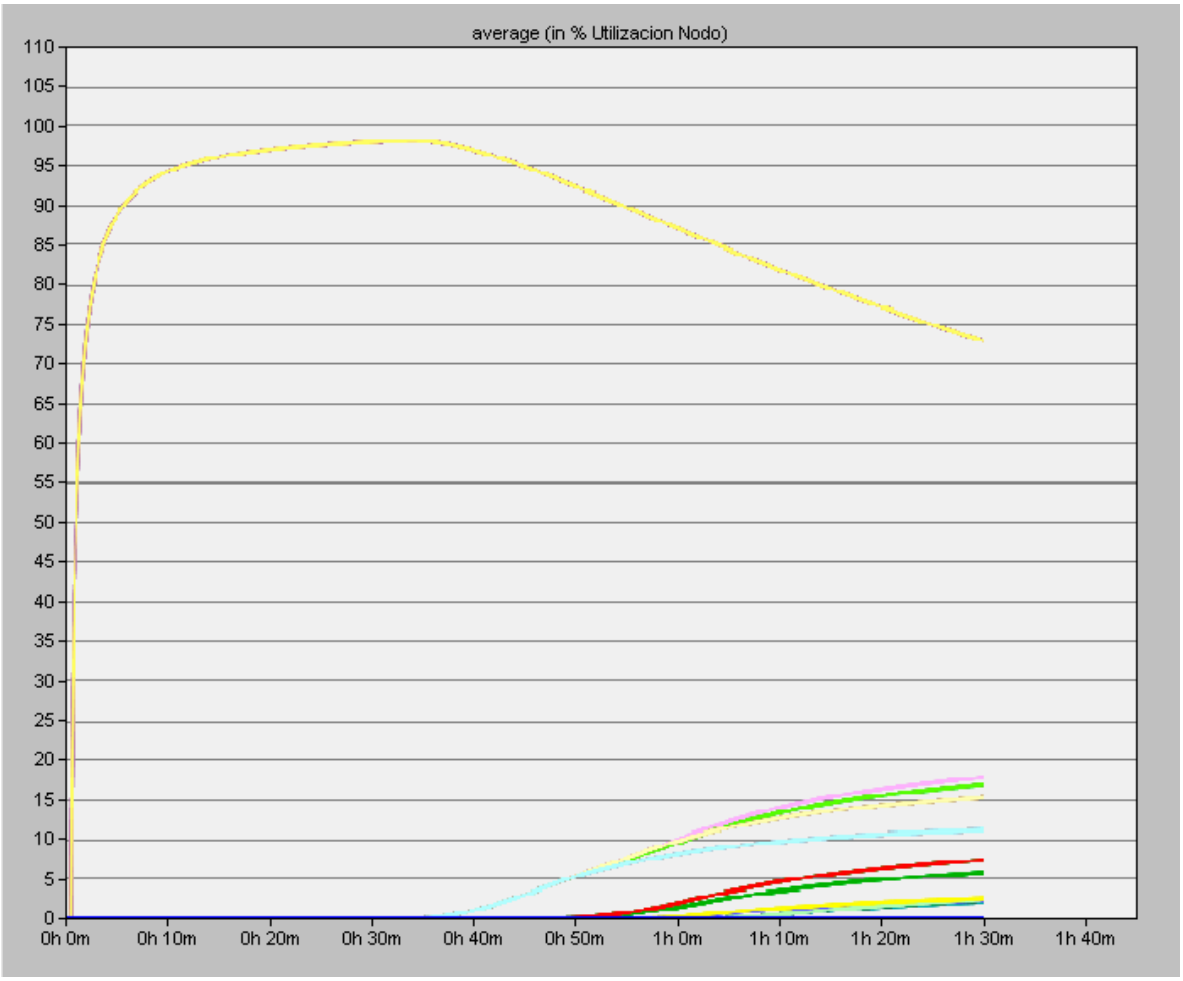
Anexo 62 Paquetes en nodos cóncavos GDSTR



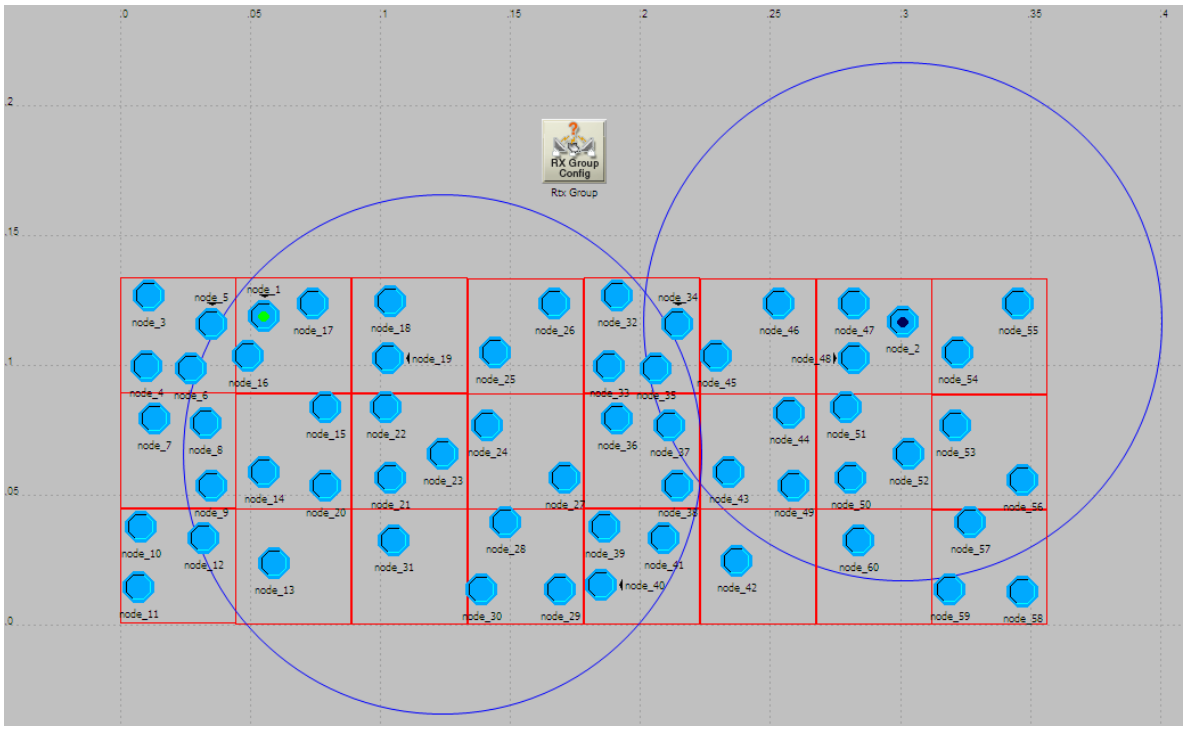
Anexo 63 Energía de los nodos en GDSTR



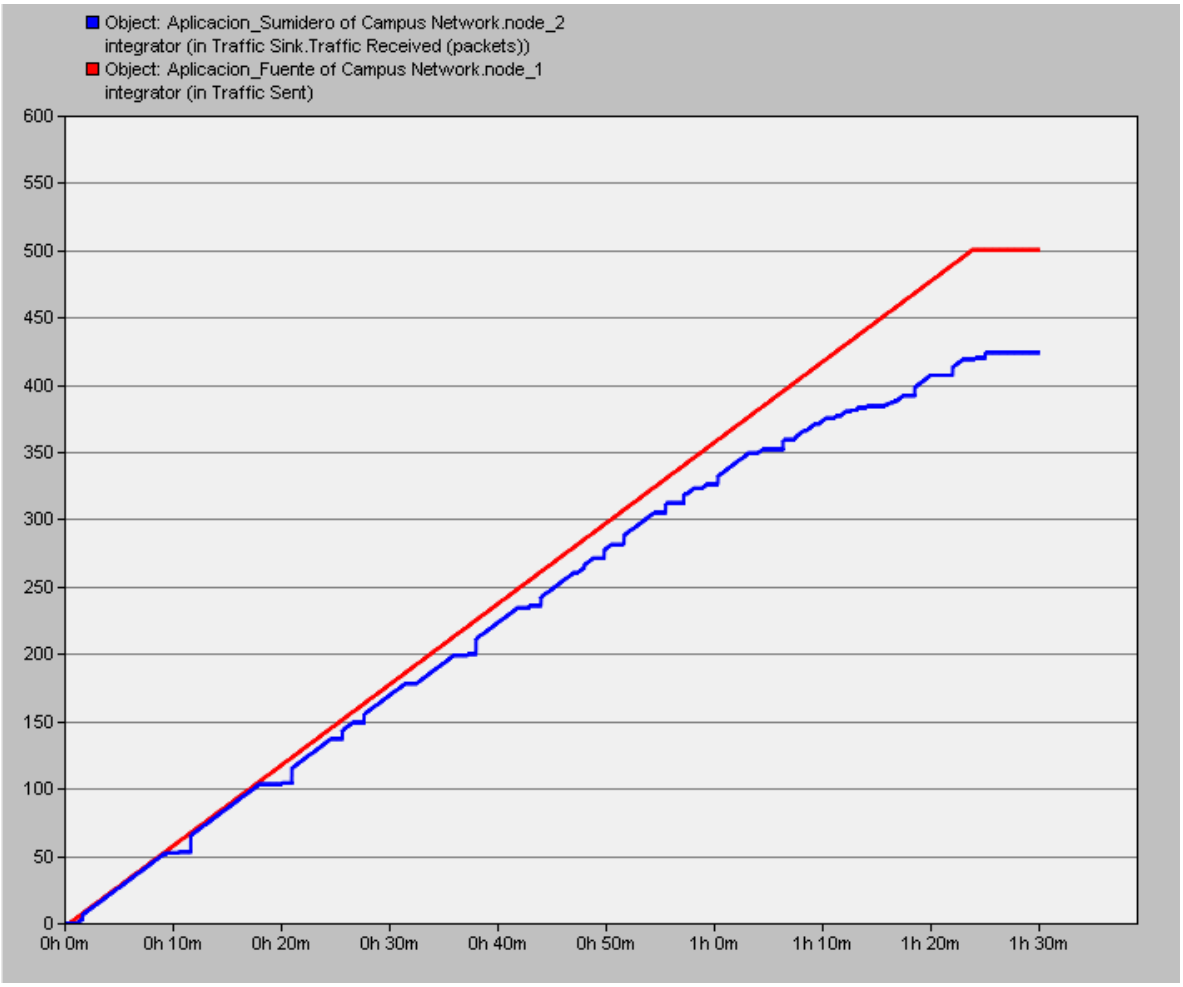
Anexo 64 Porcentaje promedio de utilización de los nodos GDSTR



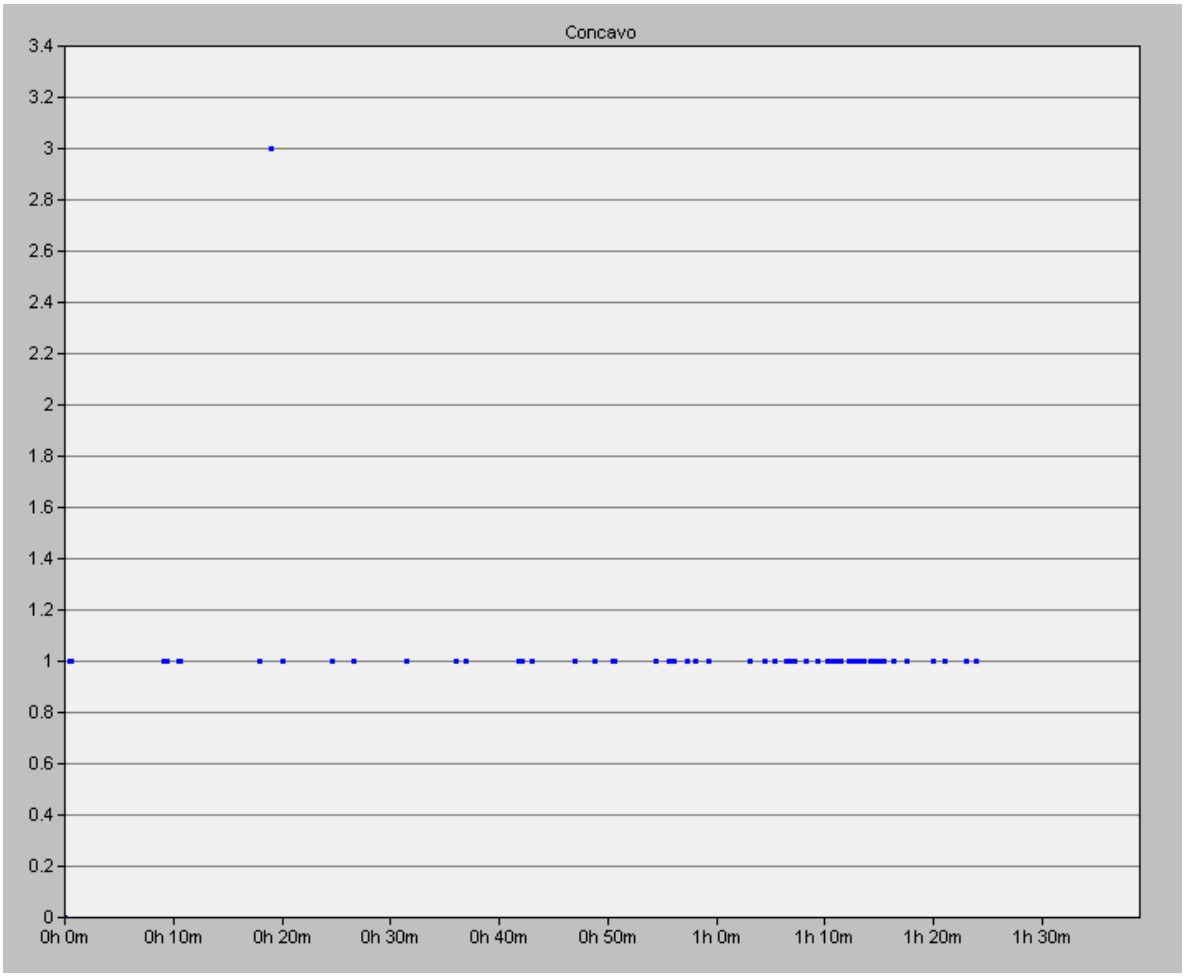
Anexo 65 Escenario de GAF



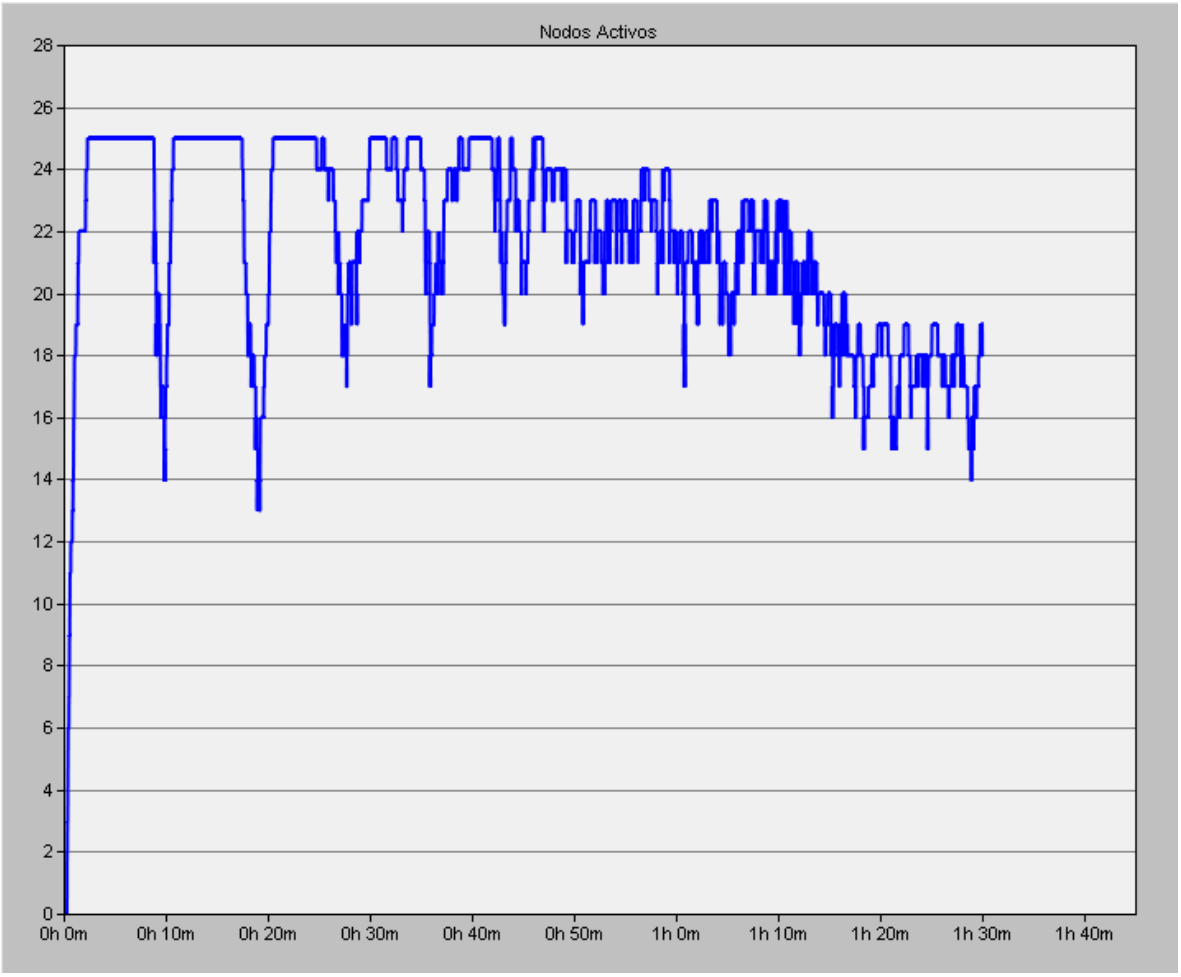
Anexo 66 Paquetes recibidos y paquetes generados GAF



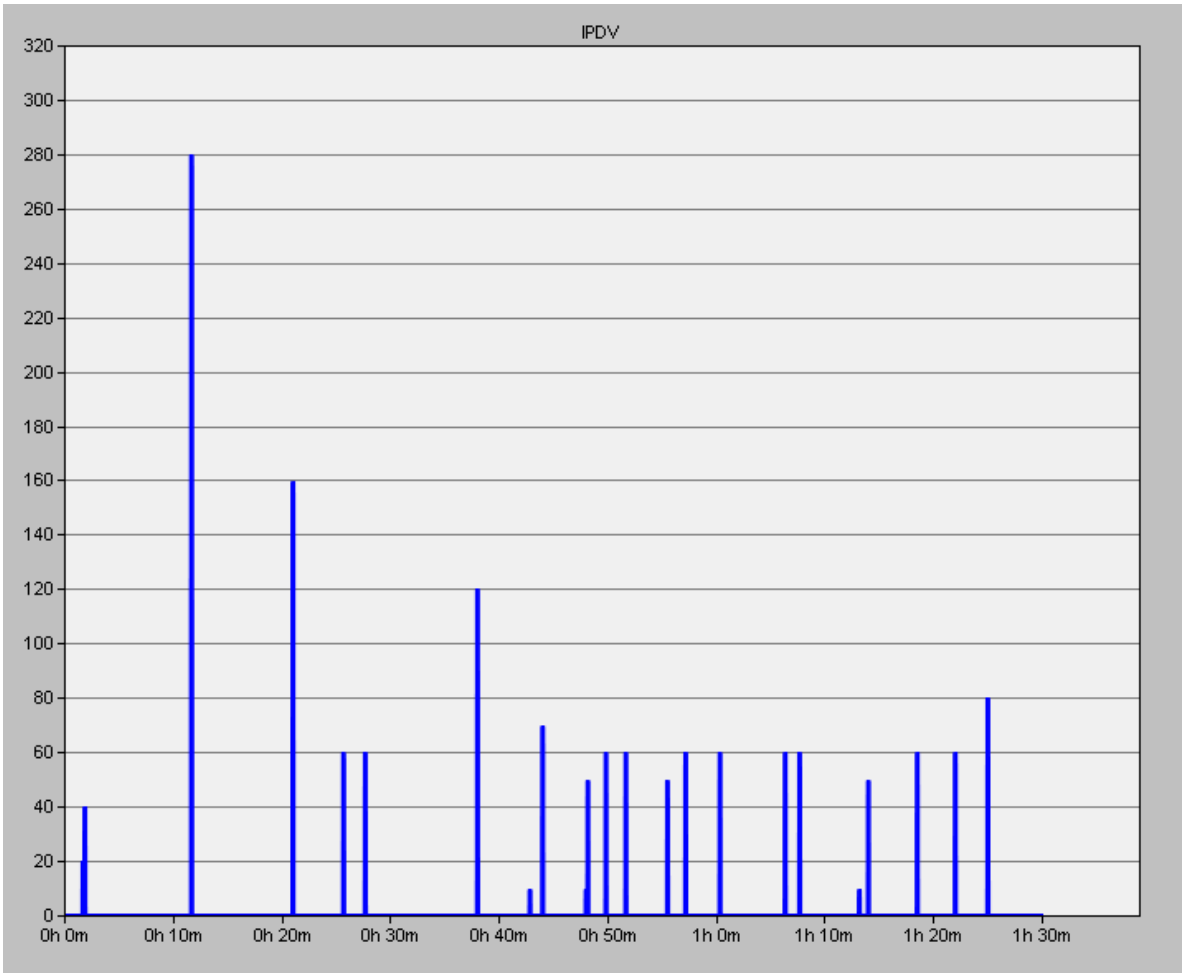
Anexo 67 Paquetes en nodos cóncavos GAF



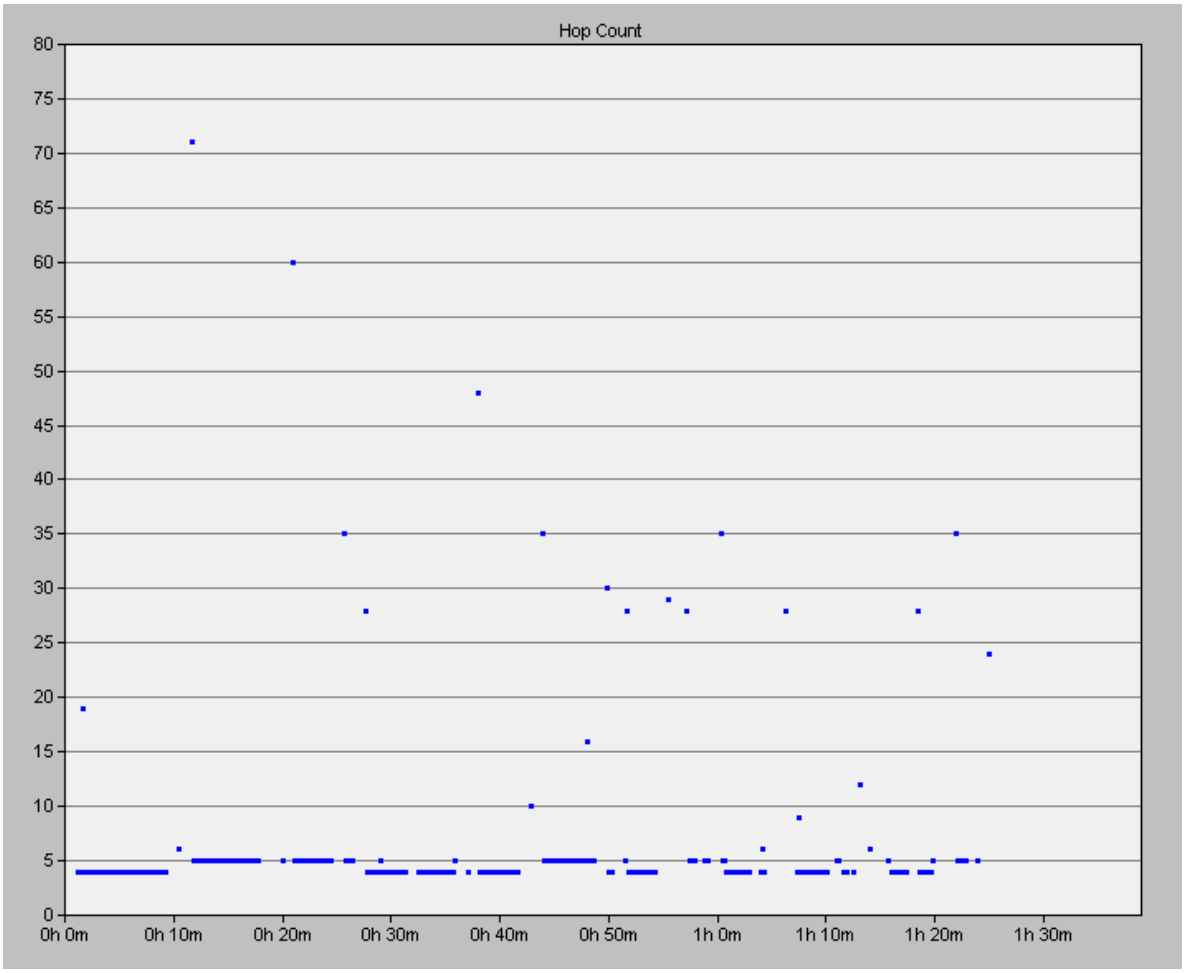
Anexo 68 Nodos Activos GAF



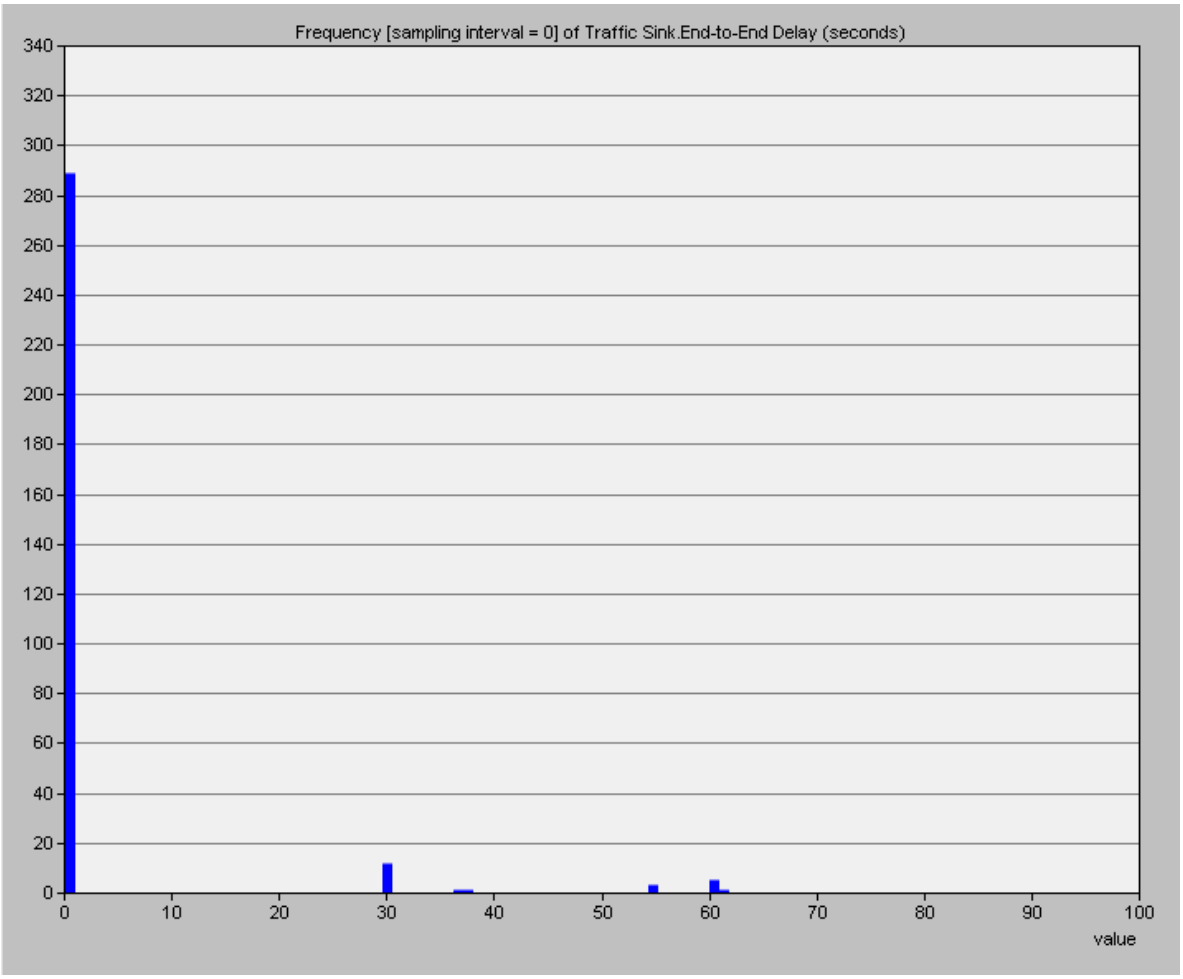
Anexo 69 IPDV GAF



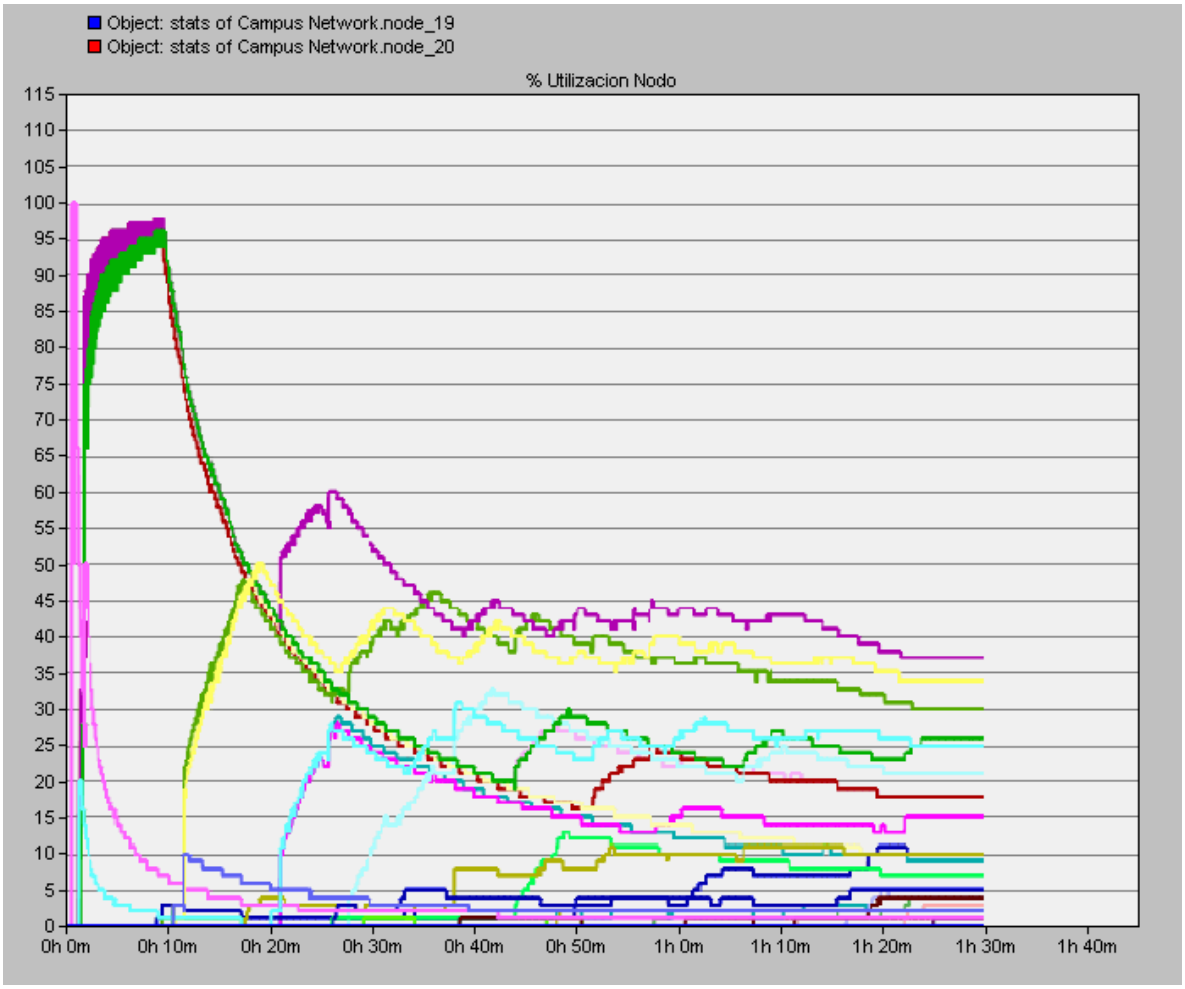
Anexo 70 *Hop Count* GAF



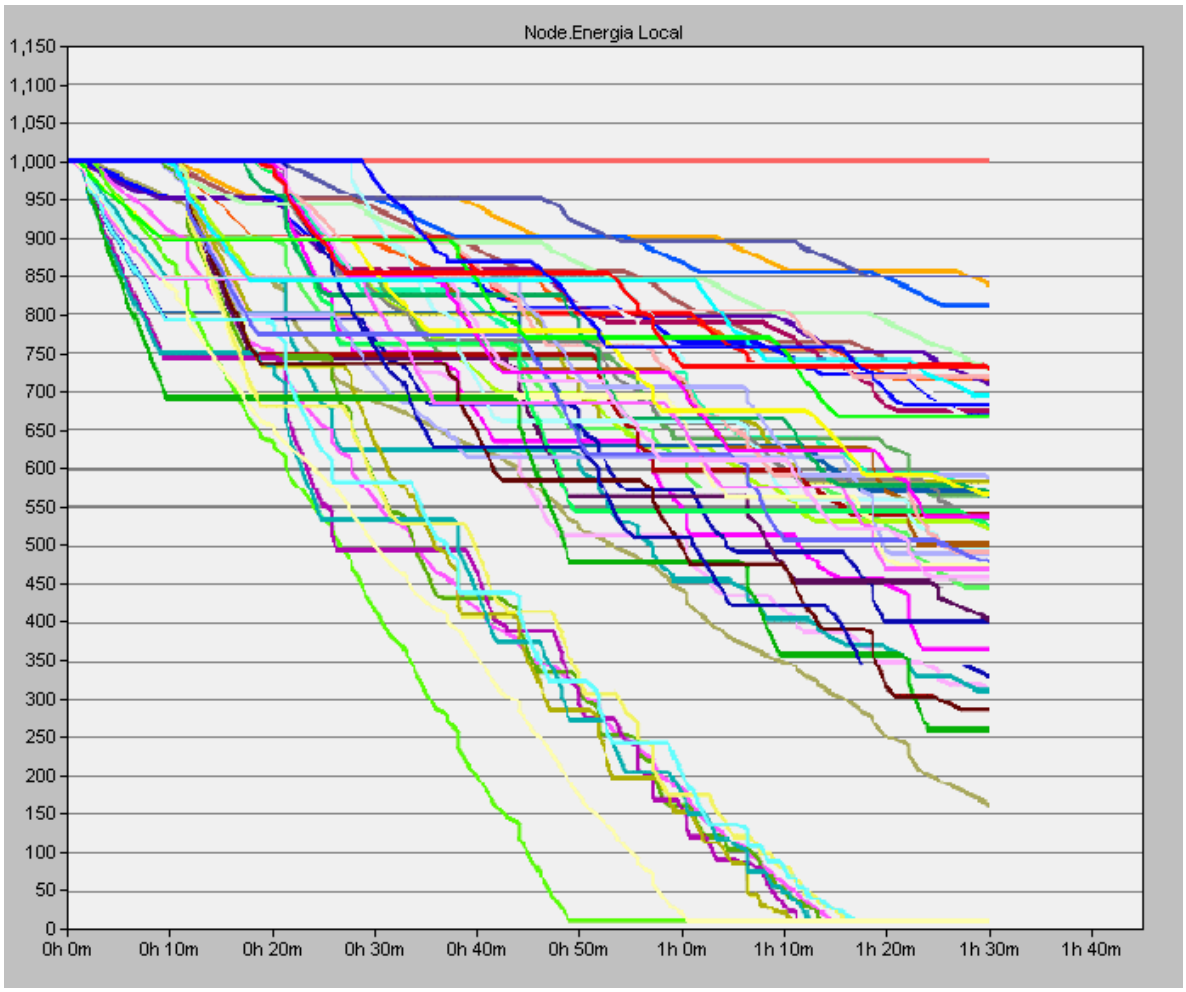
Anexo 71 *End to end Delay* GAF



Anexo 72 Porcentaje de utilización de los nodos GAF



Anexo 73 Energia de los nodos en GAF



Anexo 74 Porcentaje de utilización de la red