Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Estudio de sistemas adaptables de ruteo para sistemas distribuidos móviles

T E S I S

QUE PARA OPTAR POR EL GRADO DE:
D O C T O R A E N C I E N C I A S

P R E S E N T A:

A R E L L A N O V Á Z Q U E Z M A G A L I

DIRECTOR DE TESIS:
DR. HÉCTOR BENÍTEZ PÉREZ
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

MÉXICO, D.F. Agosto, 2015

Dedico este trabajo a mi familia, en especial a mis padres:
Mary y *Alberto*.

A mi hermana *Blanca*.

A mi tío *†Alberto* .

A mis amigos: *Caro, Casandra, Maricruz, Melina, Pilar, Suyin*,
Carlos Humberto, Cesarunix, Daniel , Iván, Johnny y *Toño*.

Ohana significa familia y tu familia
nunca te abandona ni te olvida.

— Lilo

Puedes llegar a cualquier parte,
siempre que andes lo suficiente.

Lewis Carroll

AGRADECIMIENTOS

Al Dr. Héctor Benítez Pérez por la confianza en la idea inicial y por apoyo a lo largo de la realización del trabajo de maestría y doctorado.

A los Drs. Jorge Ortega Arjona, Fabian García Nocetti, Gerardo Espinosa Pérez y Javier Gómez Castellanos por el tiempo y dedicación para revisar este trabajo de tesis.

A Dra. Magdalena Hérnandez y M.C. Mirella Ramirez por su asesoría y apoyo incondicional en cuestiones académicas y personales.

A mis compañeros de cubículo M. en I. José Ángel y Ing. José Alfredo por su amistad, apoyo moral y técnico.

A los revisores no oficiales de la la tesis y artículos: M. en I. Maricruz López Torres, Dra. Raquel Perales, Dr. Carlos Piña, Dr. César Daniel Estrada y M.C. Carlos Minutti.

A los ingenieros Joel Durán y Adrían Durán por el apoyo técnico.

Al Dr. Miguel Robles, a la Dra. Ma. Elena Lárraga y la facultad de ciencias UAEM por haberme impulsado a cursar estudios de posgrado.

A mis alumnos de la UAEM, por darle sentido a lo aprendido estos años.

A la UNAM por brindarme una excepcional vida académica y cultural.

Al proyecto PAPIIT UNAM INI008133 y CONACYT 176556 por el financiamiento recibido.

A CONACYT por la beca asignada para mi manutención durante mis estudios de posgrado [1] .

Al personal administrativo del Posgrado en Ciencia e Ingeniería de la Computación por su excelente labor.

A todos ustedes
Mag ='-'=
Agosto 2015

[1] No. de becario: 265975

Resumen

Un sistema distribuido móvil (SDM) está compuesto de dispositivos con memoria local, capacidad de comunicación inalámbrica, fuente de poder independiente, entre otros; comúnmente sin dependencia de una infraestructura dada, actuando como nodos. Cada nodo expone no sólo su posición relativa respecto al resto de los dispositivos móviles, también su estado, además puede actuar como router. Generalmente un SDM se describe como una topología dinámica porque cambia con el movimiento, entrada y salida de nodos del rango de alcance del propio SDM. Tal comportamiento dinámico hace difícil establecer y mantener una ruta de comunicación entre 2 nodos presuntamente remotos, puesto que todos los nodos de la ruta podrían no estar disponibles para establecer comunicación posteriormente.

Los algoritmos tradicionales de encaminamiento consideran la red como un todo, buscando determinar una ruta óptima; a través un algoritmo de descubrimiento de rutas cada que la topología cambia, esto requiere generar constantemente un estado global del SDM.

En esta tesis se propone el **Algoritmo de Encaminamiento por Consenso** (*AEC*) para la búsqueda de rutas, que toma en cuenta las características de los SDM, tales como el estado del planificador de tareas de cada nodo, la disponibilidad del canal de comunicación entre un par de nodos, la movilidad, así como distancia entre nodos, con el objetivo de reducir la inundación de la red. La propuesta incluye versiones para nodos estáticos y para nodos móviles, con el fin de estudiar el comportamiento en ambos casos.

El algoritmo obtiene una ruta a través de una topología dinámica, estableciendo sucesivos consensos entre nodos. Se delimitan grupos locales, cada nodo obtiene información de encaminamiento de los nodos vecinos, mientras se mantenga una conexión activa. Esta estructura de comunicación proporciona resistencia a los cambios locales de la topología dinámica: cualquier cambio se mantiene en una escala local del SDM; esto significa que sólo afecta a un subconjunto del total de nodos, en el que están involucrados como sistema distribuido sin afectar a todo el SDM. Por lo tanto, no es necesario recalcular toda la ruta, porque sólo estos grupos de nodos son afectados por el cambio. Para calcular el costo de enviar paquetes entre 2 nodos, las métricas de encaminamiento se definen en términos de cuánto tiempo se tarda en entregar un paquete a través de una ruta determinada. Normalmente, cada algoritmo de encaminamiento tiene sus propios indicadores.

La métrica propuesta para *AEC* es una función de disponibilidad que evalúa las condiciones locales de la red (en algunos enlaces en particular) y el estado del nodo en turno. Al decir que esta métrica evalúa localmente, se refiere a que sólo se evalúan las condiciones de la red de un nodo y sus vecinos, así como de los enlaces de comunicación existentes entre ellos.

El algoritmo de encaminamiento por consenso usa encaminamiento de origen dinámico entre nodos que quieren establecer comunicación. El algoritmo no usa mensajes de advertencia para indicar cambios en la red , por lo tanto se reduce el consumo de ancho de banda, particularmente cuando existen largos periodos sin cambios en la topología de la red.

Se realizaron simulaciones de redes de hasta 500 nodos, mostrando que *AEC* funciona mejor para redes menos de 100 nodos. En las simulaciones se comparó *AEC* contra el algoritmo de inundación, resultando que *AEC* encuentra rutas más cortas y menos costosas (de acuerdo a la función de disponibilidad), aunque el algoritmo de inundación es más

rápido. Se hicieron experimentos con dispositivos Arduino Yun, donde se implementó el algoritmo.

Índice general

Índice de figuras **xiii**

Índice de tablas **xvii**

Introducción **1**

1 Marco teórico **13**

1.1 Conceptos generales 13

1.2 Estados globales 15

1.3 Consenso 16

1.3.1 Algoritmos de consenso 18

1.4 Representación de redes 22

1.5 Algoritmos de rutas más cortas entre pares de nodos 24

1.5.1 Algoritmo de Bellman-Ford 24

1.5.2 Algoritmo de Dijkstra 26

1.6 Redes de comunicación 29

1.7 Encaminamiento 33

1.7.1 Tipos de encaminamiento 35

1.7.2 Ciclos de encaminamiento. 36

1.8 Resumen 38

2 Trabajo relacionado **39**

2.1 Algoritmos de encaminamiento 39

2.1.1 Algoritmos de encaminamiento adaptativo 40

2.1.2 Algoritmo de inundación 43

2.1.3 Encaminamiento Epidémico 46
2.2 Protocolos de encaminamiento 46
2.2.1 Vector de distancia (RIP) 47
2.2.2 Estado de enlace (OSPF) 49
2.2.3 TORA (*Temporally-Ordered Routing Algorithm*) 51
2.2.4 Virtual Ring Routing (VRR) 53
2.2.5 Grid 54
2.2.6 Dynamic Source Routing protocol (DSR) 56
2.2.7 Whirlpool Routing Protocol (WARP) 57
2.2.8 Encaminamiento adaptativo en MANETS basado en toma de decisiones 57
2.3 Algoritmos que utilizan consenso 58
2.3.1 Consenso en redes oportunistas 58
2.3.2 Consensus routing: The Internet as a distributed system . . 59
2.3.3 Algoritmo de votaciones por promedio ponderado 61
2.4 Resumen 63

3 Algoritmo de Encaminamiento por Consenso 65
3.1 Algoritmo para caso de estudio de topología "fija" 67
3.1.1 Versión de un salto 71
3.1.2 Versión de dos saltos 74
3.1.3 Versión de uno y dos saltos 77
3.2 Algoritmo para caso de estudio de topología móvil 80
3.3 Resumen 84

4 Experimentación 85
4.1 Descripción de las simulaciones 86
4.2 Especificaciones de las simulaciones para la versión de topología fija con variación de carga 87
4.3 Especificaciones de las simulaciones para la versión de topología móvil 89
4.4 Especificaciones de las simulaciones de influencia de la movilidad en el éxito del algoritmo 89
4.5 Especificaciones de las comparativas 90

4.5.1 Especificaciones de la comparativa *AEC* vs Algoritmo de Dijkstra . . . 91
4.5.2 Especificaciones de la comparativa *AEC* vs Inundación . . 91
4.6 Especificaciones técnicas de la experimentación en hardware . . . 92
4.6.1 Esquema centralizado . . . 95
4.6.2 Esquema distribuido . . . 97
4.7 Modulos de comunicación . . . 100
4.7.1 Programa principal . . . 105
4.7.2 Clase Vecino . . . 107
4.7.3 Biblioteca del algoritmo . . . 113
4.8 Resumen . . . 114

5 Resultados **117**
5.1 Simulación con matrices de valores fijos . . . 117
5.1.1 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto . . . 118
5.1.2 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de dos saltos . . . 118
5.1.3 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de uno y dos saltos . . . 121
5.2 Simulación con una matriz de carga variable . . . 125
5.2.1 Simulación con la matriz de carga variable para búsqueda de un salto . . . 125
5.2.2 Simulación con una matriz de carga variable para búsqueda de dos saltos . . . 126
5.2.3 Simulación con una matriz de carga variable para búsqueda de uno y dos saltos . . . 129
5.3 Ejemplo general de la versión de una matriz con valores fijos . . . 132
5.4 Simulación de la versión móvil. . . . 134
5.4.1 Algoritmo de inundación. . . . 136
5.4.2 Comparativa con el algoritmo de inundación. . . . 139
5.5 Experimentación en hardware: Arduino ™Yun . . . 141
5.6 Resumen . . . 148

6 Conclusiones **149**

Conclusiones **149**

Apéndices **153**

A Diagramas de Flujo **153**

B Ejemplos númericos **161**

B.1 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto 161

B.2 Simulación con una matriz de carga variable para búsqueda de un salto 169

Referencias **183**

Glosario **191**

Índice alfabético **194**

Índice de figuras

1.1 Representación del concepto de consenso. 17
1.2 Ejemplo de consenso regular o por inundación. 20
1.3 Ejemplo de consenso jerárquico. 21
1.4 Ejemplo de consenso aleatorio. 22
1.5 Red de 6 nodos. 24
1.6 Esquema de fragmentación de paquetes para ser enviados por la red [Moh98]. 30
1.7 Latencia de la comunicación. 32
1.8 Escenario para ciclos transitorios [HMMD02] 37

2.1 a) Inundación clásica . b) Inundación temporal. c) Inundación espacial. 45
2.2 Ejemplo de una red ejecutando RIP. 47
2.3 Relación entre el anillo virtual y la red física [CCN06]. 53
2.4 Grid. 55

3.1 Ejemplo de un grafo y los valores correspondientes a la función de disponibilidad. Se considera que el nodo $i = 1$, y $l -$ vecindario con $l = 1, J_1 = \{2, 5, 6, 9\}$. 69
3.2 Comportamiento de la función 3.3. 70
3.3 Ejemplo de SDM. 82

4.1 Nomenclatura de UML. 93
4.2 Caso de Uso. 94
4.3 Esquema físico. 95

4.4 Fases 1 y 2, esquema centralizado. 96
4.5 Fases 3 y 4, esquema centralizado 96
4.6 Fase 5, esquema centralizado 97
4.7 Topología de SDM de 24 nodos. 98
4.8 Formación de un grupo fuertemente conectado 99
4.9 Formación de grupos fuertemente conexos pero desconexos entre si. 100
4.10 Diagrama de actividad del estado del Nodo Fuente del caso de uso. 103
4.11 Diagrama de actividad del estado del Nodo Vecino del caso de uso. 104
4.12 Esquema real con 15 Arduinos Yun ®. 105
4.13 Simbología de los diagramas de funciones 107
4.14 Diagrama de la clase Vecino. 107
4.15 Diagrama de la relación de las funciones de la biblioteca Comunicación. 108
4.16 Diagrama de la relación de las funciones de la biblioteca Algoritmo. 113
4.17 Diagrama general. 115

5.1 Grafo utilizado para el ejemplo de la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de dos saltos. . 118
5.2 Ruta obtenida por la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de dos saltos. 121
5.3 Grafo utilizado como ejemplo para la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de uno y dos saltos. 122
5.4 Ruta obtenida por la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de uno y dos saltos. 125
5.5 Grafo utilizado para el ejemplo de la simulación con la matriz de carga variable para búsqueda de dos saltos. 126
5.6 Ruta obtenida de la simulación con la matriz de carga variable para búsqueda de dos saltos. 129
5.7 Grafo del ejemplo de la simulación con la matriz de carga variable para búsqueda de uno y dos saltos. 129
5.8 Ruta obtenida de la simulación simulación con la matriz de carga variable para búsqueda de uno y dos saltos. 132

5.9 Ejemplo de la versión móvil. 135
5.10 Comparativa de las rutas obtenidas por el algoritmo de a) Inundación y b) *AEC*, para la ruta $(25, 32)$. 140
5.11 Comparativa de las rutas obtenidas por el algoritmo de a) Inundación y b) *AEC*, para la ruta $(11, 31)$, el algoritmo de inundación no encuentra el destino. 141
5.12 A la izquierda los 15 Arduino ™Yun y a la derecha el esquema completo. La pantalla está conectada a la laptop y de esta se controlan todos los arduinos, via ssh. 142
5.13 Topología experimental para los dispositivos Arduino Yun. 143
5.14 Secuencia de búsqueda del nodo 3 desde el nodo 4 utilizando el algoritmo *AEC*. 144
5.15 Retardo de la obtención de información propia. 145
5.16 Retardo de la búsqueda del nodo objetivo en el vecindario inmediato. 145
5.17 Retardo de la activación del nodo ganador. 146
5.18 Secuencia de búsqueda del nodo 9 desde el nodo 2 utilizando el algoritmo *AEC*. 147
5.19 Retardo de la obtención de información propia. 147
5.20 Retardo de la búsqueda del nodo objetivo en el vecindario inmediato. 148
5.21 Retardo de la activación del nodo ganador. 148

A.1 Diagrama de flujo del programa principal. Este módulo manda a llamar el resto de las funciones. 154
A.2 Diagrama de flujo de la función obtiene_ganador donde se obtiene el resultado del proceso de consenso. 155
A.3 Diagrama de flujo de la función cosvec donde se implementa la búsqueda de vecinos. 156
A.4 Diagrama de flujo de la función costo, usada para calcular el costo de cada enlace. 157
A.5 Diagrama de flujo de la función calcula_vecinos. 158
A.6 Diagrama de flujo del programa principal de la versión móvil. . . 159
A.7 Diagrama de flujo de la función mueve nodos, donde determina la dirección a la que se moverán los nodos móviles. 160

B.1 Grafo utilizado en la simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto. 162
B.2 Ruta obtenida en la simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto. 169
B.3 Grafo utilizado en simulación con una matriz de carga variable para búsqueda de un salto. 170
B.4 Ruta obtenida en la simulación con una matriz de carga variable para búsqueda de un salto. 182

Índice de tablas

1.1 Ejemplo de tabla de encaminamiento. 34

3.1 Representación matricial del SD. Ejemplo: $i = 1, l = 1; J_1 = \{2, 5, 6, 9\}$ 72
3.2 Representación matricial del SD 75
3.3 Representación matricial del SD. Ejemplo: $i = 1, l = 2, J_1 = \{2, 5, 6, 9\}$ 78

4.1 Especificaciones de la versión de un salto. 88
4.2 Especificaciones de la versión de dos saltos. 88
4.3 Especificaciones de la versión de uno y dos saltos. 89
4.4 Especificaciones de las simulaciones de la versión móvil. 90
4.5 Especificaciones de las simulaciones de influencia de la movilidad. 90
4.6 Especificaciones de la comparativa *AEC* vs Algoritmo de Dijkstra. 91
4.7 Especificaciones de la comparativa de inundación. 92
4.8 Lista de dispositivos Arduino ™Yun. 106

5.1 Resultados de la versión de un salto para encontrar la ruta entre el nodo 85 y el 234. 133
5.2 Resultados de la versión de dos saltos para encontrar la ruta entre el nodo 85 y el 234. 134
5.3 Resultados de la versión de uno y dos saltos para encontrar la ruta entre el nodo 85 y el 234. 134
5.4 Resultados de la versión móvil del algoritmo. 136

5.5 Resultados de la implementación del algoritmo de “inundación ”, rutas obtenidas entre el nodo 81 y el 75. 138
5.6 Resultados de la Prueba 1. 140
5.7 Resultados de la Prueba 2. 141

Hazlo o no lo hagas, pero no lo intentes.

Yoda

Introducción

Un sistema distribuido móvil (SDM) está compuesto de dispositivos con memoria local, capacidad de comunicación inalámbrica, fuente de poder independiente, entre otros; comúnmente sin dependencia de una infraestructura dada, actuando como nodos. Cada nodo expone no sólo su posición relativa respecto al resto de los dispositivos móviles, también su estado, además puede actuar como router: en un SDM cada nodo se comunica con sus vecinos no sólo como emisor y receptor, sino como router, enviando mensajes a todos los que no están en el rango de transmisión del emisor.

Generalmente un SDM se describe como una topología dinámica porque cambia con el movimiento, los nodos pueden entrar o salir del rango de alcance del propio SDM. Tal comportamiento dinámico hace difícil establecer y mantener una ruta de comunicación entre 2 nodos presuntamente remotos, puesto que todos los nodos de la ruta podrían no estar disponibles para establecer comunicación posteriormente.

Problema

Los algoritmos tradicionales de encaminamiento consideran la red como un todo, buscando determinar una ruta óptima entre 2 puntos de ésta; se utiliza un algoritmo de descubrimiento de rutas cada que la topología cambia, esto requiere generar constantemente un estado global del SDM.

En general, los protocolos de encaminamiento sólo inician la búsqueda de rutas si se requiere. La mayoría de los algoritmos de encaminamiento utilizados para las

topologías similares a los SDM hacen uso de la inundación como mecanismo de búsqueda de rutas [SK04].

Inundación significa que cada nodo en un SDM retransmite todos los mensajes entrantes a los nodos dentro de su vecindario, con la única restricción que el mensaje sólo puede ser enviado una vez por cada nodo a su vecindario, es decir aunque un nodo reciba el mismo mensaje de más de un nodo emisor, sólo lo retransmitirá una vez. La inundación es fácil de implementar; sin embargo representa un enfoque de fuerza bruta. Por lo tanto es ineficiente porque incrementa el tráfico en la red de manera exponencial. Dado que la inundación consume de manera sustancial el ancho de banda disponible, es importante reducir la frecuencia de descubrimiento de rutas, por lo tanto la inundación de la red.

Los protocolos de encaminamiento convencionales no están diseñados para los cambios en la topología dinámica. En ambientes de red , los enlaces entre nodos presentan fallas y en ocasiones el costo del enlace aumenta debido a la congestión, sin embargo, los routers no se mueven. En un ambiente de nodos funcionando como routers la convergencia a una ruta nueva puede ser lenta, sobre todo al utilizar algoritmos basados en vector de distancia [JM96a] .

Los algoritmos de encaminamiento utilizados en las redes Ad Hoc, como AODV [Sun97],TORA [PC97] y OLSR [JMC$^+$01] no son factibles para resolver el problema de los SDM, porque no consideran el estado de los planificadores de tareas locales de cada nodo. Otros algoritmos conocidos como wormhole [Sey98], o algoritmos basados en vector de distancia o estado de enlace [CSRL01, MR07]) tampoco pueden ser usados, puesto que reservan el canal y lo mantienen, esto no es una opción a considerar para una topología variable no son una opción a considerar para la construcción de ruta. .

En general, cuando los nodos se mueven lo suficientemente rápido y con relativa frecuencia , la mejor estrategia de encaminamiento es inundar la red, con la esperanza de que el nodo en movimiento sea alcanzado por una de las muchas copias. Por otro lado , cuando el movimiento del nodo es muy lento o poco frecuente, no se da la sobrecarga por los nulos cambios en la información de encaminamiento. El encaminamiento de origen dinámico se mueve entre estos 2 estados, provocado por la velocidad con la que se solicitan las rutas. Si un nodo con el que otro nodo quiere comunicarse se está moviendo muy rápidamente, el encaminamiento

de origen dinámico inunda el red con una solicitud de ruta y a continuación, envía un paquete de datos tan pronto como se encuentra el destino y una ruta responde [JM96a].

Objetivo

En esta tesis se propone el **Algoritmo de Encaminamiento por Consenso** (*AEC*) para la búsqueda de rutas, que tome en cuenta las características de los SDM, tales como el estado del planificador de tareas de cada nodo, la disponibilidad del canal de comunicación entre un par de nodos, la movilidad, así como distancia entre nodos, con el objetivo de reducir la inundación de la red. La propuesta incluye versiones para nodos estáticos y para nodos móviles, con el fin de estudiar el comportamiento en ambos casos.

Se plantea que el algoritmo obtenga una ruta a través de una topología dinámica, estableciendo sucesivos consensos entre nodos. Se delimitarán grupos locales, cada nodo obtendrá información de encaminamiento de los nodos vecinos, mientras se mantenga una conexión activa. Esta estructura de comunicación proporciona resistencia a los cambios locales de la topología dinámica: cualquier cambio se mantiene en una escala local del SDM, esto significa que sólo afecta a un subconjunto del total de nodos, en el que están involucrados como sistema distribuido sin afectar a todo el SDM. Por lo tanto, no es necesario recalcular toda la ruta, porque sólo en estos grupos de nodos son afectados por el cambio. Para calcular el costo de enviar paquetes entre 2 nodos, las métricas de encaminamiento se definen en términos de cuánto tiempo se tarda en entregar un paquete a través de una ruta determinada. Normalmente, cada algoritmo de encaminamiento tiene sus propios indicadores, que asignan un valor a una ruta, sobre la base de un factor como el número de saltos, ancho de banda, fiabilidad de enlace, etc.

La métrica propuesta es una función de disponibilidad que evalúa las condiciones locales de la red (en algunos enlaces en particular) y el estado del nodo en turno. Al decir que esta métrica evalúa localmente, se refiere a que sólo se evalúan las condiciones de la red de un nodo y sus vecinos, así como de los enlaces de comunicación existentes entre ellos. Es necesario cuantificar los costos de las rutas, ya que puede formarse más de una. Los saltos entre nodos serán considerados

como una de las unidades de medida. Esta unidad de medida puede ser limitada, al definir un vecindario, i.e., es posible que el algoritmo sólo calcule rutas en un $l - vecindario$, donde l es el número de nodos intermedios y k es la longitud de la ruta desde el nodo fuente hasta el nodo destino.

El algoritmo de encaminamiento por consenso usa encaminamiento de origen dinámico entre nodos que quieren establecer comunicación. El algoritmo no usa mensajes de advertencia para indicar cambios en la red , por lo tanto se reducirá el consumo de ancho de banda, particularmente cuando existen largos periodos sin cambios en la topología de la red.

Las principales dificultades prácticas a resolver son:

1. Para llevar a cabo el consenso se requiere que todos los participantes tengan el mismo número de datos. Por lo tanto el algoritmo descartaría a los segmentos del Sistema Distribuido (SD) que no se encuentren fuertemente conexos e incrementa la probabilidad de que se presenten ciclos de encaminamiento. Por lo que la topología ideal sería a todos los nodos muy cercanos y aun así los nodos de las orillas serían descartados.

2. Soportar la desincorporación de los nodos del SDM, principalmente los que sean vecinos de nodos con cardinalidad baja, porque al darse esta situación un grupo de nodos quedaría aislado del resto, sin posibilidad de conexión.

3. Distribución de información, distribuir el cálculo del consenso.

4. Escalabilidad, a pesar de que el número de nodos crezca que el tiempo que tome encontrar la ruta al destino tarde una cantidad de tiempo conveniente.

Hipótesis

En un SDM se puede calcular una ruta móvil, global y concurrente , en un contexto de cambio de topología, a partir de establecer consensos sucesivos entre grupos locales de nodos.

Meta

Dado lo anterior se propone como meta establecer una ruta a través de la red basándose en datos locales y que esta sea óptima para las condiciones particulares de la red y de cada nodo en cada instante t. De esta manera obtener una ruta que es segura y resistente a los cambios en la red, puesto que los nodos de la que estará conformada serán elegidos por tener las mejores condiciones para retransmitir mensajes en el tiempo t para cada paso de la ruta.

Contribuciones

Por las características del algoritmo, no se usan mensajes periódicos de actualización de rutas, lo que reduce el consumo de ancho de banda, sobre todo en periodos donde la movilidad es poca o nula. Se evitará la redundancia de rutas.

El hecho de utilizar un algoritmo de consenso para el cálculo de una nueva ruta permite obtener una ruta que no se afectará con cambios menores alejados de su alcance y que el cálculo de cada paso de esta ruta será la mejor opción para las condiciones particulares del SDM.

Aplicaciones

Este algoritmo puede ser usado para comunicación y trazado de rutas entre agentes móviles, para aplicaciones de tráfico para teléfonos móviles, así como en otras situaciones que se requiera reducir la inundación del medio inalámbrico, con pocos nodos y se cuente con capacidad de procesamiento. Como en la toma de decisiones para vuelo en grupo o distribución de procesos en un cluster de alto rendimiento.

Antecedentes

La revisión bibliográfica sobre el encaminamiento de paquetes en redes inalámbricas de dispositivos móviles se remonta al menos a 1973, cuando la Agencia de Defensa, en su división de Proyectos de Investigación Avanzada de EE.UU. comenzó la Red de Radio de Paquetes DARPA (PRNET) proyecto del proyecto [JT87]

y su sucesor: las redes adaptables con funciones de supervivencia (SURAN) [Ste95]. PRNET soportaba la configuración automática y el mantenimiento de las vías de comunicación de conmutación de paquetes en una red de moderada movilidad que se comunican a través de radios. El protocolo de encaminamiento PRNET utilizaba una forma de encaminamiento de vector de distancia, incluyendo en cada nodo de la transmisión un paquete de actualización de encaminamiento cada 7,5 segundos (llamado un paquete de organización de paquetes de radio o PROP, en PRNET). El encabezado de cada paquete de datos contenia las direcciones de los nodos del origen y del destino, el número de saltos desde la fuente hasta el momento, y el número de saltos que faltaban hasta llegar al destino (con base en la tabla de encaminamiento del remitente). Los nodos recibían de forma promiscua todos los paquetes y podían actualizar sus tablas de encaminamiento basándose en la información de encabezado, utilizando el protocolo de enlace de datos *hop-by-hop* y confirmación de recibido , ya sea utilizando reconocimientos explícitos (activo) o reconocimientos pasivos contra estos paquetes recibidos.

También, la red de paquetes de radio Amateur originalmente sólo se utilizaba como única fuente de encaminamiento con rutas de origen explícitas construidas por el usuario, aunque algunos habían considerado la posibilidad de un esquema de direccionamiento de origen más dinámico. Se desarrolló un sistema conocido como NET/ROM para permitir que las decisiones de encaminamiento fueran automatizadas, utilizando una forma de protocolo de encaminamiento por vector de distancia en lugar de encaminamiento de origen. NET/ROM también permitía la actualización de la tabla de encaminamiento, basada en la información de la dirección de origen en las cabeceras de los paquetes que recibe (citado en [JM96a]).

Un problema particular con el uso de los protocolos de encaminamiento de vector de distancia en redes con nodos móviles, es la posibilidad de formar ciclos de encaminamiento. Con el fin de eliminar esta posibilidad, Perkins y Bhagwat propusieron la adición de números de secuencia para las actualizaciones de encaminamiento en el “Destination-Sequenced Distance Vector ” (DSDV) [PB94]. Estos números de secuencia se utilizan para comparar la edad de la información en un actualización de encaminamiento, y permitir a cada nodo seleccionar preferentemente rutas en base a la información más actualizado. DSDV también utiliza actualizaciones de encaminamiento activas para acelerar la convergencia de rutas.

Con el fin de amortiguar la fluctuación de ruta y reducir la congestión de un gran número de actualizaciones desencadenadas después de que una ruta cambia, cada nodo mantiene DSDV información sobre la frecuencia con la que se ve a los cambios de ruta y puede retrasar algunas actualizaciones de encaminamiento.

Existe una gran cantidad de trabajos sobre protocolos de encaminamiento en medios inalámbricos. Estos protocolos se pueden clasificar en cinco tipos: reactivos, proactivos, híbridos, jerárquicos, y basados en coordenadas. Los protocolos reactivos realizan descubrimiento de rutas sobren demanda por medio de inundaciones de la red y que retrasan los paquetes hasta que las rutas se establecieron. Por ejemplo, AODV [Sun97], DSR [JM96b] y TORA[PC97] son protocolos reactivos. Los protocolos proactivos mantienen las rutas entre todos los pares de nodos. Se inundan con paquetes de información a través de la red siempre que existan cambios en la topología, pero no incurren en demoras o gastos generales para visualizar cubrir rutas sobre demanda. DSDV [PB94], OLSR [JMC$^+$01], y WRP [MGLA96] son ejemplos de protocolos proactivos. En general, los protocolos proactivos funcionan bien en escenarios estáticos mientras que los protocolos reactivos funcionan mejor en escenarios móviles.

Los protocolos de encaminamiento híbridos han sido propuestos para superar los problemas de desempeño causados por los frecuentes descubrimientos de rutas utilizados en los protocolos de encaminamiento proactivos. Los protocolos de encaminamiento híbridos incorporan características de los protocolos reactivos y proactivos [PH06]. Sin embargo, para cada solicitud de ruta, los protocolos híbridos no inundan completamente la red, es difícil balancear entre el costo de intercambiar información periódicamente (i.e. proactividad), con el descubrimiento de rutas usando inundación controlada (i.e. reactividad) [RHS03]. Otros protocolos, como los geográficos o “bypass” [SK04], reducen la frecuencia de inundación permitiendo algún nodo retransmisor que comience una limitada búsqueda de rutas en el caso de que una ruta falle [Toh97, Sun97], o emplear un mecanismo local de reparación de ruta [SGLA01, AGG00]. Los protocolos de encaminamiento híbridos, ya sea que utilicen inundación controlada o reparación de rutas, están enfocados en reducir la perdida de paquetes y no a la utilización eficiente del ancho de banda durante la recuperación de la ruta. Sin embargo, a causa de la movilidad, los pro-

tocolos multiruta causan perdidas de paquetes adicionales y retrasos, debido a la dependencia de las rutas potencialmente obsoletas en los cachés [MD05].

Los protocolos híbridos como ZRP [HPS02] y SHARP [RHS03] pueden lograr un buen rendimiento a través de una gama más amplia de escenarios mediante la combinación de ambos componentes. Ellos dividen la red en zonas. Los nodos mantienen las rutas de manera proactiva dentro de su zona por medio de inundar cuando hay cambios en la topología. Las rutas entre esas zonas son descubiertas sobre demanda con un mecanismo optimizado de inundación.

Los protocolos jerárquicos y basados en coordenadas no inundan la red. Por ejemplo, Lanmar [PGH00] y L + [CM02] son protocolos jerárquicos, y GPSR [KK00] y BVR [FFR$^+$04] protocolos están basados en coordenadas. Ellos usan las direcciones de ubicación dependiente de ruta. Estos identificadores pueden cambiar con la movilidad y, en algunos protocolos, con la congestión y fallos [CM02] [PGH00], [FFR$^+$04]. Por lo tanto, estos protocolos utilizan tanto un identificador fijo y una dirección de ubicación-dependiente para cada nodo y por lo general requieren mecanismos para buscar la ubicación de un nodo dado su identificador fijo [DPH05]. Estos mecanismos reducen la resistencia a fallos y aumentan la complejidad.

Otro protocolo de ruteo llamado VRR [CCN06] combina aspectos de ruteo proactivo, establece y mantiene rutas, sin utilizar inundación por medio de combinar direccionamiento utilizando un anillo lógico y un mapa físico. Utiliza un esquema de números aleatorios para identificar nodos entre 0 y un límite superior finito y conocido, esto implica la necesidad de contar con un mecanismo de consenso para asignar los números a cada nodo o la existencia de un nodo centralizado que se encargue de la asignación de identificadores de forma global. Lo que restringe la escalabilidad del protocolo.

Existe una investigación realizada en 2008 sobre un protocolo de encaminamiento basado en matrices de adyacencia en redes móviles Ad Hoc. Esta investigación propone la creación de un nodo agente que recopila la información la topología de la red y crea la matriz de adyacencia de ésta [LKLK08].

El nodo agente es un nodo diseñado para implementar el protocolo basado en matrices de adyacencia. Este nodo tiene la misma tabla de encaminamiento para otros nodos. El nodo agente es el que inicia el proceso de descubrimiento de la

topología y puede agrupar la información y enviar a todos la información de la topología de red, en la forma de una matriz de adyacencia [LKLK08].

En una red Ad Hoc, todos los nodos periódicamente mandan un mensaje "Hello", para revisar su conectividad con los nodos vecinos. En este esquema, cada nodo mantiene una tabla de la información de los vecinos. El nodo agente hace un *broadcast* con una petición a todos los nodos vecinos para que estos a su vez los manden a sus vecinos, así sucesivamente. Cuando se acaban los vecinos siguientes, se manda un mensaje de regreso con la dirección del nodo y la matriz de adyacencia con sus vecinos, al recibir el mensaje de respuesta, el nodo agente puede obtener la matriz de adyacencia de un salto de cada nodo en toda la red. Puede suceder que un nodo reciba más de un mensaje, en este caso, el nodo checa el número de secuencia del mensaje y almacena o actualiza la matriz de adyacencia con el número de secuencia mayor [LKLK08].

Un trabajo importante en el área se refiere a algoritmos de encaminamiento que intentan evitar o disminuir el uso de mensajes de difusión (broadcast), como el caso del algoritmo de Bypass Routing Algorithm [SK04] que se comporta como un mecanismo de recuperación de rutas y recuperación local de errores. Esencialmente, recupera desde una ruta fallida, primero el nodo busca una ruta alternativa en su memoria cache de rutas. Si la ruta existe, el nodo parcha la ruta rota con la ruta alternativa. Si no es posible la recuperación, se inicia la búsqueda de vecinos para encontrar un enlace que comunique con el destino (usando broadcast reducido). Cuando la respuesta llega, el nodo repara las rutas afectadas por el enlace fallido con la información de conexión detallada.

Existen otros algoritmos que utilizan un sistema de localización geográfico para poder determinar la dirección hacia donde se encuentra el nodo destino y así poder dirigir la ruta hacia el destino [ZWXS11].

Algunos autores han propuesto usar algoritmos de consenso en sistemas distribuidos para encontrar un encaminamiento, se han hecho trabajos usando consenso lógico. En el trabajo descrito en [FMBB10] se utiliza una vector de eventos que influyen en la conectividad, cada nodo es capaz de evaluar estos eventos, así como su estado anterior al evento y en base a esto tomar una decisión, en versiones centralizadas y distribuidas. Los trabajos previos a este articulo en los que se plantean las bases del consenso lógico son [FB13] y [FDB11].

Se han incorporado mecanismos de consenso en protocolos de ruteo. Por ejemplo en [JKBK+08] separan el envío de paquetes en dos distintos modos lógicos; un modo estable que asegura que la ruta es elegida sólo después de llegar a un acuerdo de un estado consistente de una vista global. Un modo transitivo que asegura alta disponibilidad cuando un paquete se encuentra con un router que no pose una ruta estable, ya sea porque existe una falla en el enlace o porque el algoritmo aún no termina de calcular una ruta estable. El algoritmo de consenso se usa para calcular un estado global del sistema a partir de las instantáneas distribuidas que cada sistema autónomo envía.

AEC representa un espacio único de diseño de protocolos de encaminamiento. Cada nodo de *AEC* mantiene información mínima de si mismo y de su vecindario más cercano. Se propone utilizar un algoritmo de consenso para la búsqueda de rutas en un SDM, con el propósito de obtener una decisión local, de esta manera reducir el costo de la búsqueda de rutas y sin tener un estado global del sistema. En este sentido *AEC* es en parte proactivo. Al tener sólo información local en cada nodo se evita inundación a gran escala.

El diseño de *AEC* se inspira en el uso de una función de disponibilidad para cada nodo [AV11] y la implementación de un algoritmo de consenso basado en votantes de promedio ponderado [LSBB01], para obtener un resultado local a la cada etapa de la construcción de la ruta. Se diseñó un esquema centralizado y otros distribuido basados en [FB13].

Organización de la tesis

Este trabajo se divide en 7 capítulos, 2 apéndices. En el capitulo 1, se planea la motivación del trabajo de tesis, se expone la hipótesis en que se basa esta investigación, así como el objetivo, alcance y la contribución que tiene este trabajo al área de redes y sistemas distribuidos.

El capitulo 1, presenta el marco teórico donde se desarrolla este trabajo, incluye conceptos básicos de matemáticas discretas y teoría de grafos, así como conceptos generales de redes y algoritmos de encaminamiento.

El capitulo 2 expone el trabajo relacionado que fue revisado y analizado.

El capitulo 3 describe el algoritmo de encaminamiento por consenso, sus versiones.

En el capitulo 4 se presenta la descripción de las simulaciones y experimentos realizados, así como algunas comparativas con algoritmos conocidos.

En el capitulo 5 se presentan y discuten los resultados obtenidos en la experimentación. Y finalmente las Conclusiones. Se incluyen 2 apéndices, el apéndice A incluye los diagramas de flujo de cada función del algoritmo de encaminamiento por consenso. El apéndice B muestra ejemplos numéricos del funcionamiento de las simulaciones realizadas en el capitulo 4.

Aprendí muy pronto la diferencia entre conocer el nombre de algo y saber algo.

Richard Feynman

CAPÍTULO

1

Marco teórico

El rápido crecimiento de las redes inalámbricas y las comunicaciones móviles han limitado la disponibilidad de las bandas del espectro por la creciente demanda. Los algoritmos de encaminamiento deben adaptarse a las nuevas necesidades que han surgido con la incorporación de dispositivos inalámbricos, las nuevas características de estos, su movilidad y capacidades de procesamiento.

En una red inalámbrica los problemas de conectividad se agravan conforme el número de nodos en la red incrementa. Se presentan problemas como los son: la interferencia entre nodos, el número de posibles rutas y el calculo constante de rutas.

Para la mejor compresión de este trabajo se introducen algunas definiciones básicas de teoría de grafos. La *topología* de la red es determinada por el subconjunto de nodos activos y el conjunto de enlaces activos a lo largo de los cuales la comunicación directa puede ocurrir.

1.1 Conceptos generales

Sea G un grafo $G = (V, E)$ que representa una red (donde V es el conjunto de todos los nodos de la red y hay una arista $(v_1, v_2) \in E \subseteq V^2$ si y solo si los nodos

v_1 y v_2 pueden comunicarse directamente uno con el otro) y se transforma en un grafo $T = (V_T, E_T | V_T \subseteq V, E_T \subseteq E)$ [KW05].

DEFINICIÓN 1 ***Grado de un vértice.***

Sea G un grafo y $v \in V$. El número de aristas incidentes a v en G es llamado grado (o valencia) de un vértice $v \in G$, y se denota como $d_G(v)$, o simplemente $d(v)$ cuando G no necesita referencia explicita. Cualquier ciclo en v se cuenta 2 veces cuando se calcula el grado de v. El mínimo y máximo grado de los vértices del grafo G se denota respectivamente como $\delta(G)$ o simplemente δ (para el mínimo), y $\Delta(G)$ o simplemente Δ (para el máximo) [BR00].

DEFINICIÓN 2 ***Recorrido y rutas.*** *Sea un recorrido (walk) de longitud h desde un nodo i al nodo j es una sucesión de h aristas de la forma $(n_0 \to n_1)(n_1 \to n_2) \dots (n_{h-1} \to n_h)$, donde $n_0 = i$ y $n_k = j$. Una ruta es un recorrido en la cual todos los vértices son diferentes,i.e., $n_l \neq n_m \; \forall \; 0 \leq l \neq m \leq h$. Un recorrido cerrado de longitud h, también llamada un ciclo de longitud h, es un recorrido que empieza en el nodo i y regresa después de h saltos al mismo nodo i [Mie11].*

DEFINICIÓN 3 ***Matriz de adyacencia.***

La matriz de adyacencia A de un grafo G con N nodos es una matriz de $N \times N$ con elementos $a_{ij} = 1$ si solo y si el par de nodos (i, j) es conectado por un enlace de G, de otra manera $a_{ij} = 0$. Si el grafo es no dirigido, la existencia de un enlace implica que $a_{ij} = a_{ji}$, la matriz de adyacencia $A = A^T$ es una matriz real simétrica. Se asume que el grafo no contiene ciclos en el mismo nodo ($a_{ii} = 0$) ni múltiples enlaces entre 2 nodos [Mie11].

DEFINICIÓN 4 ***Conectividad.*** *La conectividad (o conectividad de un vértices) de un grafo conectado G (distinto a un grafo completo), , $\kappa(G)$, es el número más pequeño de vértices que pueden ser removidos para desconectar G [AW00].*

DEFINICIÓN 5 ***Vecindario.***

El conjunto de todos los vértices adyacentes al vértice v es llamado el vecindario de v y es denotado como $N(v)$. Cada par de vértices comunicado por un enlace directo es llamado un salto (one hop). Un $l-$vecindario es la unión de los vecindarios de vértices adyacentes desde un nodo fuente s a l saltos.

1.2 Estados globales

El cálculo local de un proceso p consiste de una secuencia $c_p^0, c_p^1, \ldots$ de estados de proceso donde c_p^0 es un estado inicial del proceso p. La transición de un estado $c_p^{(i-1)}$ a c_p^i es marcado por la ocurrencia de un evento e_p^i en p.

$$Ev = \cup_{p \in P} \{e_p^1, e_p^2, \ldots\}$$

Sobre eventos del proceso p se define un orden local causal como

$$e_p^i \prec_p e_p^j \Leftrightarrow i \leq j.$$

La comunicación de un proceso se ve reflejada en su estado, i.e, si el canal desde p a q existe entonces el estado $c_p^{(i)}$ es una lista de mensajes $sent_{pq}^i$ que p ha enviado a q.

El propósito de un algoritmo de instantáneas (snapshots) es construir explícitamente una configuración del sistema distribuido compuesto de estados locales de cada proceso. El proceso p toma una instantánea local guardando un estado local c_p^* llamado instantánea de p. Si la instantánea del estado es $c_p^{(i)}$, p toma la instantánea entre $e_p^{(i)}$ y $e_p^{(i+1)}$. Los eventos $e_p^{(j)}$ con $j \leq i$ son llamados "pre- instantáneas " de eventos de p. Los eventos con $j > i$ son llamados "post- instantáneas " del evento p.

Una instantánea global S^* consiste en una instantánea del estado c_p^* para cada proceso de $p \in P$, que se escribe:

$$S^* = \{c_{p1}^*, \ldots, c_{pN}^*\}.$$

DEFINICIÓN 6 ***Instantánea*** S^****.*** *Instantánea* S^* *es factible si para cada dos procesos vecinos* p *y* q, $rcvd_{pq}^* \subseteq sent_{pq}^*$.

DEFINICIÓN 7 ***Corte de*** Ev***.*** *Un corte* C *de* Ev *es un conjunto* $L \subseteq Ev$ *tal que* $e \in L \wedge e\prime \prec_p e \Rightarrow e\prime \in L$.

DEFINICIÓN 8 *La instantánea de* S^* *es significativo en el calculo de* C *si existe una ejecución* $E \in C$ *tal que* γ^* *es una configuración de* E.

El predicado del estado global es una función que mapea desde el conjunto de estados globales en el sistema $\wp$ a $\{True, False\}$. Una vez que el sistema entra a un estado en el cual es predicado es $True$, esto vuelve $True$ a todos los futuros estados alcanzables desde ese estado [Cou01]. El concepto de estado global será usado cambiando los procesos por nodos dentro del SD y SDM en *AEC*.

1.3 Consenso

Ante la proliferación de dispositivos y vehículos con recursos computacionales embebidos, se ha abierto un área de oportunidad a las aplicaciones cooperativas. Tales aplicaciones cooperativas necesitan desarrollar varias capacidades de control cooperativo. El control cooperativo a su vez enfrenta retos como los son el desarrollo de un sistema compuesto de subsistemas, el uso moderado de las comunicaciones entre vehículos, la evaluación de metas individuales y grupales.

Muchos de los estudios actuales señalan la utilización de esquemas centralizados como solución factible, sin embargo asumen el conocimiento global de la red, la capacidad de planificar las acciones de grupo de forma centralizada o la comunicación perfecta e ilimitada entre los vehículos participantes [eB08].

Un esquema de coordinación centralizada se basa en la suposición de que cada miembro del equipo tiene la capacidad de comunicarse a una ubicación central o compartir información a través de una red completamente conectada. Como resultado, el esquema centralizado no escala bien con el número de vehículos. El régimen centralizado puede resultar en una falla catastrófica del sistema global debido a su punto de falla. Además, las topologías de comunicación del mundo real suelen no estar completamente conectadas.

En muchos casos, dependen de las posiciones relativas de los vehículos y de otros factores ambientales y, por tanto, están cambiando de forma dinámica en el tiempo. Además, los canales de comunicación inalámbricos están sujetos a tener múltiples rutas, multi-trayecto, la atenuación de señal o la falla de algún elemento de la red. Por lo tanto, el control cooperativo, en presencia de restricciones de comunicación del mundo real, se convierte en un reto importante [eB08]. En base a los retos definidos, el concepto de consenso como una opción para solucionar este reto.

En las redes de agentes (o sistemas dinámicos), consenso significa llegar a un acuerdo con respecto a un determinado interés que depende del estado de todos los agentes. Un algoritmo de consenso (o protocolo) es una interacción o regla que especifica el intercambio de información entre un agente y todos sus vecinos en la red [OSFM07]. Formalmente se define como:

DEFINICIÓN 9 ***Consenso.***

Consenso (llamado también consenso regular) está especificado en términos de 2 funciones primitivas: proponer y decidir. Cada proceso (o nodo) tiene un valor inicial para el acuerdo, a través de la primitiva proponer. Los valores propuestos son privados de proceso y la acción de proponer es local. Esta acción dispara eventos de difusión en los cuales los procesos intercambian los valores propuestos para eventualmente alcanzar el acuerdo. Todos los procesos correctos deben decidir un valor a través de la primitiva decidir. Este valor será uno de los valores propuestos. El consenso debe satisfacer las siguientes propiedades:

Terminación *Cada proceso correcto eventualmente decide algún valor.*

Validación *Si un proceso decide v, entonces v fue propuesto por algún proceso.*

Integridad *Ningún proceso decide 2 veces en la misma ronda.*

Acuerdo *Dos procesos correctos no deciden un valor distinto.*

[GR06]

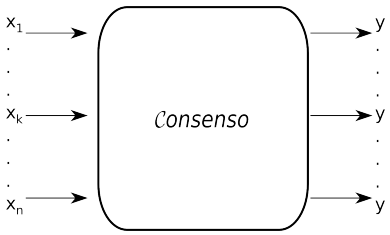


Figura 1.1: Representación del concepto de consenso.

Existen diferentes algoritmos de consenso, en particular en este trabajo se explican 3 tipos de consenso: consenso regular, consenso jerárquico y consenso aleatorio. Más adelante se hablará del consenso promedio, dado a que este tiene un

planteamiento distinto a los 3 algoritmos antes mencionados, distinto en el sentido que calcula un valor en base a las entradas y no lo elige un valor entre las entradas . Los ejemplos presentados a continuación han sido adecuados al problema de elegir un enlace con el costo más bajo entre un par de nodos.

1.3.1 Algoritmos de consenso

Definiendo la notación usada en los ejemplos posteriores, se considera que el grafo $G = (V, E)$ representa a la red (donde V es el conjunto de todos los nodos de la red y E es el conjuntos de aristas), sea A conjunto de nodos que participan en el consenso tal que $A \subset V | A = 1, \ldots, 5$. Cada nodo propondrá un enlace que parte de si mismo hacia alguno de sus vecinos a un salto, la notación $propose(v, y, c) | v \in A$, y es un vecino a un salto y c es el costo asociado a este enlace. Por simplicidad solo se expresará como $P(c)$ [GR06].

A continuación se presenta de manera resumida algunos de los algoritmos aplicados al caso de estudio, que es la búsqueda de rutas en una red, en los ejemplos solo se muestra como definir un paso de la ruta.

Consenso regular

También llamado consenso por inundación. La idea básica del algoritmo es el siguiente. Los procesos siguen rondas secuenciales. Cada proceso mantiene el conjunto de valores propuestos (propuestas) que ha visto, y este conjunto se ve aumentado cuando se pasa de una fase a otra (y nuevos valores propuestos son conocidos). En cada ronda, cada proceso difunde su propio conjunto a todos los procesos, es decir, el proceso inunda el sistema con todas las propuestas se ha recibido en rondas anteriores. Cuando un proceso recibe una propuesta presentada por otro proceso, este la une a su propio conjunto. En cada ronda, cada proceso calcula la unión de todos los conjuntos de valores propuestos recibidos hasta el momento.Un proceso decide un valor específico en su conjunto cuando se sabe que tiene reunido todas las propuestas que serán vistas por cualquier procedimiento correcto [GR06].

A continuación se explica cuando una ronda termina y un proceso de se mueve de una ronda a la siguiente(1), cuando un proceso sabe que es seguro decidir (2), y cómo un proceso de selecciona el valor para decidir(3)[GR06] .

1. Cada mensaje se etiqueta con el número de ronda en el que el mensaje fue transmitido. Cuando una ronda termina, en un proceso dado p_i, cuando p_i ha recibido un mensaje de todos los procesos que no son considerados sospechosos por p_i en esa ronda. Es decir, un proceso no deja una ronda a menos que reciba mensajes etiquetados con esa ronda, de todos los procesos de los que se sospecha que han terminado incorrectamente durante la ronda [GR06].

2. Se alcanza una decisión de consenso cuando un proceso sabe que tiene el mismo un conjunto de valores propuestos como todos los procesos correctos. En una ronda donde un nuevo fallo es detectado, un proceso p_i no está seguro de tener exactamente el mismo conjunto de los valores como los otros procesos. Para saber cuando es seguro decidir, cada proceso mantiene un registro de los procesos de los cuales no sospechaba en la ronda anterior, y de cuántos procesos que ha recibido una propuesta en la ronda actual. Si una ronda termina con el mismo número de procesos de los que no se sospecha en la ronda anterior, la decisión puede ser tomada. En cierto sentido, los mensajes transmitidos por todos los procesos que se trasladaron a la actual ronda alcanzan su destino [GR06].

3. Para tomar una decisión, un proceso puede entonces aplicar cualquier función determinista para el conjunto de valores acumulados, siempre que esta función es la misma en todos los procesos (es decir, haya sido aprobada por todos los procesos con anticipación). En este caso, el proceso determina el valor mínimo: que implícitamente se asume que el conjunto de todas las propuestas posibles está totalmente ordenado y el orden es conocido por todos los procesos. (Los procesos también podrían escoger el valor propuesto por el proceso con la identidad más bajo, por ejemplo.) Un proceso que decide, simplemente difunde la decisión a todos los procesos usando *best-effort broadcast* [GR06].

Una ejecución del algoritmo se ilustra en la Figura 1.2. Proceso de p_1 bloquea durante la primera ronda después de la difusión de su propuesta. Sólo p_2 ve esa propuesta. Ningún otro proceso se bloquea. Por lo tanto, p_2 ve las propuestas de todos los procesos y puede decidir. Esto es porque el conjunto de procesos de la

que recibe propuestas en la primera ronda es el mismo que el conjunto inicial de procesos que inician el algoritmo. Proceso de p_2 toma el mínimo de las propuestas y decide el valor 3. Procesos p_3 y p_4 detectar la caída de p_1 y no puede decidir. Así que avanzan a la siguiente ronda, es decir, a la ronda 2. Tenga en cuenta que si cualquiera de estos procesos decidió el mínimo de las propuestas que tenía después de la ronda 1, habrían decidido de otra manera, es decir, el valor 5. Desde p_2 ha decidido, p_2 difunde su decisión a través de una emisión de mejor esfuerzo. Cuando se dictó la resolución, procesa p_3 y p_4 también decide 3 [GR06] .

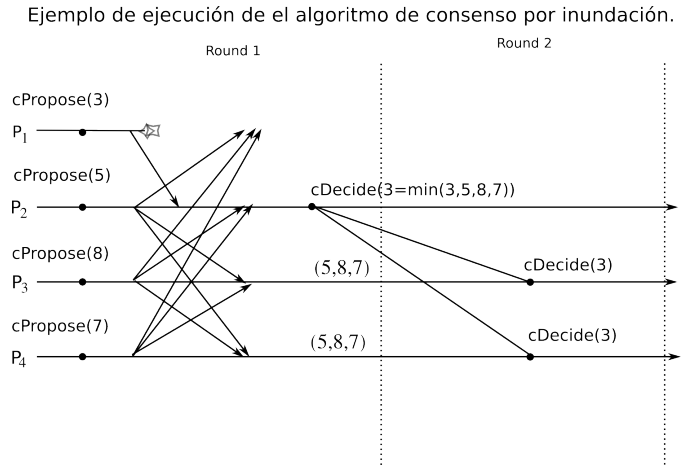


Figura 1.2: Ejemplo de consenso regular o por inundación.

Si no hay fallas, el algoritmo requiere un único paso de comunicación: todos los procesos deciden al final de la ronda 1, si se sospecha que hay un proceso que se bloqueó. Cada fallo puede causar a lo sumo un paso de comunicación adicional. Por lo tanto, en el peor de los casos el algoritmo requiere N pasos , si $N - 1$ procesos se bloquean en secuencia. Si no hay fallas, el algoritmo intercambia $2N^2$ mensajes. Hay N^2 intercambios de mensajes adicionales para cada ronda, donde un proceso se bloquea [GR06].

Consenso jerárquico

En el consenso jerárquico los nodos son ordenados por prioridad. El nodo con mayor jerarquía impondrá su valor a todos conforme pasan las rondas, ningún nodo puede difundir si su superior no lo ha hecho antes, si un nodo se desconecta antes

de imponer su propuesta, deja de difundirse su valor y se difundirá el valor del nodo que le sigue en jerarquía, si este nodo no falla su valor se impondrá al de todos los nodos [GR06].

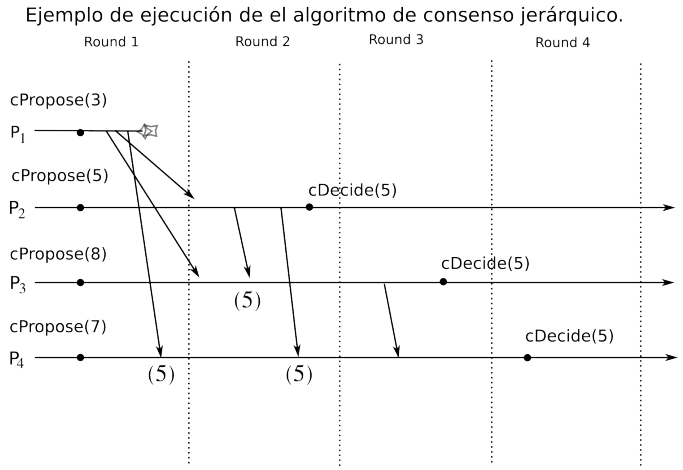


Figura 1.3: Ejemplo de consenso jerárquico.

Como se ve en la Figura 1.3, Proceso p_1 decide 3 y difunde su propuesta a todos los procesos, pero se bloquea. Procesosp_2 y p_3 detectar el fallo antes que entregar la propuesta de p_1 y avanzar a la siguiente ronda.el proceso p_4 entrega el valor de p_1 y cambia su propia propuesta en consecuencia, es decir, $p : 4$ adopta el valor de p_1. En la ronda 2, el proceso p_2 decide y emite su propia propuesta. Esto hace que p_4 para cambiar nuevamente su propuesta, es decir, ahora adopta el valor de p_2. A partir de este momento, no hay más fracasos y los procesos deciden en secuencia el mismo valor, es decir, valor de p_2 que es 5 [GR06].

El algoritmo intercambia $N - 1$ mensajes por ronda y puede ser reducido a $(N-1)/2$ si se optimiza a que los nodos no envíen mensajes a los nodos con mayor categoría que ellos mismos.El algoritmo también requiere pasos de comunicación N para terminar [GR06].

Consenso aleatorio.

Esta variante de consenso asegura la integridad del acuerdo. Cada nodo tiene un valor inicial que propone a través de la primitiva $P(c)$. El algoritmo consta de n rondas, cada ronda está compuesta por 2 fases. En la primera cada nodo propone un

valor, si al recibir las propuestas de los nodos con los que se comunica se detecta un valor predominante, se decide este valor y se propone para la siguiente fase; si este nodo no encuentra un valor mayoritario entonces escoge un valor de manera aleatoria entre los valores iniciales y lo propone para la siguiente ronda. El algoritmo continua hasta que todos los nodos proponen el mismo valor, como se ilustra en la Figura 1.4 [GR06].

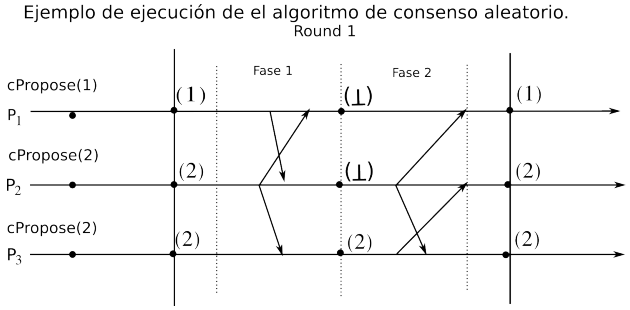


Figura 1.4: Ejemplo de consenso aleatorio.

Consenso promedio

Esta variante del consenso regular asegura la convergencia del acuerdo. Cada nodo propone un valor inicial a través de la primitiva $P(c)$. El algoritmo consta de 2 rondas, en que se propone un valor inicial, después de la propagación de valores se decide el promedio de los valores recibidos. Este se propaga hasta que todos los nodos participantes deciden el mismo valor, sin embargo en esta versión de consenso el valor del consenso no estará en el conjunto de lo valores propuestos inicialmente, es decir, es calculado.

Los conceptos de consenso revisados en esta sección se incorporarán a los de los algoritmos de encaminamiento en redes inalámbricas y SDM. A continuación se introducen conceptos sobre la representación de redes que serán utilizados en el diseño de algoritmo propuesto en este trabajo.

1.4 Representación de redes

Existen múltiples maneras de representar la conectividad de una red, entre éstas una red puede ser vista como un grafo, a su vez, un grafo puede representarse con listas

de aristas, listas de adyacencia. Los grafos pueden ser representados como matrices. Existen 2 tipos de matrices que son las más usadas para representar grafos. Una es la basada en la adyacencia de los vértices y la otra basada en la incidencia de los vértices y aristas [Ros00].

Para el caso de una topología fija, esta puede ser representada como un grafo y este a sus vez por una matriz de adyacencia.

DEFINICIÓN 10 ***Matriz de adyacencia***

Suponemos que $G = (V, E)$ es un grafo simple, donde $|V| = n$. Si suponemos que los vértices de G son listados arbitrariamente como $v_1, v_2, \ldots, v_n$. La matriz de adyacencia A (o A_G) de G, con respecto a la lista de vértices, es la $n \times n$ matriz binaria con 1 en su entrada (i, j) cuando v_i y v_j son adyacentes y 0 si no son adyacentes. En otras palabras, si su matriz de adyacencia es $A = [a_{i,j}]$, entonces:

$$a_{i,j} = \begin{cases} 1 & \text{si } v_i, v_j \text{ es una arista de } G \\ 0 & \text{de otra manera} \end{cases} \tag{1.1}$$

Nótese que una matriz de adyacencia está basada en el orden de los vértices. Por lo tanto existen $n!$ diferentes matrices de adyacencia para el grafo con n vértices, dado que hay $n!$ formas posibles de ordenar n vértices [Ros00].

La matriz de adyacencia de un grafo simple es simétrica, lo que significa que $a_{i,j} = a_{j,i}$, puesto que ambas entradas son iguales. Además el grafo simple no tiene ciclos, entonces cada entrada $a_{i,i}, i = 1, 2, \ldots, n = 0$. Cuando hay pocas aristas en el grafo, la matriz de adyacencia es una matriz dispersa, ya que tiene pocas entradas distintas a 0 [Ros00].

Otra manera de representar grafos es usando matrices de incidencia [Ros00].

DEFINICIÓN 11 ***Matriz de incidencia.*** *Sea $G = (V, E)$ un grafo no dirigido. Si suponemos que $v_1, v_2, \ldots, v_n$ son vértices y $e_1, e_2, \ldots, e_m$ son aristas de G. Entonces la matriz de incidencia con respecto al orden de V y E es la matriz $n \times m, M = [m_{i,j}]$, donde*

$$m_{i,j} = \begin{cases} 1 & \text{cuando la arista } e_j \text{ es incidente con } v_i \\ 0 & \text{de otra manera} \end{cases}$$

[Ros00].

Estos conceptos son necesarios para entender como funcionan y como se representan las redes móviles, así como algunos los algoritmos de encaminamiento usados actualmente, todo esto para considerar los aspectos y características exitosas de éstos así como las deficiencias.

1.5 Algoritmos de rutas más cortas entre pares de nodos

En la literatura existen una gran variedad de algoritmos para encontrar la ruta más corta [MR07, CSRL01], como los algoritmos de Bellman-Ford y variaciones del algoritmo de Dijkstra 1.5.2. A continuación se explican ambos algoritmos de manera detallada.

1.5.1 Algoritmo de Bellman-Ford

Este algoritmo usa la simple idea de calcular las rutas más cortas entre 2 nodos en una forma centralizada. En un ambiente distribuido, un enfoque de vector de distancia es tomado para calcular las rutas más cortas [MR07, CSRL01].

Son usados 2 nodos genéricos para explicar el algoritmo, etiquetados como i y j, en una red de N nodos, algunos estarán directamente conectados, como los nodos 4 y 6 de la Figura 1.5. Observando la Figura 1.5, no todos los nodos están

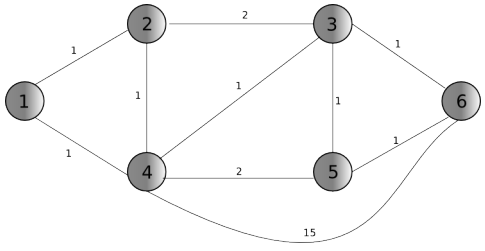


Figura 1.5: Red de 6 nodos.

conectados directamente, en estos casos para encontrar la distancia entre 2 nodos es necesario pasar por otros nodos y enlaces [MR07].

Definamos la siguiente notación para el costo de los nodos:

$d_{i,j}$ =Costo del enlace entre los nodos i y j.

$\overline{D_{i,j}}$ = Costo del cálculo de la ruta mínima desde el nodo i al nodo j.

Si 2 nodos están directamente conectados, entonces, el costo del enlace $d_{i,j}$ toma un valor finito.

Considerando de nuevo la Figura 1.5, los nodos 4 y 6 están directamente conectados con un costo de 15, podemos escribir $d_{4,6} = 15$, sin embargo, los nodos 1 y 6 no están directamente conectados, entonces $d_{1,6} = \infty$ [MR07].

La diferencia entre $d_{i,j}$ y $\overline{D_{i,j}}$, es que los nodos 4 y 6, el costo mínimo actual es de 2, el cual tiene la ruta $4-3-6$, es decir, $\overline{D_{4,6}} = 2$, mientras que $d_{4,6} = 15$, para los nodos 1 y 6, encontramos que $\overline{D_{1,6}} = 3$, mientras que $d_{1,6} = \infty$ [MR07].

Por lo tanto, el costo mínimo de la ruta puede ser obtenido entre 2 nodos en una red, sin importar si están o no directamente conectados, al menos que algún nodo esté totalmente aislado de la red. Ahora lo importante es como calcular el costo mínimo entre 2 nodos de la red [PD07].

Consideremos un nodo genérico k en la red, el cual está directamente conectado a cualquier nodo final, asumimos que k está directamente conectado al nodo j, es decir $d_{k,j}$ tiene un valor finito [MR07].

Las siguientes ecuaciones, son conocidas como las ecuaciones de Bellman, que deben ser satisfechas por la ruta más corta del nodo i al nodo j:

$$\overline{D_{i,j}} = 0, \forall i. \tag{1.2}$$

$$\overline{D_{i,j}} = min_{k \neq j}\{\overline{D_{i,j}} + d_{k,j}\} \text{para } i \neq j. \tag{1.3}$$

Una variación del algoritmo se usa cuando el costo mínimo se obtiene por iteraciones del número de saltos. Específicamente definimos el término para el costo mínimo en términos del número de saltos h, como sigue:

$\overline{D_{i,j}}^{h}$ = Costo de la rutas de mínimo costo desde el nodo i al nodo j cuando más de h número de saltos es considerado [MR07].

Este algoritmo que itera en términos del número de saltos se define como :

DEFINICIÓN 12 ***Algoritmo de Bellman-Ford.***
Inicializar todos los nodos i y j en la red:

$$\overline{D_{i,i}}^{(0)} = 0, \forall i;\ \overline{D_{i,j}}^{(0)} = \infty, \textit{para } i \neq j. \tag{1.4}$$

Para $h = 0$ *hasta* $N - 1$*, hacer:*

$$\overline{D_{i,i}}^{(h+1)} = 0, \forall i. \tag{1.5}$$

$$\overline{D_{i,j}}^{(h+1)} = min_{k \neq j}\{\overline{D_{i,k}}^{(h)} + d_{k,j}\}, \textit{para } i \neq j. \tag{1.6}$$

1.5.2 Algoritmo de Dijkstra

Este algoritmo calcula la ruta más corta para todos los destinos desde una fuente, en vez de solo para un par especifico de nodos en un tiempo. Consideremos un nodo genérico i en una red de N nodos desde donde queremos calcular las rutas más cortas a todos los nodos en la red. La lista de N nodos se denotará por $N = \{1, 2, \ldots, N\}$. Un destino genérico será denotado como j ($j \neq i$). Usaremos los siguientes términos:

$d_{i,j}$ =Costo del enlace entre los nodos i y j.

$\underline{D}_{i,j}$ = Costo del cálculo de la ruta mínima desde el nodo i al nodo j.

El algoritmo divide la lista de N nodos en 2 listas: comienza con una lista permanente S, la cual representa los nodos ya considerados y una lista tentativa S' de nodos que aún no han sido considerados [MR07].

Mientras el algoritmo progresa, la lista S se expande con nuevos nodos incluidos, mientras que la lista S' disminuye cuando un nuevo nodo es incluido en S, éste es borrado de S', el algoritmo para cuando la lista S' está vacía. Inicialmente, tenemos $S = \{i\}$ y $S = N \backslash \{i\}$ [MR07, CSRL01].

El algoritmo tiene 2 partes:

- Como expandir las listas.
- Como calcular la ruta mínima a los nodos que son vecinos de los nodos de la lista S.

La lista S se expande en cada iteración, considerando un nodo vecino k del nodo i con la ruta menos costosa desde el nodo i. En cada iteración, el algoritmo considera la vecindad de nodos k, la cual no está aún en S, para ver si el costo mínimo cambia desde la última iteración [MR07].

Usando el ejemplo de la Figura 1.5. Suponemos que el nodo 1 quiere encontrar las rutas más cortas hacia todos los nodos de la red. Inicialmente $S = \{1\}$ y $S' = \{2, 3, 4, 5, 6\}$ y las rutas más cortas hacia todos los nodos que son vecinos directos del nodo que pueden ser fácilmente encontrados, mientras que el costo de los nodos restantes es ∞, i.e.

$$\underline{D}_{1,2} = 1, \underline{D}_{1,4} = 1, \underline{D}_{1,3} = \underline{D}_{1,5} = \underline{D}_{1,6} = \infty. \tag{1.7}$$

Para la siguiente iteración, notamos que el nodo 1 tiene 2 vecinos directos: el nodo 2 y el nodo 4 con $d_{1,2} = 1$ y $d_{1,4} = 1$. respectivamente, todos los otros nodos no están directamente conectados al nodo 1, entonces el costo directo sigue siendo ∞. Como los nodos 2 y 4 son vecinos con el mismo costo mínimo, podemos escoger cualquiera de los 2. Supongamos que escogemos el nodo 2, entonces tenemos $S = \{1, 2\}$ y $S' = \{3, 4, 5, 6\}$. Ahora preguntamos al nodo 2 el costo a sus vecinos directos que ya están en el conjunto S. En la Figura 1.5 se nota que los vecinos del nodo 2 son el nodo 3 y 4. Entonces, comparamos y calculamos el costo desde el nodo 1 para estos 2 nodos y vemos si hay alguna mejora:

$$\underline{D}_{1,3} = min\{\underline{D}_{1,3}, \underline{D}_{1,2} + d_{2,3}\} = min\{\infty, 1 + 2\} = 3. \tag{1.8}$$

$$\underline{D}_{1,4} = min\{\underline{D}_{1,4}, \underline{D}_{1,2} + d_{2,4}\} = min\{1, 1 + 1\} = 1. \tag{1.9}$$

No hay ninguna mejora en el costo del nodo 4, entonces mantenemos la ruta más corta original. Para el nodo 3, ahora tenemos la ruta mínima : 1-2-3. Para el resto de los nodos, el costo sigue siendo ∞. Esto completa esta iteración. Seguimos con la siguiente iteración y encontramos que el siguiente intermediario es $k = 4$ y el proceso continua como se explicó anteriormente. La generalización de este algoritmo se puede apreciar en la definición 2 [MR07].

DEFINICIÓN 13 ***Algoritmo de Dijkstra para la ruta más corta.***

1. *Comenzar con un nodo fuente i en la lista de nodos permanentes ya considerados, i.e., $S = \{i\}$; el resto de los nodos se ponen en la lista de los nodos tentativos, etiquetada como S'. Iniciar*

$$\underline{D}_{i,j} = d_{i,j}, \forall j \in S'. \tag{1.10}$$

2. *Identificar un nodo vecino (intermediario) k (que no esté en la lista S) con el costo mínimo de la ruta desde el nodo i, i.e., encontrar $k \in S'$ tal que*

$$\underline{D}_{j,k} = min_{min \in S'} \underline{D}_{i,m}. \tag{1.11}$$

   Agregar k a la lista de nodos permanentes S, i.e., $S = S \cup \{k\}$,
   Quitar a k de la lista de nodos tentativos S', i.e.,$S = S \backslash \{k\}$,
   Si S' está vacía, parar.

3. *Considerar la lista de nodos vecinos, N_k, de los nodos intermediarios k (pero no considerar los que están en S) para revisar por una mejora en el costo de la ruta más corta, i.e., para $j \in N_k \cap S'$.*

$$\underline{D}_{i,j} = min\{\underline{D}_{i,j}, \underline{D}_{i,k} + d_{k,j}\}. \tag{1.12}$$

[MR07], [CSRL01]

Es necesario conocer a fondo el funcionamiento de los algoritmos de encaminamiento en todas sus modalidades, ya que esto permite conocer las deficiencias. Entre las debilidades de éstos encontramos que no se considera en ningún momento el estado del planificador del nodo que funcionará como enrutador, es importante tomarlo en cuenta puesto que si el planificador no asigna ni espacio ni memoria para transmitir el mensaje, el nodo no puede cumplir la función de enrutador.

Un protocolo de encaminamiento debe adaptarse a los cambios de la red. Un enrutador que deje de funcionar intempestivamente puede afectar la capacidad de enviar paquetes a su destino. Si los cambios de la red demandan que todas la tablas de encaminamiento sean actualizadas, el retraso hasta que este proceso concluya (se conoce como **convergencia**) es muy significativo. Otro problema de interés, es la posibilidad de un ciclo en el encaminamiento (**routing-loop**) causado por una convergencia lenta. Un ciclo en el encaminamiento ocurre cuando los paquetes permanecen en un segmento de la red (describiendo un ciclo) sin alcanzar su destino nunca [Mac05].

Un protocolo de ruta simple (**single path**) debe escoger la mejor ruta de las rutas disponibles como la ruta primaria (basándose en las métricas para cada ruta), solo si la ruta primaria falla se puede usar una segunda ruta. Los protocolos

de encaminamiento difieren en como deben enviarse los paquetes a otros routers. Algunos protocolos como RIP e IGRP usan paquetes de difusión (*Broadcast*), en la cual son recibidos por cada dispositivo en la red. Los protocolos más sofisticados utilizan paquetes **Unicast** o **Multicast** (EIGRP, OSPF y BGP) los cuales son recibidos por un dispositivo o un grupo de dispositivos en particular [Mac05].

Como se detalló en el planteamiento de problema (en la sección del capítulo) , se propone utilizar un algoritmo de consenso para la búsqueda de rutas en un SDM, con el propósito de obtener una decisión local. A continuación se presentan conceptos básicos y se introduce un algoritmo de consenso en particular. En [LSBB01] se reporta un algoritmo de consenso basado en votantes promedio ponderados (que se presenta en la siguiente sección a detalle), este algoritmo fue adaptado para usarlo en el *algoritmo de encaminamiento por consenso* que se presenta en este trabajo, de esta forma se obtiene un acuerdo local que será el siguiente paso en la ruta y este a su vez comenzará un nuevo consenso y así en forma sucesiva hasta alcanzar el nodo objetivo.

1.6 Redes de comunicación

Una red de comunicación está hecha de nodos y enlaces. Dependiendo del tipo de red, los nodos tienen distintos nombres. En una red de comunicación el tráfico fluye desde el nodo origen hacia el nodo destino.Tres factores son importantes para determinar el desempeño en una red de comunicación, estos factores son la topología, el método de conmutación (*switching*) y el algoritmo de encaminamiento. La topología se define como los nodos conectados en la red. La topologías de malla e hipercubo son las más populares empleadas en la topología de computadoras paralelas tal como el Cubo cósmico de calTech [Sei85] . El hipercubo es simétrico y regular.

El método de conmutación (*switching*) determina la manera que los mensajes visitan los enrutadores intermedios a lo largo de la ruta a su destino. Una gran cantidad de dispositivos utilizan el esquema de *switching* llamado *wormhole*, debido a su bajos requisitos de memoria y más importante, porque ha hecho la latencia casi independiente de la distancia entre la fuente y los nodos destino. En el swichting

wormhole, el mensaje es divido en una secuencia de unidades de tamaño fijo, llamadas *flits*. La cabecera de cada *flit* establece la ruta a seguir a través de la red, mientras los datos faltantes siguen una tendencia de línea de ensamble [Sey98].

La comunicación en una red depende de la latencia [1] y ésta a su vez está condicionada por la técnica de conmutación (*switching*) [Sey98]. Existen 2 enfoques para el encaminamiento de paquetes, basados en las acciones que toma el dispositivo de conmutación (*switch*) cuando comienzan a llegar los *flits*.

1. Almacenar y reenviar.

2. Atravesar.

    - Atravesar Virtualmente.
    - *Wormhole.*

Por ejemplo el algoritmo *wormhole* , para la comprensión de este algoritmo es necesario introducir los conceptos de *phit* y *flit*. Se entiende como *phit* a la unidad más pequeña de información en la capa física, la cual se transfiere a través de un enlace físico en un ciclo; en cambio, un *flit* es una pequeña unidad de información en la capa de enlace del tamaño de una o varias palabras. El algoritmo *wormhole* funciona de la siguiente manera: Un paquete es dividido en n *flits*. El *flit* que encabeza el paquete gobernará la ruta, los *flits* restantes seguirán en *pipeline* a la cabecera (Figura 1.6) [DYN97].

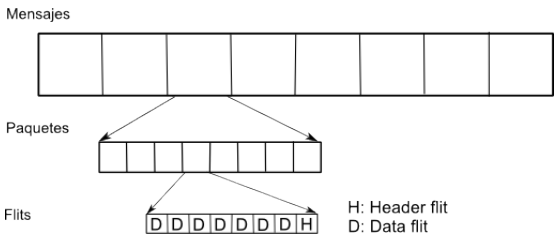


Figura 1.6: Esquema de fragmentación de paquetes para ser enviados por la red [Moh98].

[1] Se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red y el tiempo de reenvío, almacenamiento de los nodos intermedios.

Por lo tanto el mensaje se asemeja a un gusano a través de la red, finalmente se reensambla en el destino. Los paquetes son trasmitidos desde un nodo como *flits*, se identifican 2 tipos de *flits*: *Flits* cabecera y *flits* de datos. El *flit* cabecera trata de obtener un canal de comunicación mientras que los *flits* de datos son transmitidos por el canal que ya se ha obtenido. Un canal se libera sólo cuando el último flit del mensaje ha pasado a través de él. Los flits de 2 mensajes distintos no pueden ser intercalados. Si la cabecera no puede continuar avanzando por la red, debido a que los canales de salida están ocupados, entonces la ruta queda bloqueada por los flits estancados, esto bloquea otras posibles comunicaciones. Este algoritmo es muy susceptible a presentar bloqueos mutuos. En caso de no presentarse, la latencia de la comunicación es:

$$T_{WH} = d(t_r + t_w + t_m) + max(t_w, t_m)M. \tag{1.13}$$

Donde:

$d =$ Longitud de la ruta.

$\mu =$ la tasa máxima en $bits/segundo$ en el que los bits pueden ser transferidos a través de cada cable de un canal físico.

$B = \mu(phits/segundo) =$ ancho de banda $= w\mu(bits/segundo)$.

$w =$ tamaño del $flit =$ ancho de banda en bits.

$M =$ Tamaño del paquete en $flits$.

$t_r =$ Tiempo de decisión de encaminamiento en un solo router.$(segundos)$

$t_w =$ latencia interna de conmutación en el router, una vez que se ha tomado la decisión de encaminamiento $(segundos/phit)$.

$t_m = 1/B =$ latencia entre el canal y el router$(segundos/flits)$.

como se ve en la Figura 1.7 [DYN97].

Eventualmente los paquetes son encaminados desde una fuente hacia un destino. Tales paquetes pueden necesitar atravesar muchos puntos de cruce, similares

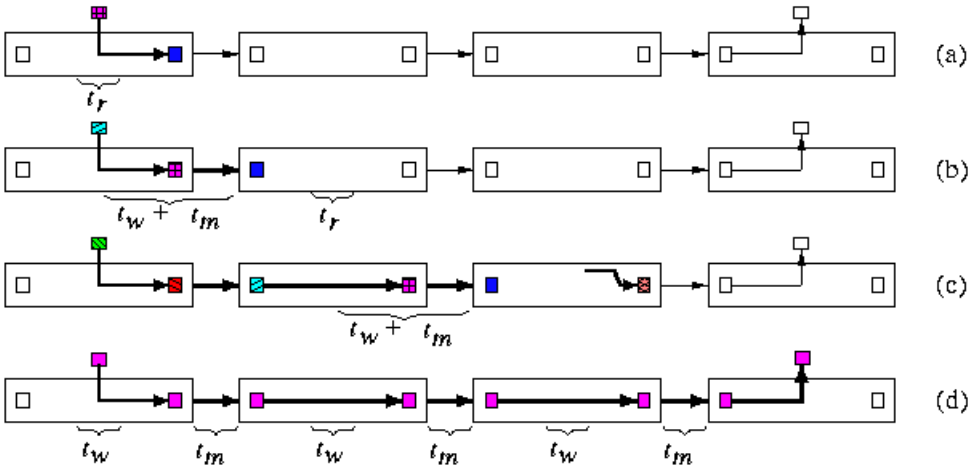


Figura 1.7: Latencia de la comunicación **(a)**La cabecera es copiada en el *buffer* de salida después de tomar una decisión sobre la ruta a seguir. **(b)** El *flit* cabecera es trasferido a un segundo enrutador y otro *flit* lo sigue. **(c)** El *flit* cabecera llega a un enrutador con el canal de salida ocupado y la cadena completa de *flits* se detiene, bloqueando ese canal. **(d)** La cadena de *flits* en caso de no tener conflicto de canales disponibles, establece un *wormhole* (hoyo de gusano) a través de los enrutadores [DYN97].

a las intersecciones de calles en una red de transporte. Estos puntos de cruce en Internet se llaman enrutadores. Las funciones de los enrutadores son leer la dirección de destino, marcar el paquete IP como paquete entrante para consultar su información e identificar el enlace al cual el paquete será reenviado y reenvía el paquete [MR07].

Haciendo una analogía entre la red de comunicación y una de transporte, el número de carriles y el límite de velocidad es similar al enlace de red que conecta 2 enrutadores, éste es limitado por cuanta información puede ser transmitida por unidad de tiempo, llamada ancho de banda o capacidad de enlace; esto es representado por una tasa de datos (por ejemplo 1.54 mega bits por segundo). Una red entonces carga tráfico en su enlace y a través de sus enrutadores hacia su eventual destino; el tráfico en la red se refiere a los paquetes generados por diferentes aplicaciones, tales como el Internet o el correo electrónico[MR07].

Si suponemos que el tráfico aumenta de repente, entonces los paquetes generados pueden ser agregados a la cola de un enrutador o pueden ser descartados,

puesto que el enrutador tiene una cola de capacidad finita, conocida como *buffer*, para temporalmente guardar paquetes acumulados. Como el protocolo TCP/IP permite la posibilidad de que un paquete IP no sea entregado o sea descartado durante el camino, el *buffer* finito del enrutador no es problema. Desde el punto de vista de la eficiencia, es deseable no tener perdida de paquetes (o que sea mínima) durante el tránsito [MR07].

1.7 Encaminamiento

Una característica muy importante de una red de comunicación es el camino desde un nodo fuente al nodo destino. La ruta puede ser establecida manualmente, por lo que esa ruta seria una *ruta estática*. En general, es deseable usar un algoritmo de encaminamiento para determinar la ruta [MR07].

El objetivo del algoritmo de encaminamiento es, en general, dictado por la exigencia de la red de comunicación y el proveedor de servicios quiere imponer sobre sí mismo. Mientras que las metas pueden ser distintas para diferentes tipos de redes de comunicación, éstas pueden ser clasificadas en 2 categorías generales:

- **Orientadas al usuario**. Que una red esté orientada al usuario, significa que la red provee un buen servicio a cada usuario, tal que el tráfico puede moverse desde la fuente al destino rápidamente para ese usuario. Esto no debe ser para un usuario específico a expensas de otros usuarios entre la fuente o el destino de otros nodos de la red [MR07].

- **Orientadas a la conexión**.Generalmente es dar respuesta de cómo proporcionar un encaminamiento eficiente y justo para que la mayoría de los usuarios reciban un servicio bueno y aceptable, en lugar de proporcionar la ”mejor”servicio a un usuario específico. Este punto de vista es en parte necesario porque hay una cantidad finita de los recursos en una red, por ejemplo, la capacidad de la red [MR07].

La IP asume que la computadora está directamente conectada a una red local (por ejemplo, una LAN Ethernet) y que puede enviar paquetes directamente a cualquier otra computadora sobre esa misma red; si la dirección de destino es en la

red local, IP simplemente accede al medio físico de transmisión y envía el paquete [MR07].

El problema aparece cuando la dirección de destino queda en otra red, en cuyo caso TCP/IP recurrirá a un *gateway* para enviar indirectamente los paquetes. Se denomina *gateway* a un dispositivo (ya sea otra computadora o un dispositivo específicamente diseñado a tal efecto) que está conectado a más de una red y tiene la capacidad para redirigir (forward) paquetes entre esas redes [MR07].

Para determinar a qué *gateway* deberán enviarse los paquetes, TCP/IP extrae la dirección de red del nodo de destino y consulta una tabla, denominada tabla de encaminamiento, en la cual se listan las redes conocidas y los *gateways* que pueden utilizarse para alcanzarlas [MR07].

Red	Gateway
170.25.1.0	170.25.3.254
170.25.2.0	170.25.3.254
170.25.3.0	*
170.25.4.0	170.25.3.253

Tabla 1.1: Ejemplo de tabla de encaminamiento.

En la Tabla 1.1, el asterisco indica que no es necesario ningún *gateway* para llegar a la red en cuestión dado que la máquina tiene conexión directa a la misma. El encaminamiento hacia redes remotas se realiza en base a direcciones de red ; además, la tabla de encaminamiento especifica tanto la red local como las remotas, y que para el caso de éstas últimas, indica la dirección IP de alguna máquina en la red local que puede ser utilizada para alcanzarla [Sib02].

La ruta que seguirán los paquetes desde el origen hasta su destino se va decidiendo a medida que los mismos viajan por la red. Cada nodo es responsable de determinar cual es el próximo *salto* en dirección al nodo final en función del contenido de su tabla de encaminamiento. Este modelo de encaminamiento asume que si el nodo de destino no pertenece a la red local, deberá haber una entrada en la tabla de encaminamiento que especifique el *gateway* a utilizar. En otras palabras, asume que todos los nodos están al tanto de la estructura de la red. En consecuencia, cada vez que la estructura de la red cambia (por ejemplo, cuando se agrega o elimina

una subred), el administrador debería actualizar las tablas de encaminamiento en todos los nodos. Igualmente, si la red se interconectara a otra red, una nueva entrada deberá agregarse en las tablas de encaminamiento de cada máquina [Sib02].

Siguiendo con este razonamiento, a medida que la red crece y se interconecta a otras redes las tablas de encaminamiento se hacen mas largas y complejas; inclusive sería posible que fueran virtualmente imposibles de construir o mantener, en especial si la red se conecta a Internet (formada por miles de redes independientes)[Sib02].

Por supuesto, existen previsiones para enfrentar estos problemas: el encaminamiento dinámico o adaptativo y las rutas por defecto [Sib02].

1.7.1 Tipos de encaminamiento

El algoritmo de encaminamiento indica el camino que un mensaje debe tomar para alcanzar sus destino, seleccionando el canal de salida adecuado. Este canal puede seleccionar de un conjunto de posibles opciones de acuerdo al tamaño del conjunto, los algoritmos de encaminamiento son clasificados en 3 categorías:

1. Encaminamiento determinista. Asignan un camino simple entre cada par de fuente y destino. Esta forma de encaminamiento es popular debido a que se evitan situaciones de bloqueo mutuo, resultando una aplicación simple. Sin embargo, en un mensaje que se trasmite por este método no se pueden utilizar rutas alternativas para evitar la congestión de canales a lo largo del camino hacia el destino, por lo que el rendimiento de la red es bajo [PSA06].

2. Algoritmos de encaminamiento completamente adaptativo. En este tipo de algoritmos le permite al mensaje explorar los caminos disponibles, es decir el conjunto de opciones llega a su máximo en estos algoritmos, por lo tanto es posible hacer un mejor uso de los recursos de la red pero implica mayor complejidad para el bloqueo mutuo [PSA06].

3. Algoritmos de encaminamiento parcialmente adaptativo. Estos algoritmos tratan de combinar las ventajas de las otras 2 categorías para producir un encaminamiento con limitada adaptabilidad y establecer un equilibrio entre el rendimiento y la complejidad del enrutador. Estos algoritmos permiten seleccionar una ruta de un subconjunto de todas las posibles rutas, es decir,

limitan el tamaño del conjunto de las posibles opciones. La mayoría de estos algoritmos están basados en algoritmos planares adaptativos, sobre todo han sido propuestos para redes de malla y de hipercubo [PSA06].

1.7.2 Ciclos de encaminamiento.

Las inconsistencias en el estado del encaminamiento son causadas por ciclos de encaminamiento en el Internet. Un ciclo de encaminamiento causa que paquetes de queden atrapados en un ciclo, lo que ocasiona que sean descartados o retrasados, dependiendo de cuanto tiempo el paquete permanezca en el ciclo. Los protocolos de encaminamiento de Internet agregan direcciones en los prefijos, por lo que un paquete puede verse dirigido a un ciclo de encaminamiento por causa de un un prefijo. Y que muchos paquetes pueden ser capturados en un ciclo, lo que impacta de manera sustancial a la cantidad de trafico; los ciclos de encaminamiento pueden causar que los paquetes se queden atrapados en un ciclo atravesando el mismo enlace en la red varias veces y mostrarse como réplicas en la traza del paquete original. Hay dos formas de ciclos de encaminamiento que pueden ocurrir: transitorio y persistente [HMMD02].

Los ciclos transitorios ocurren como parte de operación normal del protocolo de encaminamiento a causa de los diferentes retardos en la propagación de información a diferentes partes de la red. Los ciclos transitorios pueden ser resueltos sin la intervención humana, como la convergencia de protocolos de encaminamiento. Los ciclos persistentes se incrementan por varias razones, sobre todo por desconfiguraciones del router para eliminar los ciclos persistentes se requiere de la intervención humana. Los ciclos persistentes son difíciles de analizar por dos razones, primero son raros, segundo ellos ocurren a lo largo de múltiples sistemas autónomos y se requiere la comparación de muchos grupos de operación de redes para ser analizados [HMMD02].

Los protocolos de encaminamiento distribuyen la información a través de la red de manera que todos los routers en la red eventualmente convergerán a una vista consistente de la red. Consecuentemente, los ciclos de encaminamiento ocurren como resultado de una inconsistencia temporal que se incrementa durante la con-

vergencia del proceso, o como resultado de inconsistencia permanente causada por la desconfiguración u oscilación de rutas [HMMD02].

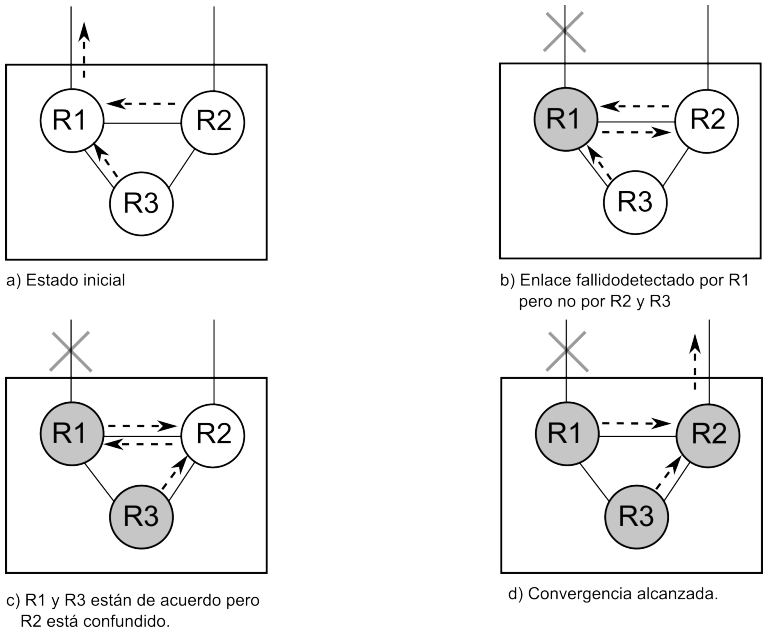


Figura 1.8: Escenario para ciclos transitorios [HMMD02] .

Un ejemplo simple de la evolución de los ciclos transitorios se muestra en la Figura 1.8, se considera una red pequeña con tres nodos $R1, R2$ y $R3$ estos nodos están conectados con lineas sólidas representando enlaces físicos, y líneas punteadas representando el flujo del tráfico entre nodos. Los nodos con estados consistentes de encaminamiento son mostrados con el mismo sombreado. El tráfico desde los tres nodos hacia otros nodos inicialmente viaja desde $R1$ como se muestra en a); $R2$ también se muestra como ruta alternativa a otras redes [HMMD02].

El enlace que conecta $R1$ a otras redes falla y entonces $R1$ es el primero en detectar la falla. El tráfico ahora debe fluir a otra red vía $R2$ como se muestra en d). Hasta que $R2$ reporte la falla de $R1$ continuará dirigiendo el tráfico a otros nodos en la red de $R1$. Como $R1$ conoce que su enlace se ha roto hay una ruta alternativa vía $R2$, envía su tráfico de vuelta a $R2$ resultando en un ciclo de tráfico como se muestra en el b). En c) se actualiza la información que ha alcanzado a $R3$ antes que a $R2$. Después $R3$ comienza a enviar tráfico destinado para otras redes a $R2$. Como

$R2$ es todavía inconsistente de que $R1$ ha fallado, continua enviando su tráfico a $R1$, el cuál regresa a $R2$. Finalmente, en d) la información acerca de la falla del enlace alcanza $R2$ y este detiene el envío de tráfico a otras redes a $R1$ y usa su propio enlace. Entonces el ciclo ha desaparecido [HMMD02].

1.8 Resumen

En este capítulo se presentan conceptos generales de teoría de grafos y matrices, en estos conceptos se basa la representación matemática de la topología del SDM que se consideró para el diseño de *AEC*. Se introducen conceptos de consenso, los cuales fueron necesarios para el diseño de *AEC*. Se incluyen conceptos de encaminamiento en redes y algoritmos de búsqueda de rutas, así como la definición de ciclos de encaminamiento.

"No quiero caminar entre locos", dijo Alicia. "Oh, no puedes hacer nada", le respondió el gato, "todos estamos locos aquí".

Alicia en el pais de las Maravillas.
Lewis Carroll.

CAPÍTULO

2

Trabajo relacionado

En el presente capítulo se ofrece una reseña del "estado del arte" referente a los diferentes protocolos de ruteo Ad Hoc existentes, describiendo el funcionamiento de aquellos protocolos que fueron precursores en el tema y que ofrecieron las base para el desarrollo del protocolo *AEC*.

El trabajo de investigación realizado se encuentra dentro del área de redes y sistemas distribuidos. Se revisaron conceptos sobre encaminamiento en redes y protocolos de encaminamiento. Como se menciona en el capítulo , sección existe gran cantidad de algoritmos de encaminamiento para redes Ad Hoc, los cuales no son la mejor opción para usarse en SDM. En este capítulo se describen los principales algoritmos de encaminamiento, sus enfoques, así como el algoritmo de consenso en el que se basó este trabajo.

2.1 Algoritmos de encaminamiento

La tabla de encaminamiento de un nodo puede construirse de dos maneras. Una posibilidad consiste en que el administrador (por medio de *scripts* que se ejecutan al inicializar el sistema, o por medio de comandos ejecutados interactivamente) introduzca manualmente las entradas de la tabla. Esta técnica se denomina encaminamiento estático, debido a que la tabla de encaminamiento se construye cuando

la computadora se prende y no varía con el tiempo. La otra posibilidad es ejecutar en cada nodo un programa que actualice automática y periódicamente la tabla de encaminamiento. Dichos programas se basan en el hecho de que una computadora siempre tiene acceso a otras computadoras conectadas a la red local; esto se traduce en que las tablas de encaminamiento contienen inicialmente al menos las direcciones de las redes locales [Sib02].

De esta forma, si todos los nodos de la red ejecutan un programa de estas características (llamado demonio de encaminamiento) al cabo de cierto tiempo habrán *descubierto* por si mismas la estructura de la red y construido sus tablas automáticamente. Mas aún, si se produjera algún cambio en la estructura de la red, bastaría con que alguna de las computadoras lo detectara para que en pocos segundos esa nueva información se propagará por toda la red [Sib02].

Por ejemplo, la ruta utilizada para enviar un mensaje del sitio A al sitio B sólo se elige cuando se envía un mensaje. Debido a que la decisión se toma dinámicamente, a distintos mensajes se les pueden asignar rutas diferentes. El sitio A tomará una decisión para enviar el mensaje al sitio C; éste a su vez, decidirá enviarlo al sitio D y así sucesivamente. Con el tiempo, un sitio entregará el mensaje al sitio B. En general, un sitio envía un mensaje a otro sitio sobre el enlace menos utilizado en ese momento en particular. Los mensajes pueden llegar en cualquier orden, este problema se soluciona anexando un número de secuencia a cada mensaje. Esta estrategia se denomina encaminamiento dinámico o adaptativo y tiene la ventaja de que, al ser automático, permite eliminar las tareas administrativas relacionadas con el mantenimiento de las tablas de encaminamiento, sin embargo, este tipo de encaminamiento es el más complejo de preparar y correr [Sib02].

En redes de comunicación, un término genérico para referirse a la medida de distancia (sin asignar una unidad) se le denomina costo, costo del enlace, costo de distancia o métrica de enlace [MR07].

2.1.1 Algoritmos de encaminamiento adaptativo

Los algoritmos de encaminamiento dinámicos cambian el camino según los cambios en el tráfico de la red o de la topología. Un algoritmo dinámico puede ser ejecutado ya sea de manera periódica o de manera directa con respuesta a los cambios

de topología o cambio en los costos de enlace, estos algoritmos son más susceptibles a problemas tales como oscilaciones en las rutas, y lazos en las rutas.

Encaminamiento óptimo alternante

Considera cualquier nodo fuente y cualquier nodo destino, con n rutas entre ellos, indexados por $i = 1, 2, \cdots, n$. El problema más simple y óptimo de distribuir el tráfico en el nodo de entrada a las rutas disponibles para minimizar el retardo promedio desde el momento que la fuente entra hasta que llega a su destino [Agn76].

En este modelo se asume que los mensajes llegan con una función de densidad de probabilidad de Poisson al nodo fuente con una razón γ por segundo y tienen longitudes distribuidas exponencialmente de $\frac{1}{\mu}$ bits. Los mensajes son puestos en una cola de uno de los n *buffers*, correspondientes a las rutas de salida. Se asume que cada *buffer* tiene capacidad infinita. En cada ruta el primer enlace cambia la capacidad de H_i bps, $i = 1, 2, \cdots, n$, el tiempo gastado en el nodo fuente por mensaje asignado a la i-ésima ruta es T_i segundos, este valor es calculado como :

$$T_i = \frac{1}{\mu H_i - \lambda_i}. \tag{2.1}$$

Donde λ_i es el radio en el cual el tráfico es asignado a la i-ésima ruta en mensajes por segundo [Agn76].

Adicionalmente a T_i, los mensajes al tomar la i-ésima ruta tienen un retardo adicional de T_i' después de que dejan el nodo fuente. Este retardo incluye el retardo de propagación y procesamiento y retardos en nodos intermedios de la ruta i, entonces la suma de $T_i + T_i'$ es el tiempo de transmisión sobre la ruta i. El retardo de transmisión promedio está dado por la fórmula:

$$T = \frac{1}{\gamma} \sum_{i=1}^{n} \lambda_i (T_i + T_i'). \tag{2.2}$$

El problema de encaminamiento óptimo es minimizar T, sujeto a las restricciones:

$$\gamma = \sum_{i=1}^{n} \lambda_i, 0 \leq \lambda_i \leq \mu H_i, i = 1, 2, \cdots, n. \tag{2.3}$$

Para este algoritmo se usa el Teorema de Kuhn-Tucker para estudiar las condiciones en las cuales el encaminamiento adaptable es satisfecho [Agn76].

Al menos que el nodo fuente se encuentre saturado (i.e. $\gamma \geq \sum_i \mu H_i$), es fácil ver que $\lambda_i < \mu H_i$ se mantiene para todas las i porque un encaminamiento óptimo debe mantener cualquier enlace proveniente de un nodo saturado, antes que cualquier otro enlace [Agn76].

En este esquema, en vez de que cada nodo en la red mantenga una tabla con una entrada para cada posible destino y para cada posible vecino; la entrada en la tabla para la localización (i, j) es el tiempo estimado por un mensaje enviado vía el i-ésimo vecino para alcanzar el j-ésimo destino, incluyendo el tiempo perdido en el nodo que envía la espera de transmisión. La tabla se mantiene actualizada por los vecinos, que periódicamente intercambian sus estimaciones de tiempo mínimo para llegar a cada destino y usan esta información para actualizar la tabla de encaminamiento de cada nodo. Cuando sólo hay un destino, la tabla de encaminamiento en el nodo que envía, es un vector cuyo elemento i-ésimo es T_i', más una función lineal da una estimación del tiempo de espera en el *buffer* $T_i = (1 + L_i)/\mu H_i$, donde hay L_i mensajes en el *buffer* (incluyendo el que está en servicio) [Agn76].

La tabla se usa de la siguiente manera: a su llegada al nodo receptor, el mensaje es reconocido y el nodo consulta la dirección de destino del mensaje. Si es la j-ésima dirección, el nodo consulta la columna j de la tabla de encaminamiento, elige la entrada mínima en la columna y coloca el mensaje en la cola de la línea de transmisión asociada. Por último, el nodo actualiza la tabla de encaminamiento para reflejar el nuevo estado de la cola. Podemos pensar en el nodo como asignar el mensaje a la ruta i para la que el valor actual de $T_i + T_i'$ es un mínimo [Agn76].

Haciendo referencia a las hipótesis formuladas en la búsqueda de la ruta óptima, es decir, las llegadas con la función de densidad con distribución de Poisson y los mensajes de longitud exponencial, se tiene que el tiempo de transmisión a través de varias rutas deben satisfacer las siguientes condiciones promedio:

$$\frac{1}{(\mu H_i - \lambda_i) + T_i} = \frac{1}{(\mu H_j - \lambda_j) + T_j'} \text{ si } \lambda_i > 0 \text{ y } \lambda_j > 0. \tag{2.4}$$

$$\frac{1}{(\mu H_i - \lambda_i) + T_i} < \frac{1}{(\mu H_j - \lambda_j) + T_j'} \text{ si } \lambda_i = 0 \text{ y } \lambda_j = 0. \tag{2.5}$$

Esto es el tiempo esperado por el mensaje para viajar desde la fuente hasta el nodo destino es el mismo sobre cualquier ruta, es el mismo a través de cualquier fuente

para la cual el tráfico es asignado y es menor que el tiempo de viaje por una ruta que no tiene tráfico asignado a ella [Agn76].

Esta propiedad de equilibrio del encaminamiento adaptativo no corresponde a las condiciones necesarias satisfechas por el encaminamiento óptimo. Éste es un resultado sorprendente, ya que parecería que un algoritmo en que las rutas de cada mensaje han sido minimizadas con el fin de reducir el retraso de transmisión que, además, para minimizar el óptimo global es que para cada minimización individual tomando en cuenta la tabla de encaminamiento, tendrían efecto sobre las opciones de futuro. Colocando un mensaje en la cola de cualquier canal de salida agrega un retardo adicional en los subsecuentes mensajes, los cuales llegan antes de que la cola se disipe. Como resultado, los mensajes subsecuentes deben unirse a la cola y esperar una retraso adicional en este nodo para transmitir o seleccionar una ruta alternativa la cual agrega un retardo extra en algún otro sitio de la red [Agn76].

Este esquema de actualización de la tabla de encaminamiento se llama encaminamiento cuadrático, esto por la forma funcional:

$$S \triangleq (1/\mu H_i)(1 + L_i)(1 + \frac{L_i}{2}). \tag{2.6}$$

La cuál es una función cuadrática de la cola de longitud L_i y las condiciones de equilibrio cambia a:

$$\bar{S}_i + T_i' = \bar{S}_j + T_j' \text{ si } \lambda_i > 0 \text{ y } \lambda_j > 0. \tag{2.7}$$

$$\bar{S}_i + T_i' < \bar{S}_j + T_j' \text{ si } \lambda_i > 0 \text{ y } \lambda_j = 0. \tag{2.8}$$

y corresponde a las condiciones necesarias para nuestra asignación óptima. Entonces el encaminamiento equilibrado es también óptimo [Agn76].

2.1.2 Algoritmo de inundación

Se trata de un método de encaminamiento bastante simple. Al recibir un paquete, el nodo lo retransmite por todos sus enlaces, excepto aquél por el que le llegó el paquete. El principal inconveniente que plantea este método es el gran número de paquetes que se generan, que llegaría a ser infinito si no se establece alguna forma de limitación. Existen diversas posibilidades para ello. Por ejemplo, cada nodo puede mantener una lista de los paquetes ya transmitidos, y al recibir un duplicado

destruirlo y no retransmitirlo. Otra posibilidad, más simple, es limitar el tiempo de vida del paquete. En uno de sus campos puede incluirse un contador de saltos, que se decrementará cada vez que el paquete atraviese un enlace; Cuando el contador llega a cero el paquete se descarta. El valor del contador puede inicializarse al diámetro de la red [PSJ99].

Este método de encaminamiento permite encontrar todas las rutas posibles entre origen y destino, entre ellas la ruta mínima; por lo que puede utilizarse como métrica para comparar con otros métodos o para establecer la ruta de un circuito virtual. Por el mismo motivo es, además, muy robusto lo que hace adecuada su aplicación en entornos militares. El gran número de paquetes que se generan al utilizar este tipo de encaminamiento presenta el inconveniente de que, en condiciones de carga alta de la subred, puede incrementar sensiblemente el retardo de los paquetes transportados por ella [PSJ99].

Usenet y P2P (peer-to-peer) utilizan las inundaciones, así como los protocolos de encaminamiento como OSPF (Open Shortest Path First), DVMRP (Distance Vector Multicast Routing Protocol) y redes Ad Hoc inalámbricas [PSJ99]. Debido a que los paquetes son enviados a través de cada enlace de salida se desaprovecha el ancho de banda. La inundación puede aumentar el tráfico exponencialmente, sobre todo sí existen ciclos en la topología, ya que en ese caso se envían paquetes duplicados.

Es posible prevenir esto, limitando el número de saltos o en el tiempo de vida, ya que las copias duplicadas podrán circular dentro de la red sin parar. Otra posibilidad es identificar los paquetes, por ejemplo numerándolos, para que cada router mantenga una lista de los paquetes enviados y así puede evitar reenviarlos de nuevo, asegurándose de que un paquete pasa a través de él una sola vez. También puede usarse inundación selectiva, en el que el paquete se envía sólo por las líneas que aproximadamente apuntan en la dirección correcta [PSJ99].

El algoritmo de inundación tiene las siguientes ventajas. En primer lugar, no es necesario conocer el estado del enlace, las distribuciones de información, ni realizar cálculos complejos de ruta, por lo tanto se reduce los gastos generales de funcionamiento y complejidad de la aplicación.

En segundo lugar, los routers no están obligados a mantener la base de datos de la información de estado del enlace, con el consiguiente ahorro espacio de almacenamiento.

En tercer lugar, se acorta la ruta de búsqueda la conexión tiempo de configuración y puede seleccionar la mejor ruta disponible basada en múltiples criterios de calidad de servicio. Desde la última, y por lo tanto a más precisa, la información del estado del enlace se utiliza para la determinación de una nueva conexión, este puede encontrar una ruta cualificada [PSJ99].

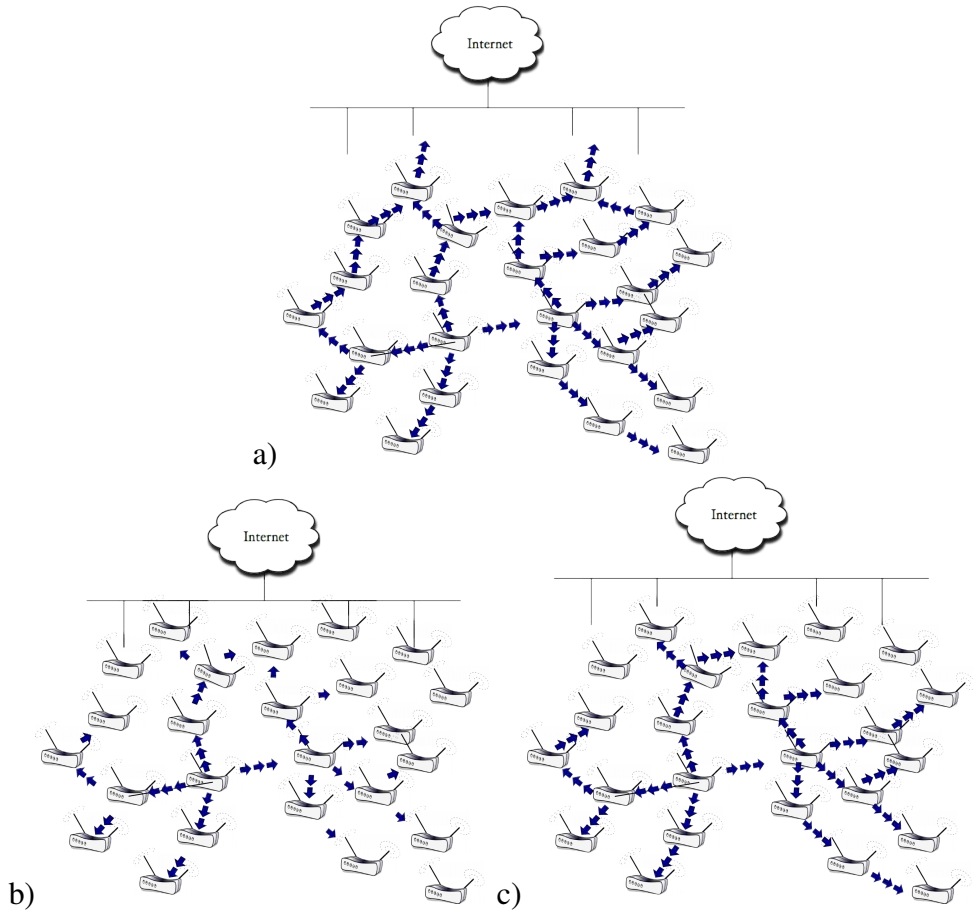


Figura 2.1: a) Inundación clásica . b) Inundación temporal. c) Inundación espacial.

Existen diferentes enfoques para el algoritmo de inundación, el enfoque clásico se muestra en el inciso a) de la Figura 2.1. En este la inundación se extiende por toda la red. Esto puede provocar problemas de escalamiento, en especial si se de

producen cambios en el medio.

Se han identificado 2 esquemas que reducen los efectos negativos de la inundación. El primer enfoque es la inundación temporal, que disminuye la inundación conforme aumenta la distancia con el router que la generó (inciso b) de la Figura 2.1).

En el caso de la inundación espacial, conforme los routers se encuentran más lejanos, estos reciben información menos detallada y menos precisa (inciso c) de la Figura 2.1).

2.1.3 Encaminamiento Epidémico

El encaminamiento Epidémico distribuye mensajes de aplicación a *host* que son llamados "cargados" en regiones de redes Ad Hoc, de esta manera el mensaje se propaga por regiones de la red. Existe una alta probabilidad de que el mensaje llegue a su destino. El propósito es maximizar la taza de entrega y minimizar la latencia de entrega. Cada *host* mantiene un *buffer* que contiene mensajes que se originan como mensajes. Hay una llave única para cada mensaje, cada *host* guarda un vector de bit que indica que mensajes están guardados en una tabla *hash* local. Para evitar carga extra en cada *host* se mantiene un cota de tamaño fijo. Suponiendo que la red contiene un nodo A y un nodo B , cuando el *host* A hace contacto con el *host* B, se transmite un vector SV_A a B. B hace una operación AND con su propio vector, así obtienen un vector con el conjunto diferencia entre los mensajes de A, se solicitan los mensajes faltantes a B. Estas sesiones anti-entropía garantizan que eventualmente el mensaje se entregará al intercambiarse con un par. Existe un contador de saltos que limita el número de intercambios de un mensaje en particular [VB00].

2.2 Protocolos de encaminamiento

Los protocolos de encaminamiento determinan la ruta óptima a través de la red usando algoritmos de encaminamiento e información de transporte sobre estas rutas. Los protocolos de encaminamiento funcionan en la capa de red del modelo de referencia OSI. Ellos usan información específica de la capa de red, incluyendo

direcciones de red, para mover unidades de información a través de la red. Nuestro problema se encuentra ubicado en la capa de aplicación del modelo de referencia OSI.

2.2.1 Vector de distancia (RIP)

La idea detrás del algoritmo de vector de distancia es sugerido por su nombre [1]. Cada nodo construye un arreglo unidimensional (un vector) que contiene las distancias (costos) hacia todos los nodos y distribuye ese vector a sus vecinos inmediatos. Inicialmente se asume que cada nodo conoce el costo del enlace a cada uno de sus vecinos inmediatos, un enlace caído se considera con costo infinito [PD07].

El protocolo de encaminamiento en redes IP RIP (*Routing Information Protocol*) es el ejemplo canónico de un protocolo de Internet construido a partir del algoritmo de Bellman-Ford descrito anteriormente [PD07].

Los protocolos de encaminamiento entre redes difieren del modelo de grafo que se describió con anterioridad en la sección 2.1. En una red de redes, la meta de los enrutadores es aprender como reenviar lo paquetes a varias redes, por lo tanto, en vez de considerar el costo de llegar a otros enrutadores, los enrutadores difunden el costo de alcanzar la red. Por ejemplo, en la Figura 2.2, el enrutador C debe informar al enrutador A del hecho de que se pueden alcanzar las redes 2 y 3 (que están directamente conectadas) con un costo de 0, las redes 5 y 6 tienen costo de 1; las redes 4 y 2 tienen costo de 2 [PD07].

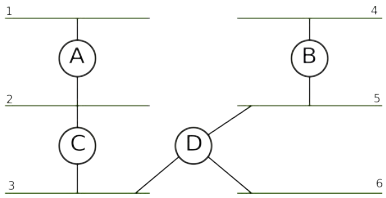


Figura 2.2: Ejemplo de una red ejecutando RIP.

RIP es la implementación más directa del encaminamiento por vector de distancia. Los enrutadores que utilizan RIP envían sus avisos cada 30 segundos, el

[1] Esta basado en el algoritmo de Bellman-Ford

enrutador también envía sus avisos cuando hay algún cambio en su tabla de encaminamiento. En la práctica, el protocolo busca alcanzar el objetivo en menos de 16 saltos, de hecho 16 saltos es equivalente a la distancia infinita [PD07].

Entre los protocolos utilizados para redes Ad Hoc basado en el protocolo de vector de distancia se encuentra el protocolo AODV:

AODV (*Ad hoc On Demand Distance Vector*)

Es un protocolo de encaminamiento diseñado para redes Ad Hoc. Este protocolo es capaz de realizar encaminamiento unicast y multicast. Es un protocolo reactivo, es decir, sólo establece una ruta a un destino si esta se le solicita [Sun97].

El algoritmo AODV permite a los nodos móviles obtener rutas rápidamente para nuevos destinos, y no exige a los nodos mantener las rutas a los destinos que no están en la comunicación activa. Adicionalmente, AODV permite para la formación de grupos multicast donde el número de miembros es libre de cambiar durante la vida de la red. AODV también define las contestaciones para roturas y cambios en la topología de la red. El funcionamiento de AODV es libre de ciclos, y evita el problema Bellman-Ford Contando al infinito y ofrece la convergencia rápida cuando la topología de la red cambia (típicamente, cuando un nodo se mueve en la red) [Sun97].

Una característica de AODV es el uso de un número de secuencia destino para cada entrada de la tabla de encaminamiento. El número de secuencia destino es creado por el destino o el *grouphead* del *multicast* para cualquier información de la ruta utilizable que envía a pedir los nodos. Usando el número de secuencia destino aseguran que sea libre de ciclos. Dada la opción entre dos rutas a un destino, un nodo solicitante siempre selecciona el que tenga un número de secuencia más grande [Sun97].

Otra característica de AODV es que las roturas en las conectividades causan de inmediato notificaciones, enviado al conjunto de nodos afectados, pero sólo a ese conjunto de nodos [Sun97].

Route Requests (RREQs), *Route Replies* (RREPs) y *Multicast Route Invalidations* (MINVs) son los tres tipos del mensaje definidos por AODV. Estos tipos

mensajes son manejados por UDP e IP normal. Así, por ejemplo, el nodo solicitante espera usar su dirección IP como la IP fuente para los mensajes. El rango de alcance del *broadcast* para los RREQs puede ser indicado por el TTL en la cabeza de IP. La fragmentación es típicamente no requerida [Sun97].

Dado que los *endpoints* de una conexión t rutas válidas para tienen cada uno, AODV no juega ningún papel. Cuando una ruta a un nuevo destino (un sólo nodo o un grupo *multicast*) es necesaria, el nodo usa una transmisión *broadcast* y manda un RREQ para encontrar una ruta al destino. Una ruta puede determinarse cuando la solicitud alcanza ya sea, el destino mismo o un nodo intermedio con una ruta actual hacia el destino. La ruta es hecha disponible para mandar de forma unicasting un RREP hacia a la fuente del RREQ. Desde que cada nodo que recibe la solicitud (un RREQ) guarda una ruta hacia la fuente de la solicitud, el RREP puede ser unicast del destino a la fuente, o de cualquier nodo del intermedio que puede satisfacer la solicitud de la fuente. En el escenario *multicast*, RREQs son también usados cuando un nodo desea unirse a un grupo *multicast*. Una bandera en el RREQ permite los nodos saber que cuando ellos reciben el RREP, ellos no están poniendo simplemente indicadores de la ruta, sino realmente están insertando una rama al árbol del *multicast* [Sun97].

2.2.2 Estado de enlace (OSPF)

El encaminamiento por estado de enlace es el segundo más usado en los protocolos de encaminamiento. Las consideraciones iniciales son las mismas que en el de vector de distancia. Se asume que cada nodo es capaz de detectar el estado de los enlace que lo comunican con los vecinos inmediatos (conectado o desconectado) y el costo de cada enlace. La idea es simple, cada nodo conoce como llegar directamente a sus vecinos inmediatos y para tener el conocimiento total, lo único que hay que hacer es difundir esta información por toda la red, cuando cada nodo tenga toda la información será capaz de armar su propio mapa de la red entera. Este protocolo está constituido de 2 mecanismos: la diseminación segura de la información del estado de los enlaces y el cálculo de rutas desde la suma de todos los estados de los enlaces conocidos [PD07].

En la práctica, cada *switch* calcula su propia tabla de encaminamiento independientemente desde el paquete de estado del enlace [1] y es usado para la realización del algoritmo de Dijkstra llamado algoritmo se búsqueda hacia adelante, que mantiene 2 listas que contiene un conjunto de entradas de la forma (Destino, Costo, Siguiente Salto), este protocolo opera como se explicó en la subsección 1.5.2, [PD07].

The Open Shortest Path First Protocol (OSPF) es uno de los estándares más utilizados, sobre todo por ser un estándar no propietario [PD07].

Uno de los protocolos para redes Ad Hoc basados en el protocolo de estado de enlace es el OLSR:

OLSR (*Optimized Link State Routing*)

Es un protocolo de encaminamiento pro-activo, que trabaja en forma distribuida para establecer las conexiones entre los nodos en una red inalámbrica Ad Hoc. La diseminación directa de información por toda la red (*flooding*) es ineficiente y muy costosa en una red inalámbrica y móvil, debido a las limitaciones de ancho de banda y la escasa calidad del canal. OLSR provee un mecanismo eficiente de diseminación de información basado en el esquema de los *Multipoint Relays* (MPR) [JMC+01].

Bajo este esquema, en lugar de permitir que cada nodo retransmita cualquier mensaje que reciba (*flooding* clásico), todos los nodos de la red seleccionan entre sus vecinos un conjunto de *multipoint relays* (retransmisores), encargados de retransmitir los mensajes que envía el nodo en cuestión. Los demás vecinos del nodo no pueden retransmitir, lo que reduce el tráfico generado por una operación de *flooding* [JMC+01].

Hay varias formas de escoger los *multipoint relays* de un nodo, pero independientemente de la forma de elección, el conjunto de MPRs de un nodo debe verificar que son capaces de alcanzar a todos los vecinos situados a una distancia de 2 saltos del nodo que los calcula (criterio de cobertura de MPR) [JMC+01].

[1] LSP: Cada nodo crea un paquete de actualización, que contiene el ID del nodo, una lista de los vecinos directamente conectados y el costo de los enlaces, un número de secuencia y el tiempo de vida del paquete

Una red encaminada con OLSR utiliza básicamente dos tipos de mensajes de control:

- Los mensajes HELLO son enviados periódicamente por cada nodo de la red a sus nodos vecinos, pero nunca son retransmitidos más allá del primer salto (1 hop) desde su emisor (alcance local). Estos mensajes contienen la lista de vecinos conocidos por el nodo emisor así como la identidad de los *multipoint relays* seleccionados por transmisor. Su intercambio permite a cada nodo de la red conocer los nodos situados a 1 y 2 saltos de distancia (es decir, aquellos a los que se puede hacer llegar un mensaje con una transmisión directa o con una transmisión y una retransmisión) y saber si ha sido seleccionado como MPR por alguno de sus vecinos [JMC+01].

- Los mensajes TC (*Topology Control*) son enviados periódicamente y de forma asíncrona. A través de ellos, los nodos informan al conjunto de la red acerca de su topología cercana. Al contrario que los HELLO, los mensajes TC son de alcance global y deben llegar a todos los nodos de la red. El conjunto de los mensajes TC recibidos por un nodo inalámbrico le permite reconstruir su base de datos topológica, computar el árbol de caminos mínimos (mediante el algoritmo de Dijkstra) y calcular así la tabla de encaminamiento hacia todas las posibles destinaciones. La diseminación de mensajes TC se hace de acuerdo con el mecanismo de *flooding* basado en MPR [JMC+01].

La gran variedad de algoritmos de encaminamiento para redes móviles no consideran condiciones locales de los nodos, como el estado del planificador del nodo, por lo tanto no es posible aplicarlos eficientemente a nuestro problema.

2.2.3 TORA (*Temporally-Ordered Routing Algorithm*)

TORA es un protocolo de encaminamiento distribuido para redes móviles e inalámbricas; este protocolo apuesta por un alto grado de escalabilidad usando un “flat”, es decir, un algoritmo de encaminamiento no jerárquico. En la operación del algoritmo, se intenta suprimir la propagación de mensajes de control a los más lejanos niveles de la red. TORA construye y mantiene un grafo acíclico dirigido con raíz en el destino [PC97].

TORA (utiliza un reloj lógico para establecer un orden temporal del cambio topológico de la red, su principal característica es que cuando un enlace falla los mensajes de control sólo se propagan alrededor del punto de falla [PC97].

El protocolo TORA, es un protocolo reactivo, se caracteriza por proporcionar al nodo remitente, no sólo uno, sino múltiples trayectos para hacerle llegar al destino. El procedimiento es el siguiente: cada nodo realiza una copia de TORA para cada destino y el protocolo crea, mantiene y cancela los trayectos de encaminamiento. El TORA asocia un peso a cada nodo de la red respecto a un destino, y los mensajes se desplazan desde un nodo con mayor peso hacia uno con peso menor; mientras los caminos descubiertos con paquetes de tipo QRY (*Query*) vienen actualizados con aquellos de tipo UPD (*Update*) [PC97].

Si un nodo necesita conocer un trayecto hacia un destino manda en difusión (*broadcast*) un paquete QRY que se propaga, hasta que no alcanza el nodo destino o a un nodo que posea un trayecto válido hacia el destino. El nodo que responda se servirá a su vez de un paquete UPD que agregará también su peso. Los paquetes UPD se enviarán en difusión de modo que permitan a todos los nodos intermedios modificar su peso convenientemente. Se deriva, por tanto, que los nodos que quieran alcanzar destinos lejanos o directamente inalcanzables, aumentan su peso local hasta el máximo valor consentido, mientras que el nodo que encuentre un nodo cercano con un peso que tienda a infinito, cambiará el trayecto [PC97].

El paquete de tipo CLR (Clear) interviene en algunos casos para reiniciar todos los estados de direccionamiento de una porción de red cuando el destino sea completamente inalcanzable [PC97].

El protocolo TORA se apoya en el protocolo para redes MANET llamado IMEP (*Internet MANET Encapsulation Protocol*) que proporciona un servicio de expedición fiable para protocolos de encaminamiento [PC97].

La agregación en un único bloque de los mensajes IMEP y TORA reduce el *overhead* de la red y prueban el estado de los nodos vecinos. Para obtener tal agregación se utiliza periódicamente un intercambio de mensajes llamados BEACON y HELLO [PC97].

2.2.4 **Virtual Ring Routing (VRR)**

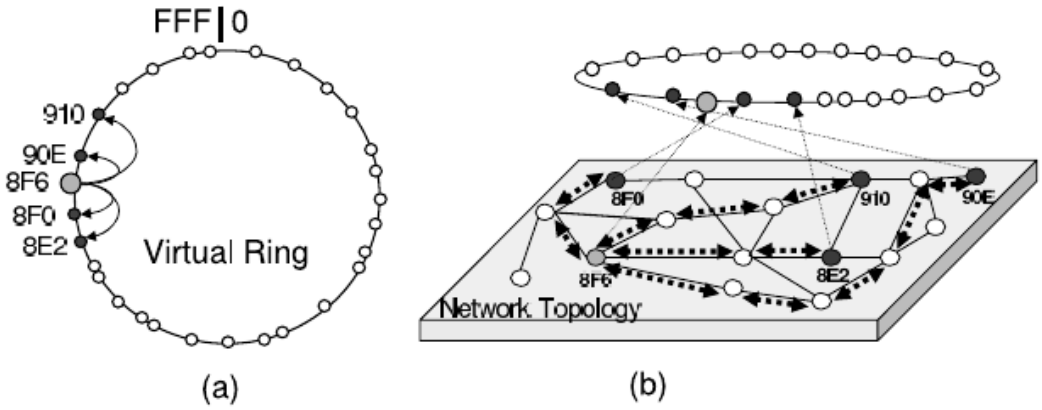


Figura 2.3: Relación entre el anillo virtual y la red física [CCN06].

Comienza planteando la necesidad de diferenciar a los nodos en forma física y lógica para lo cual hace uso de enteros aleatorios para identificar a cada nodo en la red, por ejemplo partiendo desde 0 a $FFFF$ en una red con 65535 nodos. Una vez asignados los identificadores, se forma un anillo circular desde 0 a $FFFF$, donde al ser un anillo circular el nodo $FFFF$ es el nodo anterior a 0 [CCN06].

El anillo circular lógico es completamente independiente de la topología física, y dos nodos que en el anillo circular son contiguos pueden no tener ningún contacto físico dentro de la red. Cada nodo mantiene una tabla de ruteo con dos tipos de entradas, por un lado tiene r entradas correspondientes al anillo lógico, conteniendo $r/2$ vecinos lógicos (v_{set}) del anillo a su derecha y $r/2$ vecinos lógicos (p_{set}) del anillo circular a su izquierda; y por otro lado contiene a todos sus vecinos físicos, la Figura muestra el esquema de funcionamiento de protocolo, como se muestra en la Figura 2.3 [CCN06].

También mantiene conjunto de vecinos físicos (p_{set}) con identificadores se nodos que se pueden comunicar con la capa de enlace. Como la calidad de enlace puede variar en ambientes inalámbricos, es importante para los nodos estimar la calidad del enlace inalámbrico para los candidatos a vecinos físicos. Un nodo sólo puede agregar a un vecino al conjunto p_{set} si la calidad del enlace desde el vecino hacia él está por encima del umbral. Además los nodos de VRR pueden tomar en cuenta la calidad del enlace cuando tomar decisiones de reenvío [CCN06].

VRR establece y mantiene rutas entre un nodo y cada uno de sus vecinos virtuales, esto se le llama $v_{set} - path$, como los identificadores de los nodos son aleatorios y su localización es independiente los vecinos virtuales de un nodo, estos están aleatoriamente distribuidos a través de la red física [CCN06].

Al momento de encaminar paquetes lo hace mediante el anillo lógico y mediante las entradas correspondientes a los vecinos físicos, si el destino es un nodo que existe en su tabla de ruteo el paquete será encaminado directamente al destino, y si la entrada no está en su tabla de ruteo, será enviado al vecino numéricamente más cercano, siendo este último el que continúe la secuencia de ruteo. Por otro lado, el trayecto entre 2 vecinos adyacentes en el anillo lógico es potencialmente a través de múltiples nodos, por lo cual cuando un anillo esta siendo encaminado mediante el anillo lógico, el paquete puede ser redirigido por algún nodo intermedio que conozca una mejor ruta [CCN06].

El esquema de ruteo utilizado no hace un uso eficiente de la topología física sobre la cual está desplegada la red, cada nodo solamente hace uso de sus vecinos físicos directamente conectados, y la decisión inicial de ruteo se basa en la conectividad sobre el anillo lógico. El algoritmo se basa en la probabilidad que tiene un paquete de encontrar en su camino algún nodo que tenga una mejor ruta que la provista por el anillo lógico. Dado que se basa en la probabilidad de que el paquete encuentre algún nodo que ofrezca un buen camino, existe la probabilidad de penalizar otros recorridos, forzándolos a usar el camino provisto por el anillo lógico, lo cual resulta en una decisión de ruteo ineficiente para dichos recorridos[CCN06].

2.2.5 Grid

Los servicios de localización provistos por Grid [DPH05] donde se introduce el concepto de servicios de localización para ruteo geográfico, este esquema hace uso de ruteo geográfico para localizar la posición de los nodos, cada uno de ellos debe contar con algún mecanismo para detectar cuál es su posición física (GPS) y de alguna manera poder informarla a los demás nodos. Cada nodo tendrá básicamente 2 formas de identificarse, una dirección universal y una física [Tej10].

La información de localización de alguna manera debe ser almacenada y consultada. El concepto de *hash* consistente es utilizado para localizar en forma deter-

minista uno o más lugares a partir de la dirección universal de un nodo, los nodos registran una posición geográfica en múltiples lugares, si un nodo requiere comunicarse con otro primero debe consultar su posición geográfica, para lo cual debe buscarla en la misma localización que el nodo destino utilizó para almacenar su dirección física [DPH05].

Una vez localizado un nodo en Grid, se hace uso de su posición geográfica para comunicarse con él, cada nodo en base a su posición y la de sus vecinos inmediatos, decide cuál de ellos esta físicamente más cerca del nodo destino, esta decisión puede llevar a un nodo hoja, es decir, donde la información llega un nodo que no tiene ningún vecino [DPH05].

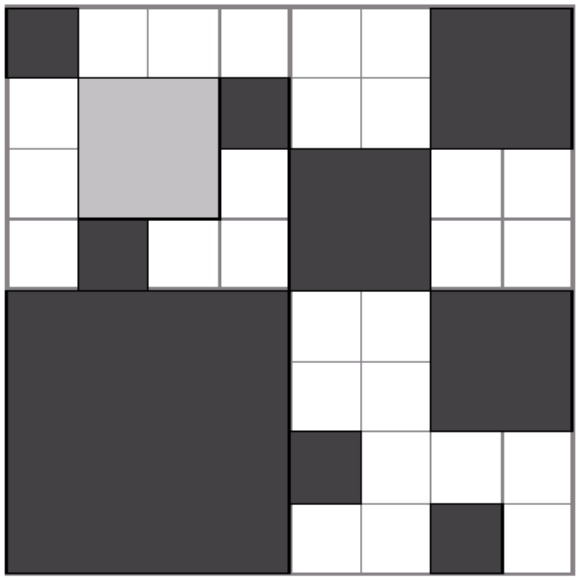


Figura 2.4: Un pedazo de la división global del mundo. Unos pocos ejemplos de cuadrados de varias órdenes se muestran con sombreado oscuro. El cuadrado ligeramente sombreado se muestra como un ejemplo de un cuadrado de 2×2 que no es un cuadrado orden-2 debido a su ubicación. Las coordenadas de la esquina inferior izquierda una orden-n del cuadrado deben ser de la forma $a2^{n-1}, b2^{n-1}$ para enteros a, b [DPH05].

Grid necesita un esquema que provee direccionamiento y ruteo, para lo cual hace uso de ruteo geográfico, el problema del ruteo geográfico es su dependencia con la densidad de la red, siendo necesario tener una red densa para disminuir la posibilidad de encontrar un nodo hoja. Grid propone una red organizada en secto-

res de diferente orden, siendo el particionamiento en sectores conocido para todos los nodos al momento de iniciarse el protocolo. Un nodo cualquiera, al determinar los nodos que serán los encargados de almacenar su dirección geográfica debe conocer en que tipo de sección se encuentra, cuantos tipos de sectores hay y quien se encuentra dentro, lo cual es un grave problema ya que implica el consenso de numerosos equipos sobre la jerarquía existente y división en sectores de la misma,como se muestra en la Figura 2.4 [DPH05] .

El protocolo tiene la necesidad de recorrer toda el área para identificar tanto los niveles de las áreas como sus integrantes, y de esa manera encontrar quienes son los posibles servidores de localización con ID cercano al de cada nodo, claramente la solución no es escalable a grandes redes ni adecuada a situaciones donde los participantes entran y salen de la red constantemente [DPH05] .

2.2.6 Dynamic Source Routing protocol (DSR)

Es un protocolo de encaminamiento simple y eficiente diseñado específicamente para su uso en redes Ad Hoc multi-hop inalámbricas de nodos móviles. DSR [JHM07] permite a la red ser completamente auto-organizada y auto-configurable, sin necesidad de ninguna infraestructura de red existente o de administración.

El protocolo se compone de los dos mecanismos principales de ”Descubrimiento de rutaz ”Mantenimiento de Ruta”, que trabajan juntos para permitir que a los nodos descubrir y mantener rutas a destinos arbitrarios en la red Ad Hoc. Todos los aspectos del protocolo operan exclusivamente en la demanda, lo que permite la sobrecarga de paquetes de encaminamiento de DSR para escalar automáticamente sólo lo que se necesita para reaccionar a los cambios en las rutas actualmente en uso. El protocolo permite que múltiples rutas a cualquier destino y permite que cada remitente para seleccionar y controlar las rutas utilizadas en el encaminamiento de sus paquetes, por ejemplo, para su uso en el equilibrio de carga o para una mayor robustez. Otras ventajas del protocolo DSR es que es libre de ciclos de encaminamiento, dado que en la red los enlaces son unidireccionales, el uso de un .[es]tado suave.[en] el encaminamiento y la recuperación rápida de las rutas cuando hay un cambio en la red. El protocolo DSR está diseñado principalmente para redes móvi-

les Ad Hoc de hasta unos doscientos nodos y está diseñado para funcionar bien incluso con tasas muy altas de movilidad.

2.2.7 Whirlpool Routing Protocol (WARP)

La idea principal del diseño de WARP [LKA+10] es que el tráfico de datos se puede utilizar para reparar una ruta ya existente e investigar de manera eficiente la topología y comunicar las rutas rotas con segmentos de ruta cercanos. Proporciona un servicio de datagramas best-effort. WARP es un protocolo de vector distancia: los nodos mantienen una medida de la ”distancia.ª un destino. Esto construye un árbol de encaminamiento en torno a un destino. Cuando un destino es estacionario, WARP opera como un protocolo de encaminamiento estándar. Los nodos estiman los costos de enlaces, los costos de ruta de calculo y enviar los paquetes a lo largo de la ruta de costo mínimo a un destino dado. WARP es un protocolo reactivo, ya que opera con vectores de distancia, el protocolo reacciona cuando un nodo detecta un destino se ha movido. En lugar de enrutar las solicitudes de inundación o emitir otro tipo de tráfico de control, WARP utiliza paquetes de datos para investigar la topología de la red, utilizando el algoritmo de encaminamiento Whirlpool para encontrar de forma rápida y eficiente el destino. WARP tiene un registro sobre la posición del nodo destino, cuando este se mueve hace una espiral entorno a la antigua localización del nodo destino, de esta manera busca un vecino que tenga aun comunicación con el nodo que se movió; se requiere una actualización continua del vector de distancia.

2.2.8 Encaminamiento adaptativo en MANETS basado en toma de decisiones

La solución propuesta en [YFG+10] se centra en el aprovechamiento de los algoritmos de encaminamiento existentes. Este enfoque se basa en cambio entre Los algoritmos de encaminamiento en tiempo real, con el fin de lograr el mejor rendimiento de encaminamiento bajo condiciones cambiantes de la red. todos nodos de la red utilizan los mismos algoritmos de encaminamiento que se consideran como

el conjunto de alternativas en la investigación previa, que son OLSR [JMC+01], DSR [JHM07], y AODV [Sun97].

El algoritmo utiliza votaciones para determinar que algoritmo escogerá para ciertas condiciones de la red, las votaciones son por mayoría, se le asigna un peso a cada nodo, este peso depende del número de vecinos que tiene, esto incrementa su influencia en la votación.

2.3 Algoritmos que utilizan consenso

En esta sección se revisan algunos trabajos que utilizan consenso para resolver problemas de redes inalámbricas.

2.3.1 Consenso en redes oportunistas

Se define una red oportunista como una red inalámbrica Ad Hoc, en la cual los dispositivos móviles tienen contactos transitorios e impredecibles, estos contactos son la única oportunidad de intercambiar información. Cada nodo lleva mensajes para propagar por la red. La desconexión eventual opermanente de un dispositivo no es una falla es un comportamiento esperado. Se espera alcanzar el vecindario inmediato del nodo destino, esto sugiere una visión global. Se llama sesión al proceso entero de comenzar el consenso y tomar la decisión, se usa como algoritmo de encaminamiento "Encaminamiento Epidémico" [VB00]. En el consenso, el primer paso es enviar S_p^r en el cual el proceso p envía su contribución para la ronda r a otro proceso, seguido de un paso de transición T_p^r , siempre que han recibido suficientes contribuciones de los otros procesos, el proceso de p toma una decisión o determina sus contribuciones para la próxima ronda y procede a esa ronda. Un nodo p puede tolerar no recibir mensajes de hasta a un tercio de los otros participantes en ronda r, mientras que todavía poder decidir o pasar a la siguiente ronda. En la práctica, en una red oportunista esto puede ocurrir porque algunos de los otros participantes no han alcanzado todavía ronda r (por ejemplo, debido a que estos participantes están actualmente en modo de suspensión), o porque algunos participantes de hecho han enviado sus contribuciones para la ronda r pero estos mensajes no tienen alcanzado nodo p todavía (y posiblemente nunca lo hará) [BLG15].

2.3.2 Consensus routing: The Internet as a distributed system

El encaminamiento de Internet, en especial encaminamiento entre dominios, ha favorecido tradicionalmente a la capacidad de respuesta, es decir, la rapidez con la red reacciona a los cambios, más de consistencia, es decir, garantizar que los paquetes atraviesen rutas aprobadas.

Un router aplica una actualización recibida de inmediato a su tabla de reenvío antes propagar la actualización a otros routers, incluyendo aquellos que potencialmente dependen del resultado de la actualización. La capacidad de respuesta se produce a costa de disponibilidad: un router [JKBK$^+$08].

Un router A piensa que su ruta a un destino es vía B pero B no está de acuerdo, sin embargo puede ser por dos motivos.

1. La ruta vieja de B al destino vía A causa ciclos.

2. B no tiene una ruta actual al destino causando un hoyo negro. BGP actualiza se conoce que causa el 30 % de pérdidas de paquetes por dos minutos o más después de un cambio en la tabla de encaminamiento inclusive si existe una ruta física viable. Además, los ciclos transitorios se contabilizan por el 90 % de todos los paquetes perdidos.

El comportamiento de un protocolo es complejo e impredecible como los routers por el diseño operan sobre estados distribuidos inconsistentes, por ejemplo enviando paquetes a través de ciclos. No hay indicador de cuando la red converge a un estado consistente. Un comportamiento impredecible hace al sistema más vulnerable a la desconfiguración, es difícil distinguir entre un comportamiento esperando y uno no esperado. La meta de un simple y práctico protocolo de encaminamiento que permite políticas de encaminamiento generales y alcanza una alta disponibilidad [JKBK$^+$08].

El encaminamiento por consenso se enfoca en la separación de dos modos distintos de envío de paquetes:

1. Un modo estable asegura que una ruta es adoptada sólo después de que todos los routers dependientes han alcanzado una vista consistente de un estado

global la salida del consenso sirve como un indicador explícito a los routers que deben adoptar un conjunto consistente de rutas procesadas antes del panorama global.

2. Un modo transitorio asegura una alta disponibilidad cuando un paquete encuentra un router que no pose una ruta estable porque el enlace correspondiente ha fallado o porque el protocolo de consenso para calcular una ruta estable no ha terminado todavía. En este caso, el router lo marca como un paquete transitorio y usa la información local acerca de las rutas disponibles heurísticamente para enviar el paquete al destino.

La principal contribución de este trabajo es un protocolo práctico de encaminamiento que permite políticas generales y alcanzar alta disponibilidad. Por ejemplo el encaminamiento por consenso es una política de encaminamiento que sistemáticamente separa de manera segura el "Liveness" [1] concerniente usando dos modos: Un modo estable para asegurar la consistencia y un modo transitorio para optimizar la disponibilidad. Seguridad significa que un router envía un paquete estrictamente a través de una ruta adoptada. "liveness" significa que la red:

1. Reacciona rápidamente a fallas o cambios en la política.

2. Asegura una alta disponibilidad, definida como la probabilidad de un paquete sea entregado exitosamente.

Por separar la seguridad y el "liveness", el encaminamiento por consenso alcanza una alta disponibilidad mientras permite a los sistemas autónomos ejecutar arbitrariamente políticas de encaminamiento. El encaminamiento por consenso alcanza esta separación usando dos ideas simples. Primero, un algoritmo de coordinación distribuida para asegura que una ruta es adoptada sólo después de que todos los routers dependientes han acordado una vista consistente de un estado global. Las rutas adoptadas son consistentes, es decir, si un router A adopta una ruta a un destino vía un router B, entonces un B adopta el correspondiente sufijo como su ruta al destino [JKBK+08]. Note que la inconsistencia implica que la ruta está libre de ciclos. Se envían paquetes usando uno de los dos distintos modos:

[1] "liveness" significa que el sistema reacciona rápidamente a fallas o cambios en la política

1. Un modo estable que sólo usa rutas consistentes calculadas usando el algoritmo de coordinación.

2. Un modo transitorio que heurísticamente envía paquetes cuando una ruta estable no está disponible.

Del conjunto de *instantáneas* distribuidos que los sistemas autónomos toman y envían, algunos de ellos pueden estar completos y algunos otros pueden estar aún en progreso, por ejemplo, las actualizaciones que están en tránsito cuando el panorama distribuido ha sido tomado, o están esperando en relojes locales que expiren antes de ser enviados a los routers vecinos. El algoritmo de consenso se usa para calcular un estado global del sistema a partir de las instantáneas distribuidas que cada sistema autónomo envía [JKBK$^+$08].

2.3.3 Algoritmo de votaciones por promedio ponderado

En el contexto de los algoritmos de consenso, uno de los trabajos más destacados para nuestro caso de estudio es el algoritmo de votaciones por promedio ponderado [LSBB01]. Este algoritmo propone una novedosa forma de incorporar el consenso utilizando el promedio de los votos que recibe como valores iniciales.

Comienza con n entradas y calcula la media ponderada de los resultados del módulo en cualquier ciclo electoral. Un factor de ponderación, w_i es asignado a todos los votantes de entrada x_i y la salida y final se calcula como $y = \frac{\sum w_i \cdot x_i}{\sum w_i}$. Los pesos pueden ser predeterminados o se puede ajustar de forma dinámica. Los pesos determinados puede basarse en estimaciones a priori de la fiabilidad de los módulos o en la probabilidad a priori de la falla de los módulos redundantes. Para el ajuste dinámico se sugieren varias estrategias. Entre ellas calcular el peso sobre la base de las distancias entre los resultados del módulo $d_{ij} = |x_i - x_j| : i, j = 1, 2, \ldots, N$ e $i \neq j$. Un valor de entrada que difiere mucho de los valores de otras entradas se le asigna un peso menor que un valor de entrada que se encuentra cerca de cualquiera de los otros valores de entrada. Entre los enfoques de ponderación de distancia basados en mediciones, se utiliza una modificación del algoritmo Lorczak. Este algoritmo utiliza la siguiente ecuación (2.9) para calcular los valores de peso de los cuales la salida del votante es calculada por la ecuación (2.10). El algoritmo usa

una aplicación especifica de un factor de tolerancia β para ajustar los valores de peso.

$$w_i = \frac{1}{1 + \prod_{i=1,j=1j\neq i}^{N} \frac{d^2ij}{\beta^2}}. \tag{2.9}$$

$$y = \frac{\sum_{i=1}^{N} w_i \cdot x_i}{\sum_{i=1}^{N} w_i} | i = 1, \ldots, N. \tag{2.10}$$

El algoritmo recibe m pares de entradas y se describe a continuación:

1. Sean $x_1, x_2, \ldots, x_m$ las entradas de los votantes y y la salida del consenso.

2. Determinar "el nivel de acuerdo" de las $m(m-1)/2$ entradas de pares de votantes, basándose en la ecuación (2.11). Esta función asigna un valor para el acuerdo de 2 entradas i, j de votantes basado en la distancia numérica entre estas.

$$s_{ij} = \frac{1}{1 + pd_{ij}^{q}}. \tag{2.11}$$

   El parámetro q ajusta el rango de extensión de esta función y p es usado para establecer el valor del punto medio de la función, a. La función es continua y genera $s_{ij} = 1$ para pares de entradas equivalentes. Para entradas distintas, se produce un valor real entre 0 y 1 tal que d_{ij} crece y s_{ij} tiende a 0. Para propósitos de comparación, se caracterizará las curvas utilizando el valor del punto medio $d_{ij} = a$ tal que $s_{ij} = 0{,}5$. Entonces $p = a^{-q}$ y $s_{ij} = \frac{1}{1+(\frac{d_{ij}}{a})^q}$. En [LSBB01] se demuestra que al aumentar el parámetro q el comportamiento de los votantes tiende a un valor estándar determinado por la mayoría de los votantes, con un umbral duro a.

   Con $q = 2$ entonces $s_{ij} = \frac{1}{1+(\frac{d_{ij}}{a})^q} = \frac{1}{1+(\frac{d_{ij}}{0,5})^2} = \frac{1}{1+4d_{ij}}$

3. Teniendo calculado el valor de acuerdo para todos los pares de votantes, se asigna el valor de peso para cada entrada i del votante basándose en 2.12

$$w_i = \frac{\sum_{j=1}^{N} s_{ij}}{m-1}. \tag{2.12}$$

   Este tipo de evaluación indica la influencia de un acuerdo entre cualquier valor de entrada y los demás valores de entrada, expresada como una función de peso.

4. Se calcula la salida del consenso dada por la ecuación (2.10).

Este algoritmo será explorado posteriormente en la propuesta del *AEC*.

2.4 Resumen

En este capítulo se exploran los distintos protocolos de encaminamiento en redes móviles, como son los basados en vector de distancia o en estado de enlace. Se explica el funcionamiento del algoritmo de inundación y se destacan las ventajas y desventajas de su uso en redes móviles. Se presenta a detalle el funcionamiento de algoritmos de encaminamiento (AODV, TORA, OSPF, VRR, WRAP y GRID) aplicados a redes móviles, un algoritmo de consenso en un entorno distribuido como el internet, otro para redes oportunistas y el algoritmo de votaciones por promedio ponderado. Este último algoritmo fue de importancia para el diseño de *AEC*.

¿Podrías decirme, por favor, qué camino debo seguir para salir de aquí? -Esto depende en gran parte del sitio al que quieras llegar- dijo el Gato. -No me importa mucho el sitio- respondió Alicia. -Entonces tampoco importa mucho el camino que tomes- le contestó el Gato.

Alicia en el pais de las Maravillas.
Lewis Carroll.

CAPÍTULO

3

Algoritmo de Encaminamiento por Consenso

En este capítulo se presenta el algoritmo de encaminamiento por consenso de manera general. Antes de la descripción del algoritmo se introducen algunos conceptos comunes en las 4 versiones del algoritmo.

Los nodos se comunican entre ellos creando enlaces con los nodos más próximos. El enlace se puede ver comprometido por varios factores:

1. El nodo está muy ocupado para atender la solicitud.
2. El enlace no es suficientemente fuerte para establecer la comunicación. Esto puede deberse a la ocupación del canal o porque el nodo se encuentra fuera del rango del nodo transmisor.

En base a lo anterior se definen los criterios de disponibilidad de los nodos como sigue:

Tiempo disponible en el planificador del nodo La tarea de transmisión y retransmisión de un mensaje a otro nodo debe tener al menos un espacio disponible

de tamaño γ. Si no existe este espacio disponible en el planificador local de tareas, el nodo no puede actuar como enrutador.

Distancia entre los nodos Sea R el radio de alcance de cada nodo. Para que la conexión entre 2 nodos exista debe estar a una distancia no mayor a R y para determinar la fuerza de la conexión por distancia debe existir un umbral ρ, es decir, si entre 2 nodos existe una distancia menor a ρ los nodos se encuentran fuertemente conectados, cuando ρ es casi igual a R, los nodos pierden conexión.

Ocupación del canal Si el canal está muy ocupado, no es posible enviar más datos a través de él.

El algoritmo de encaminamiento por consenso hace uso de una función de disponibilidad de enlace. Esta función evalúa factores locales de cada nodo y su enlace para asignar un valor numérico, que indica que tan disponible esta un enlace entre un par de nodos. La función de disponibilidad considera la distancia en términos de número de nodos (saltos) intermedios entre un par de nodos. Esta función se define como sigue:

$$f_{i,j}(l) = e^{\left(\frac{-({s_d}^2+\delta n, {hops}^2+C_{i,j}^2)}{\sigma^2}\right)}. \tag{3.1}$$

Donde s_d es el intervalo de tiempo requerido por el planificador de tareas del nodo, δ_n es la perdida de datos en el canal de comunicación, $hops$ es el alcance de los $l - vecindario$ y C_{ij} es la carga en el canal de comunicación entre el nodo i y el nodo j. s_d y $C_{i,j}$ están dados en términos de μs y kbps respectivamente, para evitar problemas con las escala ambos (μs, y kbps) han sido escalados al orden 10^1 [AV11]. Para representar la topología de la red se utiliza una matriz de adyacencia; esta matriz es una matriz simétrica a la que se nombra A, el elemento $a_{i,j} \in A$ indica si un nodo i está conectado con el nodo j. σ es la desviación estándar de los factores a evaluar. El valor numérico obtenido de la función de disponibilidad indica cuando una conexión es factible. Cada fila y columna de la matriz A representa una conexión entre nodos.

Cada nodo es usado como enrutador, si un nodo no tiene suficiente tiempo disponible en el planificador de tareas entonces no está disponible para proveer el

servicio de retransmisión de mensajes, es así que este servicio tendrá una baja prioridad (para el nodo) porque el nodo debe atender los servicios locales antes de los servicios remotos. Esto incrementa la probabilidad de que el mensaje alcanzará su destino a pesar de los cambios que sufra la topología de la ruta, porque al considerar condiciones locales se construye un paso de la ruta a la vez.

Para evitar errores comunes como son los ciclos de encaminamiento (como en RIP [PD07], que ocurren cuando la información de encaminamiento no se encuentra actualizada), el *algoritmo de encaminamiento por consenso* evalúa las condiciones de la red cuando requiere encontrar una nueva ruta, por lo que las matrices son independientes y no mantienen relación con el estado previo de la red.

Para obtener el estado completo[1] de la red a través de datos locales, se ha implementado una matriz de adyacencia. Como la matriz de adyacencia es una representación matemática de la conectividad de la red, el costo de las rutas en un vecindario acotado puede ser calculado, agregando una restricción en el número de saltos en la Ecuación 3.1. La conectividad es dependiente de la topología, si la topología cambia la conectividad también lo hará. En el proceso de elegir una opción adecuada entre un par de nodos es necesario comparar todas las posibles opciones, para este propósito es necesario calcular el costo de conectar el nodo fuente con un conjunto de nodos con conexión directa (utilizando la función de disponibilidad y un algoritmo de consenso para elegir que nodo será el siguiente paso) hasta alcanzar el nodo destino.

3.1 Algoritmo para caso de estudio de topología “fija”

El algoritmo de encaminamiento por consenso está basado en el algoritmo de votaciones de promedio ponderado [LSBB01]. El algoritmo de encaminamiento por consenso se describe a continuación, para el caso de estudio “ fijo” , es decir, la topología no cambia, solo cambian los valores correspondientes a la carga en el canal de comunicación. La variación de estos valores modificará los valores de las entradas de los algoritmos. Sin embargo las 3 versiones correspondientes a este caso comparten las definiciones.

[1] El conjunto de estados de disponibilidad de tiempo de los planificadores de los nodos de la red

Para modelar un sistema distribuido (SD) compuesto de n nodos, se consideran los siguientes parámetros y variables:

- α es la *matriz de adyacencia* del SD, la cual contiene la información sobre la conectividad del SD.

- β es la *matriz de carga* del SD, la cual contiene la información sobre la carga de cada enlace entre cada par de nodos directamente conectados.

- λ es la *matriz de tiempo disponible* del SD, la cual provee información sobre el tiempo disponible en el planificador de tareas de los nodos, cuando es necesario que actúe como enrutador.

- Considérese n_s el *nodo fuente*, desde el cual el proceso de búsqueda de rutas comienza y sea n_d el *nodo destino*, al cual el proceso de descubriendo de rutas terminará.

- $d_G(i)$ es el *grado del nodo* i, esto es el número de enlaces incidentes al nodo i.

- k es el *número total de pasos de encaminamiento* de una ruta dada.

- l es el *vecindario* con respecto al nodo, cuya extensión es expresada como número de saltos.

- J_i es el conjunto de nodos que conforman el vecindario del nodo $i = n_s$, con respecto a la matriz de adyacencia α, el conjunto J_i puede ser definido como:

$$J_i = \{j | \alpha_{ij} = 1; j = 1, \cdots, n\}; i = 1, \cdots, n. \tag{3.2}$$

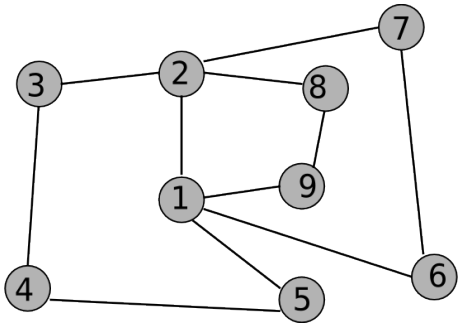


Figura 3.1: Ejemplo de un grafo y los valores correspondientes a la función de disponibilidad. Se considera que el nodo $i = 1$, y $l-$vecindario con $l = 1, J_1 = \{2, 5, 6, 9\}$.

- m es el número de enlaces que el nodo i tiene activos con otros nodos de la red, formalmente $m = |J_i|$, que es la cardinalidad del conjunto J_i.

- $f^k_{i,J_i}(l)$ para $i = 1, \cdots, m$ es la *función que evalúa la disponibilidad de un enlace*. Estos valores son usados como entradas para el algoritmo, tal como se muestra en la Figura 3.1 se muestra un ejemplo de un grafo y los valores de disponibilidad de sus enlaces.

- Sea y^k_i la salida de la operación de consenso.

El *AEC* hace uso de una *función de disponibilidad* ($f^k_{(i,J_i)}(l)$) que permite evaluar las condiciones locales de la conexión entre cada par de nodos involucrados [AVBPOA11]. Los valores resultantes de esta función son usados como entrada para el algoritmo de consenso, el cuál está basado en el algoritmo de votaciones de promedio ponderado descrito en [LSBB01].

La ecuación 3.3 describe tal función de disponibilidad, el cual es calculado por cada nodo en el SD.

$$f^k_{(i,J_i)}(l) = e^{-\left(\frac{(\lambda(i,J^h_i)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(i,J^h_i))^2}{\sigma_3^2}\right)}. \tag{3.3}$$

Donde $h = 1, \cdots, m$.

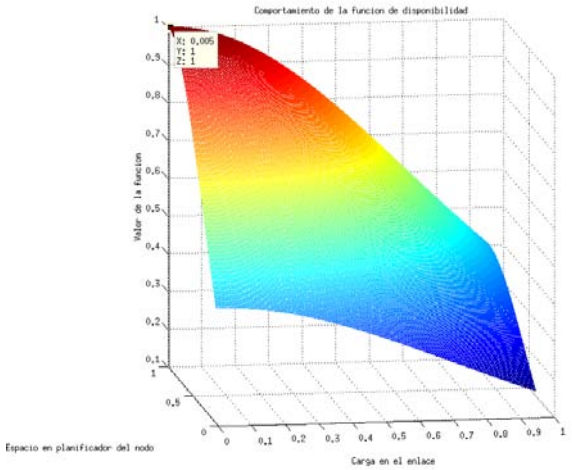


Figura 3.2: Comportamiento de la función 3.3.

Esta función de disponibilidad toma en consideración no solo varios parámetros y variables descritos anteriormente, también toma en cuenta parámetros relevantes del SD, tales como el ancho de banda disponible al instante por enlace (τ), el tiempo disponible en el planificador el nodo (Γ), y el número de saltos entre un par de nodos ($hops$). El valor obtenido por esta función está entre 0 y 1, lo que nos indica que tan disponible está ese enlace para ser parte de la ruta. Nótese que comúnmente $\lambda(J_i)$ está dado en μs, mientras que $\beta(J_i)$ está dado en *kbps*. Sea $\sigma_i|i = 1, 2, 3$ las desviaciones estándar del tiempo disponible en el planificador del nodo, número de saltos y carga en el canal de comunicación, respectivamente. Por lo tanto, para evitar inconsistencias, ambos parámetros han sido escalados a 10^1.

La función de disponibilidad describe el comportamiento mostrado en la figura 3.2, la función tenderá a 1 cuando la carga disminuya y el tiempo disponible en el procesador disminuya.

La complejidad del *AEC* depende del tamaño del vecindario (l) y del grado de los nodos, esto es, el número de nodos con los cuales cada nodo está directamente conectado ($D_G(i)$).

La implementación del *AEC* está basado en un árbol de búsqueda, en donde cada rama es evaluada para escoger el mejor candidato . Cuando el número total de nodos del SD aumenta, el vecindario l de cada nodo tiende a crecer. Entonces

el árbol de búsqueda también crece a profundidad, conforme el número de vecinos incrementa.

Vecindario

Es el conjunto de nodos adyacentes a un nodo. Para el caso de las simulaciones realizadas se utilizaron 3 diferentes vecindarios.

1. Vecindario de un salto. Este vecindario incluye a los vecinos inmediatos del nodo.

2. Vecindario de dos saltos. Este vecindario considera a los vecinos inmediatos de los nodos adyacentes al nodo fuente.

3. Vecindario de uno y dos saltos. Este vecindario es la unión de el vecindario de un salto y de dos saltos.

3.1.1 Versión de un salto

Para este algoritmo existen otras versiones, a continuación se detalla la correspondiente a un vecindario de un salto. Para el caso de estudio de topología “fija” en que solo varían los valores de la matriz de carga.

El *AEC* es desarrollado para un SD, a través de los siguientes pasos:

1. El nodo $i = n_s$ comienza la búsqueda del nodo destino n_d, primero busca entre sus vecinos directos, verificando la matriz α; la información de la carga del enlace (β); y λ, revisando el tiempo disponible en cada planificar de cada nodo vecino. Enseguida, i informa a sus vecinos que se busca al nodo n_d.

   De esta manera, se construye la matriz de adyacencia, que solo corresponde a un nodo y sus vecinos inmediatos. Tomando como ejemplo la Figura 3.1.

   En forma paralela, mientras i realiza el consenso (que se describe en los siguientes pasos) , todos los elementos de J_i buscan a n_d en sus vecinos inmediatos, en caso de no encontrarlo, comienzan el consenso con sus propios vecinos directos.

Usando el ejemplo de la figura 3.1, y considerando que $i = A$, la columna correspondiente para A en la matriz de adyacencia es:

$$[1,2] = 1.$$
$$[1,5] = 1.$$
$$[1,6] = 1.$$
$$[1,9] = 1.$$

Como se muestra en (3.1).

$$\alpha = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Tabla 3.1: Representación matricial del SD. Ejemplo: $i = 1, l = 1; J_1 = \{2, 5, 6, 9\}$

Nótese que mientras i lleva a cabo el consenso (descrito en los siguientes pasos), los nodos incluidos en J_i simultáneamente buscan n_d en sus vecinos inmediatos. Si n_d es encontrado hasta este punto, el encaminamiento se termina. De otra manera, los vecinos comienzan un nuevo consenso con sus vecinos inmediatos, calculando la función de disponibilidad, usando la ecuación 3.3.

2. Calcular las distancias numéricas de los $\frac{m(m-1)}{2}$ pares de entradas de votantes . Las distancias numéricas se calculan usando la ecuación 3.4.

$$d_{(J_i^h, J_i^{h+1})} = |f^k_{(i,J_i^h)}(1) - f^k_{(i,J_i^{h+1})}(1)|, \quad h = 1, \cdots, m. \tag{3.4}$$

3. Determinar el nivel de acuerdo de los $\frac{m(m-1)}{2}$ pares de entradas de votantes, basándose en la ecuación (3.5).

$$s_{(J_i^h, J_i^{h+1})} = \frac{1}{1 + pd^q_{(J_i^h, J_i^{h+1})}}, \quad h = 1, \cdots, m. \tag{3.5}$$

   Los parámetros p y q son constantes. El parámetro q ajusta el rango de extensión de esta función y p es usado para establecer el valor del punto medio de la función.

   Para el *algoritmo de encaminamiento por consenso* la distancia numérica es la comparación de la disponibilidad de cada enlace de datos entre los nodos vecinos.

4. Como el valor del nivel de acuerdo ha sido calculado con respecto a los pares de entradas, se asigna un valor de peso para los pares de entradas (iJ_i) votante basándose en la ecuación 3.6

$$w_{(i,J_i^\omega)} = \frac{1}{1 + \prod_{\substack{h \neq \omega, \\ h=1}}^{m} s_{(J_i^\omega, J_i^h)}}, \quad 0 < w_{(i,J_i^\omega)} < 1, \omega = 1, \cdots, m. \tag{3.6}$$

5. Se calcula la salida del consenso (ecuación 3.7)

$$y_i^k = \frac{\sum_{h=1}^{m} w_{(i,J_i^h)} \quad f^k_{(i,J_i^h)}(1)}{\sum_{h=1}^{m} w_{(i,J_i^h)}}. \tag{3.7}$$

6. La variable y_i^k es un valor nuevo (cercano a los valores de las entradas), resultado de la ecuación 3.7, entonces este valor no está en el conjunto de entradas.

   El valor de y_i^k es comparado con los valores de las entradas, de esta manera el nodo J_i^h correspondiente a la entrada asociada a la ecuación 3.8

$$\text{mín}(|y_i^k - f^k_{(i,J_i^h)}(1)|) = g^k, \quad h = 1, \cdots, m. \tag{3.8}$$

   g^k es escogido como el ganador del consenso, será el nodo fuente (i) para la siguiente iteración.

Todos los elementos del conjunto J_i han calculado su propia $y^k_{J^h_i}$ con respecto a si mismos y a sus vecinos inmediatos (su propio conjunto J_i).

Los nodos que participaron en el consenso anterior tendrán precalculada su propia $y^k_{J^h_i}$ que podrán aportar en otra ronda de consenso en un paso k posterior.

7. Finalmente, después de varios procesos de descubrimiento de ruta, algún nodo tiene al n_d en su conjunto J_i, entonces, una ruta ha sido encontrada entre n_s y n_d. Esta ruta esta conformada por todos los ganadores de cada consenso calculado. Por lo tanto, una ruta de k pasos puede ser construida por estos nodos, representados por la secuencia mostrada en 3.9:

$$n_s \rightarrow g^1 \rightarrow g^2 \rightarrow \cdots \rightarrow g^k \rightarrow n_d. \tag{3.9}$$

Cada g^k es un mínimo local, que se mantendrá si los valores de β y λ de los nodos no cambian significativamente. La elección de un mínimo local provee el nodo más conveniente para transmitir a través de nodos alcanzables en k pasos.

3.1.2 Versión de dos saltos

Esta versión se diseñó con el propósito de cuantificar cuando usar el vecindario $l = 1$ y cuando es mejor ampliarlo. En esta versión no se toman en cuanta a los vecinos inmediatos, solo se usan como puente para lograr un mayor alcance, por lo que se denotara como $l = 2'$

1. Sea l la extensión (medida en saltos) del vecindario con respecto a un nodo. Sean los valores de la función de disponibilidad de los enlaces, calculada por $f^k_{(J^h_i, J^u_{J^h_i})}(2')$ para $h = 1, \cdots, m$.

2. El nodo $i = n_s$ comienza la búsqueda del nodo destino n_d, primero busca entre sus vecinos directos, verificando la matriz α.

   En forma paralela, mientras i realiza el consenso (que se describe en los siguientes pasos) , todos los elementos de J^h_i buscan a n_d en sus vecinos

inmediatos, en caso de no encontrarlo comienzan el consenso con sus propios vecinos directos.

Cuando se tiene un vecindario $l = 2'$, usando 3.2 se tiene que $J_{J_i^h}$ son los nodos vecinos de J_i^h que a su vez es el i-ésimo vecino de i. Para indicar un elemento del conjunto se utilizará la notación $J^u_{J_i^h}$, donde $h = 1, \cdots, m$ y $u = 1, \cdots, m_h$, donde $m_h = |J_{J_i^h}|$.

Las entradas del algoritmo, como se muestra en la Tabla 3.2.

$$\alpha = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left(\begin{array}{ccccccccc} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & \mathbf{0} & 1 & 0 & \mathbf{0} & \mathbf{0} & 1 & 1 & \mathbf{0} \\ 0 & \mathbf{1} & 0 & 1 & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} \\ 0 & \mathbf{0} & 1 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{0} \\ 0 & \mathbf{0} & 0 & 1 & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} \\ 1 & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & 1 & 0 & \mathbf{0} \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{0} & \mathbf{1} & 0 & 0 & \mathbf{0} \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{1} \\ 1 & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & 0 & 1 & \mathbf{0} \end{array}\right) \end{array}$$

Tabla 3.2: Ejemplo: $i = 1, l = 2', J_1 = \{2,5,6,9\}$, $J_{J_1^1} = \{3,7,8\}$, $J_{J_1^2} = \{4\}$, $J_{J_1^3} = \{7\}$, $J_{J_1^4} = \{8\}$. Las entradas serían $f^k_{(J_1^h, J^u_{J_A^h})}(2') = \{f^k_{(2,3)}(1), f^k_{(2,7)}(1), f^k_{(2,8)}(1), f^k_{(5,4)}(1), f^k_{(6,7)}(1), f^k_{(9,8)}(1)\}$ donde $h = 1, \cdots, m; u = 1, \cdots, m_h$.

Si no se encuentra el nodo destino n_d en ese conjunto (o conjuntos) entonces se calcula la función de disponibilidad de los vecinos de J_i^h con la ecuación 3.3 .

3. Sea $v = |J_{J_i^1}| + \cdots + |J_{J_i^m}|$.

4. Calcular las distancias numéricas de los $\frac{v(v-1)}{2}$ pares de entradas de votantes . Las distancias numéricas se calculan usando la ecuación 3.10.

$$d_{(J^u_{J^h_i}, J^{u+1}_{J^h_i})} = |f^k_{(J^h_i, J^u_{J^h_i})}(2') - f^k_{(J^h_i, J^{u+1}_{J^h_i})}(2')|, \quad u = 1, \cdots, m_h; h = 1, \cdots, m. \tag{3.10}$$

5. Determinar el nivel de acuerdo de los $\frac{v(v-1)}{2}$ pares de entradas de votantes, basándose en la ecuación (3.11).

$$s_{(J^u_{J^h_i}, J^{u+1}_{J^h_i})} = \frac{1}{1 + pd^q_{(J^u_{J^h_i}, J^{u+1}_{J^h_i})}}, \quad u = 1, \cdots, m_h; h = 1, \cdots, m. \tag{3.11}$$

   Los parámetros p y q son constantes. El parámetro q ajusta el rango de extensión de esta función y p es usado para establecer el valor del punto medio de la función.

   Para el *algoritmo de encaminamiento por consenso* la distancia numérica es la comparación de la disponibilidad de cada enlace de datos entre los nodos vecinos.

6. Como el valor del nivel de acuerdo ha sido calculado con respecto a los pares de entradas, se asigna un valor de peso para los pares de entradas (iJ_i) votante basándose en la ecuación 3.12

$$w_{(J^h_i, J^\omega_{J^h_i})} = \frac{1}{1 + \prod^v_{\substack{\omega \neq u, \\ u=1}} s_{(J^\omega_{J^h_i}, J^u_{J^h_i})}}, \quad h = 1, \cdots, m; 0 < w_{(J^h_i, J^\omega_{J^h_i})} < 1. \tag{3.12}$$

7. Se calcula la salida del consenso (ecuación 3.13)

$$y^k_i = \frac{\sum^m_{h=1} w_{(J^h_i, J^u_{J^h_i})} \quad f^k_{(J^h_i, J^u_{J^h_i})}(2')}{\sum^m_{h=1} w_{(J^h_i, J^u_{J^h_i})}}, \quad u = 1 \cdots, m_h. \tag{3.13}$$

8. La variable y^k_i es un valor nuevo (cercano a los valores de las entradas), resultado de la ecuación 3.7, entonces este valor no está en el conjunto de entradas.

El valor de y_i^k es comparado con los valores de las entradas, de esta manera el nodo J_i^h correspondiente a la entrada asociada a la ecuación 3.14.

$$\text{mín}(|y_i^k - f^k_{(J_i^h, J^u_{J_i^h})}(2')|) = g^k, \quad h = 1, \cdots, m; u = 1 \cdots, m_h. \quad (3.14)$$

g^k es escogido como el ganador del consenso, será el nodo fuente (i) para la siguiente iteración.

Todos los elementos del conjunto J_i han calculado su propia $y^k_{J_i^h}$ con respecto a si mismos y a sus vecinos inmediatos (su propio conjunto J_i).

Los nodos que participaron en el consenso anterior tendrán precalculada su propia $y^k_{J_i^h}$ que podrán aportar en otra ronda de consenso en un paso k posterior.

9. Finalmente, después de varios procesos de descubriendo de ruta, algún nodo tiene al n_d en su conjunto J_i, entonces, una ruta ha sido encontrada entre n_s y n_d. Esta ruta esta conformada por todos los ganadores de cada consenso calculado. Por lo tanto, una ruta de k pasos puede ser construida por estos nodos, representados por la siguiente secuencia:

$$n_s \to g^1 \to g^2 \to \cdots \to g^k \to n_d. \quad (3.15)$$

3.1.3 Versión de uno y dos saltos

Esta versión incluye un vecindario extendido de 2 saltos, es decir el vecindario está compuesto de los vecinos directos y de los vecinos de estos, se esta manera poder ampliar el área de búsqueda.

1. Sea l la extensión (medida en saltos) del vecindario con respecto a un nodo. Sean los valores de la función de disponibilidad de los enlaces, calculada por 3.16.

$$f^k_{(i, J^u_{J_i^h})}(2) = f^k_{(i, J_i^h)}(1) + f^k_{(J_i^h, J^u_{J_i^h})}(1); \quad h = 1, \cdots, m; u = 1, \cdots, m_h. \quad (3.16)$$

2. El nodo n_s comienza la búsqueda del nodo destino n_d, primero busca entre sus vecinos directos, verificando la matriz α.

   En forma paralela, mientras i realiza el consenso (que se describe en los siguientes pasos) , todos los elementos de J_i^h buscan a n_d en sus vecinos inmediatos, en caso de no encontrarlo comienzan el consenso con sus propios vecinos directos.

   Cuando se tiene un vecindario $l = 2$, usando 3.2 se tiene que $J_{J_i^h}$ son los nodos vecinos de J_i^h que a su vez es el i-ésimo vecino de i. Para indicar un elemento del conjunto se utilizará la notación $J^u_{J_i^h}$, donde $h = 1, \cdots, m$ y $u = 1, \cdots, m_h$, donde $m_h = |J_{J_i^h}|$.

   Las entradas del algoritmo, como se muestra en la Tabla 3.3.

$$\alpha = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left(\begin{array}{ccccccccc} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ 1 & \mathbf{0} & 1 & 0 & \mathbf{0} & \mathbf{0} & 1 & 1 & \mathbf{0} \\ 0 & \mathbf{1} & 0 & 1 & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} \\ 0 & \mathbf{0} & 1 & 0 & \mathbf{1} & \mathbf{0} & 0 & 0 & \mathbf{0} \\ 0 & \mathbf{0} & 0 & 1 & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} \\ 1 & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & 1 & 0 & \mathbf{0} \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{0} & \mathbf{1} & 0 & 0 & \mathbf{0} \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{1} \\ 1 & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & 0 & 1 & \mathbf{0} \end{array} \right) \end{array}$$

Tabla 3.3: Representación matricial del SD. Ejemplo: $i = 1, l = 2, J_1 = \{2,5,6,9\}$, $J_{J_1^1} = \{3,7,8\}$, $J_{J_1^2} = \{4\}$, $J_{J_1^3} = \{7\}$, $J_{J_1^4} = \{8\}$. Las entradas serían $f_{(1,J^u_{J_1^h})}(2) = \{f^k_{(1,2)}(1) + f^k_{(2,3)}(1), f^k_{(1,2)}(1) + f^k_{(2,7)}(1), f^k_{(1,2)}(1) + f^k_{(2,8)}(1), f^k_{(1,5)}(1) + f^k_{(5,4)}(1), f^k_{(1,6)}(1) + f^k_{(6,7)}(1), f^k_{(1,9)}(1) + f^k_{(9,8)}(1)\}$ donde $h = 1, \cdots, m; u = 1, \cdots, m_h$.

   Si no se encuentra el nodo destino n_d en ese conjunto (o conjuntos) entonces se calcula la función de disponibilidad de los vecinos de J_i^h con la ecuación 3.3 .

3. Sea $v = |J_{J_i^1}| + \cdots + |J_{J_i^m}|$.

4. Calcular las distancias numéricas de los $\frac{v(v-1)}{2}$ pares de entradas de votantes . Las distancias numéricas se calculan usando la ecuación 3.17.

$$d_{(J^u_{J^h_i}, J^{u+1}_{J^h_i})} = |f^k_{(i,J^u_{J^h_i})}(2) - f^k_{(i,J^{u+1}_{J^h_i})}(2)|, \quad u = 1, \cdots, m_h; h = 1, \cdots, m. \tag{3.17}$$

5. Determinar el nivel de acuerdo de los $\frac{v(v-1)}{2}$ pares de entradas de votantes, basándose en la ecuación (3.18).

$$s_{(J^u_{J^h_i}, J^{u+1}_{J^h_i})} = \frac{1}{1 + pd^q_{(J^u_{J^h_i}, J^{u+1}_{J^h_i})}}, \quad u = 1, \cdots, m_h; h = 1, \cdots, m. \tag{3.18}$$

   Los parámetros p y q son constantes. El parámetro q ajusta el rango de extensión de esta función y p es usado para establecer el valor del punto medio de la función.

   Para el *algoritmo de encaminamiento por consenso* la distancia numérica es la comparación de la disponibilidad de cada enlace de datos entre los nodos vecinos.

6. Como el valor del nivel de acuerdo ha sido calculado con respecto a los pares de entradas, se asigna un valor de peso para los pares de entradas (iJ_i) votante basándose en la ecuación 3.19

$$w_{(i,J^\omega_{J^h_i})} = \frac{1}{1 + \prod^v_{\omega \neq u,\ u=1} s_{(J^\omega_{J^h_i}, J^u_{J^h_i})}}, \quad h = 1, \cdots, m; 0 < w_{(i,J^\omega_{J^h_i})} < 1. \tag{3.19}$$

7. Se calcula la salida del consenso (ecuación 3.20)

$$y^k_i = \frac{\sum^m_{h=1} w_{(i,J^u_{J^h_i})} \quad f^k_{(i,J^u_{J^h_i})}(2)}{\sum^m_{h=1} w_{(i,J^u_{J^h_i})}}, \quad u = 1 \cdots, m_h. \tag{3.20}$$

8. La variable y^k_i es un valor nuevo (cercano a los valores de las entradas), resultado de la ecuación 3.7, entonces este valor no está en el conjunto de entradas.

El valor de y_i^k es comparado con los valores de las entradas, de esta manera el nodo J_i^h correspondiente a la entrada asociada a la ecuación 3.21.

$$\text{mín}(|y_i^k - f^k_{(J_i^h, J^u_{J_i^h})}(2)|) = g^k, \quad h = 1, \cdots, m; u = 1 \cdots, m_h. \quad (3.21)$$

g^k es escogido como el ganador del consenso, será el nodo fuente (i) para la siguiente iteración.

Todos los elementos del conjunto J_i han calculado su propia $y^k_{J_i^h}$ con respecto a si mismos y a sus vecinos inmediatos (su propio conjunto J_i).

Los nodos que participaron en el consenso anterior tendrán precalculada su propia $y^k_{J_i^h}$ que podrán aportar en otra ronda de consenso en un paso k posterior.

9. Finalmente, despues de varios procesos de descubriento de ruta, algún nodo tiene al n_d en su conjunto J_i, entonces, una ruta ha sido encontrada entre n_s y n_d. Esta ruta esta conformada por todos los ganadores de cada consenso calculado. Por lo tanto, una ruta de k pasos puede ser construida por estos nodos, representados por la siguiente secuencia:

$$n_s \rightarrow g^1 \rightarrow g^2 \rightarrow \cdots \rightarrow g^k \rightarrow n_d. \quad (3.22)$$

3.2 Algoritmo para caso de estudio de topología móvil

A lo largo de este trabajo se asumen las siguientes condiciones:

- No todos los nodos dentro del SDM son aptos para funcionar como router, eso se determinará por medio de la función de disponibilidad 3.1.
- Los nodos del SDM se pueden mover en cualquier momento, en cualquier dirección, incluso de manera vertical u horizontal en el mismo tiempo.
- La velocidad con la que se mueven es moderada con respecto a la latencia de la transmisión de mensajes y al alcance de la transmisión de los nodos pertenecientes a la red.

- Se supone que los nodos no se mueven tan rápido como para que el único mecanismo de encaminamiento factible sea la inundación de la red.

En este caso de estudio se agrega otro factor a considerar, que es la movilidad de los nodos. Un nodo puede ser o no móvil, para esto cada nodo en su estructura contiene una bandera que indica si puede moverse. Un nodo que es móvil, puede permanecer en una sola posición por algún intervalo de tiempo, el movimiento puede ser vertical y/o horizontal. La velocidad del movimiento es independiente del proceso de consenso.

En esta versión, el sistema distribuido móvil (SDM en adelante) con n nodos, utiliza una estructura de datos por cada nodo, la cual contiene:

1. Identificador del nodo (ID).
2. La ubicación espacial en la malla (coordenadas x,y).
3. Tiempo disponible en su planificador de tareas (tiempo).
4. Lista de sus vecinos (conjunto J_i) en un rango de transmisión ρ.
5. Carga entre cada uno de sus enlaces con sus vecinos (carga).
6. Una bandera que indica si el nodo es móvil (flag).

Habiendo definido la estructura de cada nodo, el algoritmo de encaminamiento por consenso se describe a continuación:

1. Sea J_i es el conjunto de nodos pertenecientes al vecindario del nodo i, en un vecindario radial ρ (figura 3.3), el conjunto J_i se define como (3.23).

$$J_i = \{j : |x_i - x_j| \wedge |y_i - y_j| \leq \rho; j = 1, \cdots, n\}; i = 1, \cdots, n. \quad (3.23)$$

   Sean los valores de la función de disponibilidad de los enlaces, calculada por $f^k_{(i,J_i)}(\rho)|i = 1, \cdots, m$ las entradas del algoritmo.

   Sea y^k_i la salida del consenso.

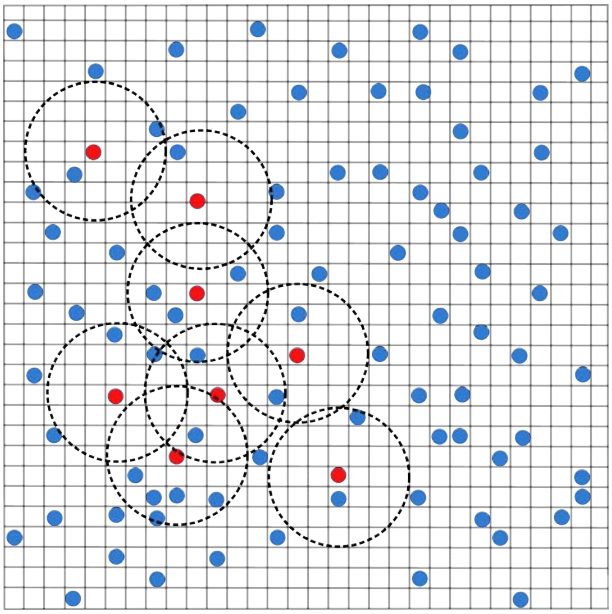


Figura 3.3: Ejemplo de SDM.

2. El nodo $i = n_s$ comienza la búsqueda del nodo destino n_d, primero busca entre sus vecinos directos.

   En forma paralela, mientras i realiza el consenso (que se describe en los siguientes pasos) , todos los elementos de J_i buscan a n_d en sus vecinos inmediatos, en caso de no encontrarlo comienzan el consenso con sus propios vecinos directos.

   Si no se encuentra el nodo destino n_d en ese conjunto, entonces se calcula la función de disponibilidad de los vecinos de i con la ecuación 3.24.

$$f^k_{(i,J_i)}(\rho) = e^{-\left(\frac{(tiempo(i,J_i^h)-\Gamma_{J_i^h})^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau_{J_i^h}-carga(i,J_i^h))^2}{\sigma_3^2}\right)} \quad h = 1, \cdots, m. \tag{3.24}$$

   Sea τ la capacidad del canal de comunicación entre el nodo i y el nodo J_i^h y Γ el tiempo necesario para atender el servicio de transmisión de datos.

   Como $tiempo(i, J_i^h)$ y $carga(i, J_i^h)$ están dados en términos de μs y $kbps$ respectivamente, para evitar problemas con las escala ambos (μs y kbps) han sido escalados al orden 10^1.

3. Calcular las distancias numéricas de los $\frac{m(m-1)}{2}$ pares de entradas de votantes . Las distancias numéricas se calculan usando la ecuación 3.25.

$$d_{(J_i^h, J_i^{h+1})} = |f^k_{(i,J_i^h)}(\rho) - f^k_{(i,J_i^{h+1})}(\rho)|, \quad h = 1, \cdots, m. \tag{3.25}$$

4. Determinar el nivel de acuerdo de los $\frac{m(m-1)}{2}$ pares de entradas de votantes, basándose en la ecuación (3.26).

$$s_{(J_i^h, J_i^{h+1})} = \frac{1}{1 + pd^q_{(J_i^h, J_i^{h+1})}}, \quad h = 1, \cdots, m. \tag{3.26}$$

   Los parámetros p y q son constantes. El parámetro q ajusta el rango de extensión de esta función y p es usado para establecer el valor del punto medio de la función.

   Para el *algoritmo de encaminamiento por consenso* la distancia numérica es la comparación de la disponibilidad de cada enlace de datos entre los nodos vecinos.

5. Como el valor del nivel de acuerdo ha sido calculado con respecto a los pares de entradas, se asigna un valor de peso para los pares de entradas (iJ_i) votante basándose en la ecuación 3.27

$$w_{(i,J_i^\omega)} = \frac{1}{1 + \prod_{h=1, h\neq\omega}^{m} s_{(J_i^\omega, J_i^h)}}, \quad 0 < w_{(i,J_i^\omega)} < 1, \omega = 1, \cdots, m. \tag{3.27}$$

6. Se calcula la salida del consenso (ecuación 3.28)

$$y_i^k = \frac{\sum_{h=1}^{m} w_{(i,J_i^h)} \quad f^k_{(i,J_i^h)}(\rho)}{\sum_{h=1}^{m} w_{(i,J_i^h)}}. \tag{3.28}$$

7. La variable y_i^k es un valor nuevo (cercano a los valores de las entradas), resultado de la ecuación 3.28, entonces este valor no está en el conjunto de entradas.

   El valor de y_i^k es comparado con los valores de las entradas, de esta manera el nodo J_i^h correspondiente a la entrada asociada a 3.29.

$$\text{mín}(|y_i^k - f^k_{(i,J_i^h)}(\rho)|) = g^k, \quad h = 1, \cdots, m. \tag{3.29}$$

g^k es escogido como el ganador del consenso, será el nodo fuente (i) para la siguiente iteración.

Todos los elementos del conjunto J_i han calculado su propia $y^k_{J_i^h}$ con respecto a si mismos y a sus vecinos inmediatos (su propio conjunto J_i).

Los nodos que participaron en el consenso anterior tendrán precalculada su propia $y^k_{J_i^h}$ que podrán aportar en otra ronda de consenso en un paso k posterior.

8. Finalmente, después de varios procesos de descubrimiento de ruta, algún nodo tiene al n_d en su conjunto J_i, entonces, una ruta ha sido encontrada entre n_s y n_d. Esta ruta esta conformada por todos los ganadores de cada consenso calculado. Por lo tanto, una ruta de k pasos puede ser construida por estos nodos, representados por la siguiente secuencia:

$$n_s \rightarrow g^1 \rightarrow g^2 \rightarrow \cdots \rightarrow g^k \rightarrow n_d. \tag{3.30}$$

La ruta no se mantiene, por las condiciones de movilidad. En caso de que se presente un ciclo de encaminamiento, este ciclo puede ser roto por un nodo móvil que cumpla con las mejores condiciones para trasmitir y de esta manera alcanzar el nodo destino.

3.3 Resumen

Se desarrollaron 2 casos generales de estudios para el diseño de *AEC*. El primero que es para una topología fija y otro para topología móvil. Los escenarios considerados para la versión de topología fija se limitan a un vecindario de 1 saltos, 2 saltos (sin considerar el vecindario inmediato) y 1 y 2 saltos (la combinación de ambos vecindarios). La versión móvil es muy parecida a la versión de un salto para la topología fija, lo que cambia en la versión móvil es la forma de determinar el vecindario y se agregaron las características de los nodos móviles.

Hay una diferencia entre conocer el camino y recorrerlo.

Morpheo. The Matrix.

CAPÍTULO

4

Experimentación

En este capítulo se desarrollan las simulaciones y experimentos que corresponden a probar el funcionamiento del algoritmo propuesto; incluyendo las versiones para topología fija y móvil, así como comparativas con el algoritmo de inundación y el algoritmo de rutas más cortas propuesto por Dijkstra.

En el caso del análisis de diseño de la simulación para las versiones con topología fija se pretende mostrar el porcentaje de éxitos (destinos alcanzados) que se tiene al cambiar el alcance del vecindario y variando los valores de carga en algunas secciones de la matriz original. Para la versión móvil se pretende estudiar la influencia de los nodos móviles en el SDM para determinar que tanto afectan para la obtención de rutas exitosas.

En el caso de las comparativas se pretende evaluar la calidad de las rutas obtenidas por *AEC* y por los otros algoritmos al incluir la movilidad de los nodos.

En el caso de los experimentos se busca estudiar el funcionamiento del algoritmo en el hardware de Arduino ® y probarlo con los valores reales de tiempo de comunicación entre dispositivos, sin las restricciones de Matlab para las simulaciones.

4.1 Descripción de las simulaciones

La ejecución de las simulaciones incluye los siguientes procesos:

- Generación de archivos de entrada: para la generación de los archivos correspondientes a las matrices fijas con las que se realizaran los experimentos, se requieren 3 matrices de las mismas dimensiones:

    1. Adyacencia: Los valores de esta matriz son binarios y la matriz es dispersa. Usando las posiciones de esta matriz con valor=1 se generan los valores de las otras 2 matrices.
    2. Carga: Los valores de carga se generan usando la función de distribución de probabilidad χ^2
    3. Espacio disponible en planificador del nodo: Estos valores son números enteros en el rango $(1, 20)$ generaron de manera aleatoria utilizado un distribución normal.

- Generación de archivos de salida: durante cada simulación se generan 2 tipos de archivos únicos; el que indica en los pasos intermedios de la ruta. Cuando el destino es alcanzado el tiempo total de la simulación es escrito en el archivo de salida donde se guardan los pasos intermedios. El otro archivo generado indica el rango de posiciones en la matriz donde los valores de carga han variado en cada llamada al algoritmo.

- Ejecución de los programas: Fue necesario implementar los algoritmos descritos en el capitulo anterior. Los diagramas de flujo que indican como están conformados los códigos se detallan en el apéndice 1.

- Procesamiento de resultados: Fue necesario desarrollar una serie de *scripts* escritos en Python para el filtrado y análisis de resultados. Se utilizaron las bibliotecas NumPy y Netwokx.

Existen diferentes tipos de topologías que fueron consideradas para generar los archivos de entrada para las simulaciones:

Redes aleatorias fijas

Son aquellas donde la disposición de los nodos en el plano sigue un patrón aleatorio, en la cual aquellos nodos que están dentro del radio de cobertura de otro nodo pueden establecer un enlace físico entre ellos, luego según las reglas del algoritmo de encaminamiento podrán convertirse en enlaces lógicos y ser utilizados para enviar o recibir paquetes. En esta topología la secuencia en la que se conectan los nodos y se convierten en parte de la red es completamente aleatoria. Los nodos durante toda la simulación no cambiaran de posición, pero sus valores carga estarán variando continuamente en algunas regiones localizadas de la red, lo que hará que no siempre sean buenos candidatos a establecer un enlace lógico.

Redes aleatorias móviles

Son aquellas donde la disposición de los nodos en el plano sigue un patrón aleatorio, en la cual aquellos nodos que están dentro del radio de cobertura de otro nodo pueden establecer un enlace físico entre ellos, luego según las reglas del algoritmo de encaminamiento podrán convertirse en enlaces lógicos y ser utilizados para enviar o recibir paquetes. En esta topología la secuencia en la que se conectan los nodos y se convierten en parte de la red es completamente aleatoria. Cada nodo contiene información sobre su posición física en el plano. Algunos nodos dentro de la red tienen la propiedad de ser móviles, estos se desplazaran a lo largo de la malla, influyendo en el cálculo del consenso del *AEC*.

4.2 Especificaciones de las simulaciones para la versión de topología fija con variación de carga

Se realizaron simulaciones, en Matlab ®, utilizando matrices de distintos tamaños, las matrices son fijas para todos las simulaciones y varían un porcentaje en los valores de carga a lo largo de la simulación. El porcentaje de variación se indica en las Tablas, así como el número de nodos de cada simulación. El objetivo de estás simulaciones fue saber el porcentaje de éxito del *AEC*.

Para el caso de un salto para la versión fija del algoritmo, las simulaciones tienen el objetivo de identificar el alcance de la propagación de la búsqueda del *AEC*.

Número de nodos	**Número de simulaciones**	**% de cambio en la matriz de carga**	**Tiempo máximo**
100	100	20 %	300 s
200	100	20 %	300 s
500	100	20 %	300 s

Tabla 4.1: Especificaciones de la versión de un salto.

Para las subversiones de un salto y dos saltos se utilizaron matrices de 100, 200 y 500 nodos. En las Tablas 4.1 y 4.2 se muestran las especificaciones de cada conjunto de simulaciones realizadas. En estas Tablas de muestran el número se simulaciones realizadas en topologías de distintos tamaños y el tiempo máximo en segundos que se dejo simular, así como el porcentaje de variación de en la matriz de carga. Para el caso del vecindario de un salto el tiempo máximo por simulación fue de 300 segundos.

En las simulaciones realizadas para el caso de un vecindario de dos saltos se permitió un tiempo máximo de 600 segundos para cada simulación. Para la versión

Número de nodos	**Número de simulaciones**	**% de cambio en la matriz de carga**	**Tiempo máximo**
100	100	20 %	600 s
200	100	20 %	600 s
500	100	20 %	600 s

Tabla 4.2: Especificaciones de la versión de dos saltos.

de uno y dos saltos, se utilizaron matrices más grandes que en las simulaciones anteriores. Las matrices utilizadas fueron de 200, 500 y 1000 nodos. La especificación de la simulación se detalla en la Tabla 4.3.

Número de nodos	Número de simulaciones	% de cambio en la matriz de carga	Tiempo máximo
200	20	20 %	3600 s
500	20	20 %	3600 s
1000	20	20 %	3600 s

Tabla 4.3: Especificaciones de la versión de uno y dos saltos.

4.3 Especificaciones de las simulaciones para la versión de topología móvil

Se realizaron simulaciones en Matlab de la versión móvil de la simulación. Cada nodo esta representado por una estructura de datos compuesta por:

- Identificador el nodo (ID).
- Posición en el eje X.
- Posición en el eje Y.
- Tiempo disponible en su planificador de tareas (tiempo).
- Carga entre el nodo que solicita y el nodo.
- Una bandera que indica si el nodo es móvil (flag)

En estas simulaciones se variaba el porcentaje de nodos móviles del número total de nodos del SDM y se probó para SDM de 30,150,200,500 y 1000 nodos. En la Tabla 4.4 se muestran los detalles de la simulación.

4.4 Especificaciones de las simulaciones de influencia de la movilidad en el éxito del algoritmo

En base a la simulación de la versión para topología móvil, se realizó otros conjuntos de simulaciones para medir la influencia de la movilidad de los nodos en el éxito de algoritmo. El objetivo de este conjunto de simulaciones es determinar que

Número de nodos	% de nodos móviles	Número de simulaciones
30	5	200
150	5	200
200	5	200
500	5	200
1000	5	200
30	10	200
150	10	200
200	10	200
500	10	200
1000	10	200

Tabla 4.4: Especificaciones de las simulaciones de la versión móvil.

porcentaje de los éxitos obtenidos en cada conjunto eran logrados por causa de los nodos móviles. En la Tabla 4.5 se muestra el número de nodos que tiene cada topología móvil, en este caso se incrementó la velocidad a la que se mueven los nodos en las últimas 2 simulaciones, mientras se calcula el nodo ganador del consenso los nodos móviles ya se han movido 4 pasos de manera independiente.

	Número de nodos	Pasos por búsqueda	% de nodos móviles	Tiempo de búsqueda
Simulación 1	100	1	10	1000 s
Simulación 2	150	1	10	1200 s
Simulación 3	100	4	10	1000 s
Simulación 4	150	4	10	1200 s

Tabla 4.5: Especificaciones de las simulaciones de influencia de la movilidad.

4.5 Especificaciones de las comparativas

Con el fin de evaluar funcionalidad del *AEC* se realizaron una serie de comparativas de rutas. Para la versión de topología fija del algoritmo se comparó con el algoritmo de Dijkstra y el algoritmo de inundación, porque estos son los más usados en

los protocolos de encaminamiento convencionales. Para la versión móvil solo de comparó con el algoritmo de inundación.

4.5.1 Especificaciones de la comparativa *AEC* vs Algoritmo de Dijkstra

Se hizo un comparativo de la versión de 1 y 2 saltos contra el algoritmo de Dijkstra, que es uno de los más usados en los algoritmos de encaminamiento convencionales. También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista
La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.

Para esta comparativa se utilizó una topología fija de 50 nodos con variación de carga. Se ejecutó el algoritmo de Dijkstra con los valores iniciales de carga, como se muestra en la Tabla 4.6. El algoritmo de Dijkstra considera la distancia física para determinar la ruta; *AEC* considera la carga y el espacio en el planificador del nodo para elegir una ruta.

	Número de nodos	**% de cambio en la matriz de carga**
AEC	50	20
Dikjstra	50	0

Tabla 4.6: Especificaciones de la comparativa *AEC* vs Algoritmo de Dijkstra.

4.5.2 Especificaciones de la comparativa *AEC* vs Inundación

Al implementar el algoritmo de inundación para compararlo con la versión de 1 y 2 saltos, se limitó a cuatro saltos. Se utilizó una matriz de adyacencia de 500 nodos, la matriz de carga y de tiempo disponible en el planificador del nodo tienen las mismas dimensiones que la de adyacencia. Las especificaciones se muestran en la

Tabla 4.7.

La matriz de adyacencia se obtiene de distribuir los nodos con una distribución normal en una malla de 500×500, cada nodo considera como vecinos a 25 unidades a la redonda, se utilizo el nodo origen y nodo destino con los mismos valores.

Número de nodos	**Número de simulaciones**
200	20
500	20
1000	20

Tabla 4.7: Especificaciones de la comparativa de inundación.

4.6 Especificaciones técnicas de la experimentación en hardware

El objetivo de este experimento es evaluar el desempeño del *AEC* en un ambiente de hardware real. De modo que con un conjunto de dispositivos se pueda establecer un nodo fuente (n_s) y fijando un nodo destino (n_d). n_s deberá comunicarse con sus vecinos y calcular una ruta hasta llegar al nodo destino. En la Figura 4.1 se muestra la nomenclatura de los diagramas UML que se utilizaran a lo largo de este capitulo.

Con la finalidad de probar el funcionamiento del algoritmo en un sistema real, *AEC* fue implementado en Python y en lenguaje C, para su uso en el hardware PCduino®, Arduino Yun (placa Arduino con Ethernet y WiFi integrado) y Arduino® respectivamente.

Para describir el comportamiento del sistema diseñado se utilizaran diagramas de caso de uso y de actividad, correspondientes al Lenguaje de Modelado Unificado. El diagrama de caso de uso. Los casos de uso están representados por elipses y los actores están, por ejemplo, los casos de uso se muestran como parte del sistema que está siendo modelado, los actores no.

Figura 4.1: Nomenclatura de UML.

Los Diagramas de Actividad ofrecen una herramienta gráfica para modelar el proceso de un caso de uso. Se pueden usar como un añadido a una descripción textual del caso de uso, o para listar los pasos del caso de uso.

El conjunto de módulos que conforman la implementación de *AEC* están basados en el caso de uso descrito en la Figura 4.2.

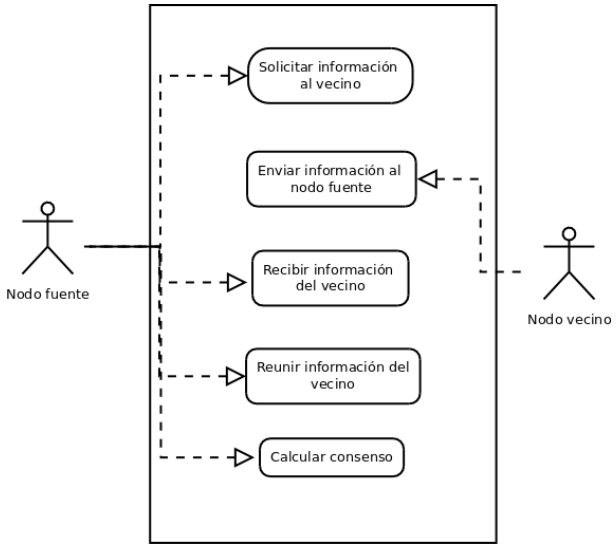


Figura 4.2: Caso de Uso.

Se plantea que cada nodo sea capaz de desempeñar 2 roles, dependiendo de la decisión que tome el algoritmo de consenso. En el caso de iniciar la búsqueda de la ruta, el nodo será **Nodo Fuente** (n_S) y realizará las funciones de solicitar información a sus vecinos, recibir información de sus vecinos, reunir la información de los vecinos y calcular el consenso. Posterior al proceso de consenso, se decide cual de los vecinos será el siguiente **Nodo Fuente**, el nodo fuente anterior deja de ser nodo fuente y adopta el rol de **Nodo Vecino**.

Cada nodo contiene el código para desempeñar ambos roles, con el uso de una bandera se activa un rol y se desactiva el otro, así cada nodo es potencialmente un nodo fuente y un nodo vecino.

La arquitectura de la prueba de concepto se basa en el esquema de la Figura 4.3.

El esquema físico está compuesto por un conjunto de dispositivos Arduino One con escudos Ethernet y otros con escudo WiFly, además otros dispositivos PcDuinos que cuentan con comunicación inalámbrica. Todos se conectan a un mismo segmento de red e intercambian mensajes con un vecindario acotado. La comunicación entre PCduinos y Arduinos con escudo Ethernet se da a través del protocolo TCP／ IP y la comunicación entre arduinos con escudo Ethernet se establecer por medio del protocolo UDP. Así cada dispositivo tiene las funciones $envia()$ y

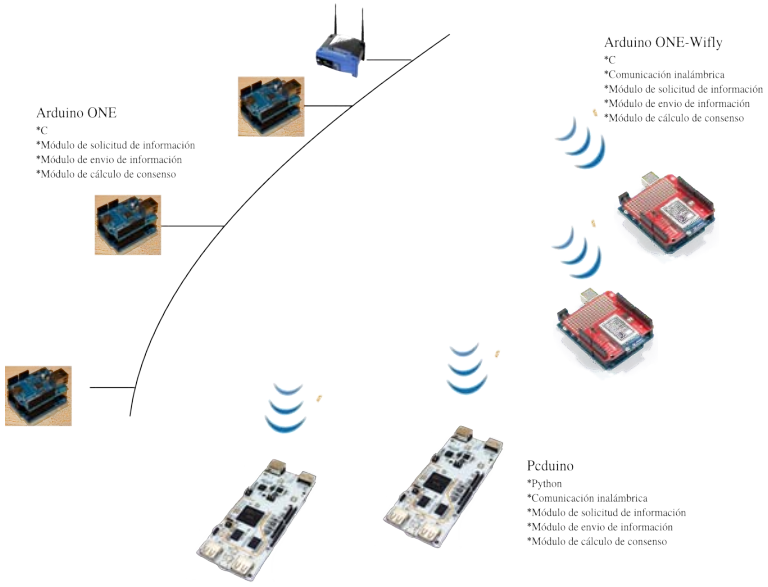


Figura 4.3: Esquema físico.

$recibe()$, y se ejecutan dependiendo del rol que esta desempeñando el dispositivo en ese momento.

Se han identificado 2 esquemas para la implementación del *AEC*. Un esquema centralizado, que utiliza al nodo fuente para formar un arreglo y distribuirlo a los participantes del consenso, con el fin de reducir las comunicaciones y el esquema distribuido que presenta algunos dilemas, a continuación se describen.

4.6.1 Esquema centralizado

El esquema de comunicación es el siguiente:

Fase 1 n_s busca entre sus vecinos al nodo destino, si no lo encuentra solicita información a sus vecinos mandando la instrucción de consenso. Como se muestra en el inciso a) de la Figura 4.4.

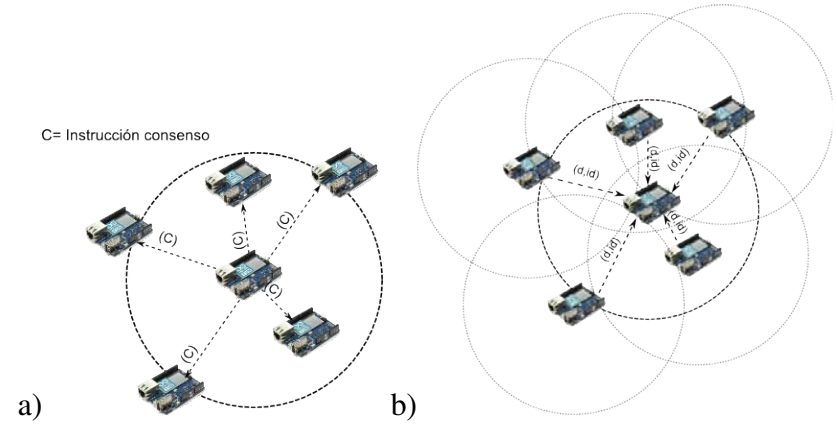


Figura 4.4: a) Fase 1. Enviar instrucción de consenso. b) Fase 2. Recibir datos de los nodos vecinos.

Fase 2 n_s recolecta los datos a evaluar de cada nodo vecino. Arma un arreglo con los datos recolectados (inciso b) de la Figura 4.4).

Fase 3 se envía el arreglo formado por los datos de los vecinos; la razón para que un nodo forme el arreglo es porque solo ese nodo tiene comunicación con todos los nodos vecinos, entre los vecinos hay algunos que no son a su vez vecinos directos (inciso a) de la Figura 4.5).

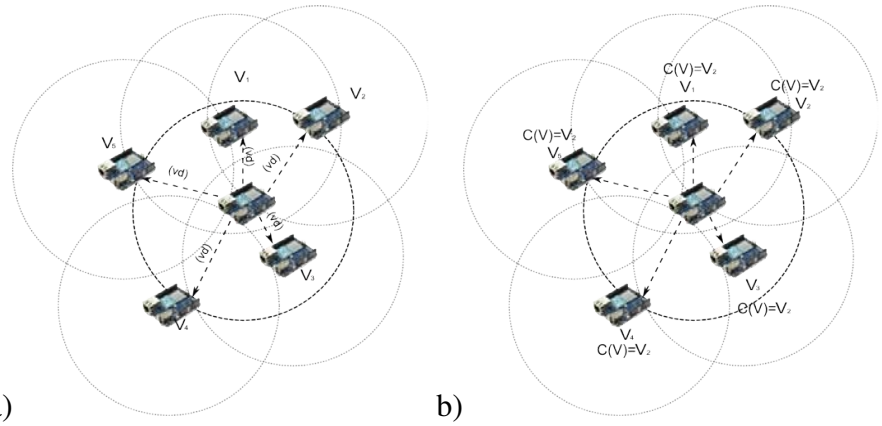


Figura 4.5: a) Fase 3. Enviar arreglos con los datos recolectados, b) Fase 4. Cálculo del consenso e identificación del ganador.

Fase 4 el consenso es calculado por cada nodo vecino, al tener los mismos datos llegarán al mismo resultado, el nodo que se identifique como ganador ini-

ciará el consenso con sus vecinos directos y los demás volverán a un estado pasivo, como se muestra en el inciso b de la Figura 4.5) .

Fase 5 el nodo ganador del proceso de consenso se ha identificado y este inicia el proceso de nuevo (Fase 1, Figura 4.4), ahora con sus vecinos (Figura 4.6).

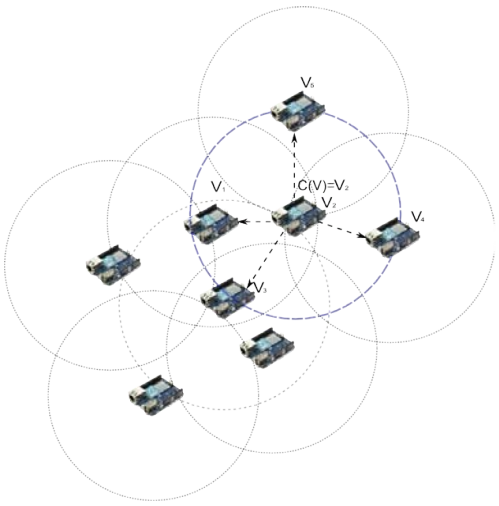


Figura 4.6: Fase 5: Iniciar consenso con el ganador anterior.

4.6.2 **Esquema distribuido**

Para este esquema se considera que se tiene un SDM compuesto por 24 nodos (Figura 4.7). En la Figura se encuentran distribuidos de forma en que los nodos del centro formen grupos fuertemente conexos y los nodos situados en el limite se descarten por tener pocas conexiones. El SDM se describe en la matriz de adyacencia α (ecuación 4.1).

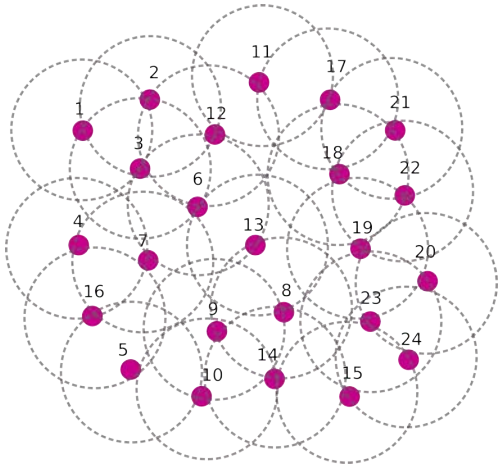


Figura 4.7: Topología de SDM de 24 nodos.

$$\alpha = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \\ 21 \\ 22 \\ 23 \\ 24 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{4.1}$$

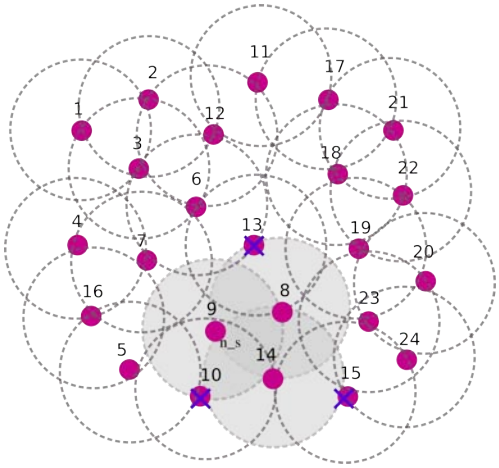


Figura 4.8: Formación de un grupo fuertemente conectado y los nodos eliminados el consenso.

Este esquema presenta varios problemas al realizarse el consenso, entre ellos:

- Para llevar a cabo el consenso todos los participantes deben tener el mismo número de entradas. Por lo tanto, el grupo donde se llevará a cabo el consenso debe ser un grafo fuertemente conexo (es decir, un grafo dirigido tal que para cualesquiera dos vértices a y b existe un camino dirigido de ida y de regreso), lo que reduce el número de vecinos a evaluar e incrementa la probabilidad de que se presenten ciclos de encaminamiento, como se puede ver en la Figura 4.8. Se muestra la formación de un un grupo que aparentemente tiene un vecindario muy grande, sin embargo al eliminar a los nodos que no tienen conexión con los demás nodos se reduce drásticamente la cardinalidad del vecindario conjunto.

- Se puede dar la formación de varios grafos fuertemente conexos en la misma topología pero desconexos entre si (un bosque), por la eliminación de nodos para el consenso, como se muestra en la Figura 4.9. En esta Figura (4.9) se muestran 3 grupos que parecen conectados entre si, pero al eliminar los nodos que no se conectan con todos los nodos con los que se conecta el grupo vecino estos grupos quedan aislados, haciendo imposible comunicar al SDM.

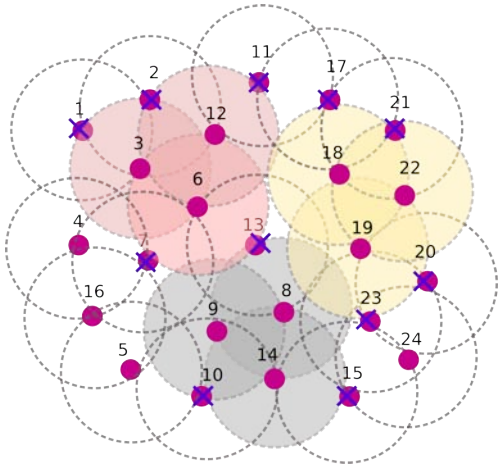


Figura 4.9: Formación de grupos fuertemente conexos pero desconexos entre si.

4.7 Modulos de comunicación

El modulo *recibe* (4.1) es un cliente Ethernet, se encarga de abrir un puerto y esperar una conexión de un servidor, cuando esto sucede escribe en el puerto serie el tamaño del paquete recibido y la dirección IP de donde se envió.

Código 4.1: Modulo recibe()

```
1
2   double recibe (){
3     long duration ;
4     startTime = millis () ;
5     EthernetClient client =server . available () ;
6     if ( client ) {
7       boolean currentLineIsBlank = true ;
8       while ( client .connected() ) {
9         if ( client . available () ) {
10          char c = client .read() ;
11          Serial . println (c) ;
12        }
13      }
14        duration = millis () - startTime ;
```

```
15            Serial . println ( duration );
16            return duration ;
17            }
18          startTime = millis ();
19          int packetSize = Udp.parsePacket();
20          if( packetSize )
21          {
22           Serial . print ("Paquete de recibido de tamanio ");
23           Serial . println ( packetSize );
24           Serial . print ("Desde ");
25           IPAddress remote = Udp.remoteIP();
26           for (int i =0; i < 4; i++)
27           {
28              Serial . print (remote[ i ], DEC);
29              if (i < 3)
30              {
31                 Serial . print (".");
32              } }
33           Serial . print (", port ");
34           Serial . println (Udp.remotePort());
35
36          // leer el paquete dentro del buffer
37           Udp.read( packetBuffer ,UDP_TX_PACKET_MAX_SIZE);
38           Serial . println ("Contents:");
39           Serial . println ( packetBuffer );
40
41          // mandar replica a la IP remota por el puerto acordado
42           Udp.beginPacket(Udp.remoteIP(), 2500);
43           Udp.write(ReplyRecibe);
44           Udp.endPacket();
45        }
46           duration = millis () - startTime;
47           Serial . println ( duration );
48           return duration ;
49           Serial . flush ();
50           delay(10);     }
```

El modulo 4.2 se encarga de enviar a un dirección IP (especificada en el código) y por un puerto dado un paquete de datos.

Código 4.2: Modulo envia()

```
1  void envia(){
2
3      IPAddress envio(192,168,10,52);
4      Udp.beginPacket(envio, 2500);
5      Serial.print("enviando paquete de tamanio ");
6      Udp.write(ReplyEnvia);
7      Udp.endPacket();
8      int packetSize = Udp.parsePacket();
9      if(packetSize){
10       Serial.print("Paquete recibido de tamanio ");
11       Serial.println(packetSize);
12       Serial.print("desde ");
13       IPAddress remote = Udp.remoteIP();
14       for (int i=0; i<4; i++){
15         Serial.print(remote[i], DEC);
16         if (i<3){
17           Serial.print(".");
18           }
19         }
20         Serial.print(", puerto ");
21         Serial.println(Udp.remotePort());
22         Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);
23         Serial.println("Contenido:");
24         Serial.println(packetBuffer);
25         Serial.flush();
26       }
27     }
```

En el caso de python se escribió una biblioteca para el manejo de las comunicaciones,que utiliza sockets para establecer comunicación por un puerto dado hacia el Arduino con escudo Ethernet. Esta biblioteca consta de una función de envío y

otra de recepción de mensajes. Así como una para enviar la información del nodo, otra que solicita la información del nodo y otra que obtiene la carga del nodo. En el diagrama de actividad mostrado en la Figura 4.10 se muestra el estado Nodo Fuente del diagrama de caso de uso mostrado en 4.2.

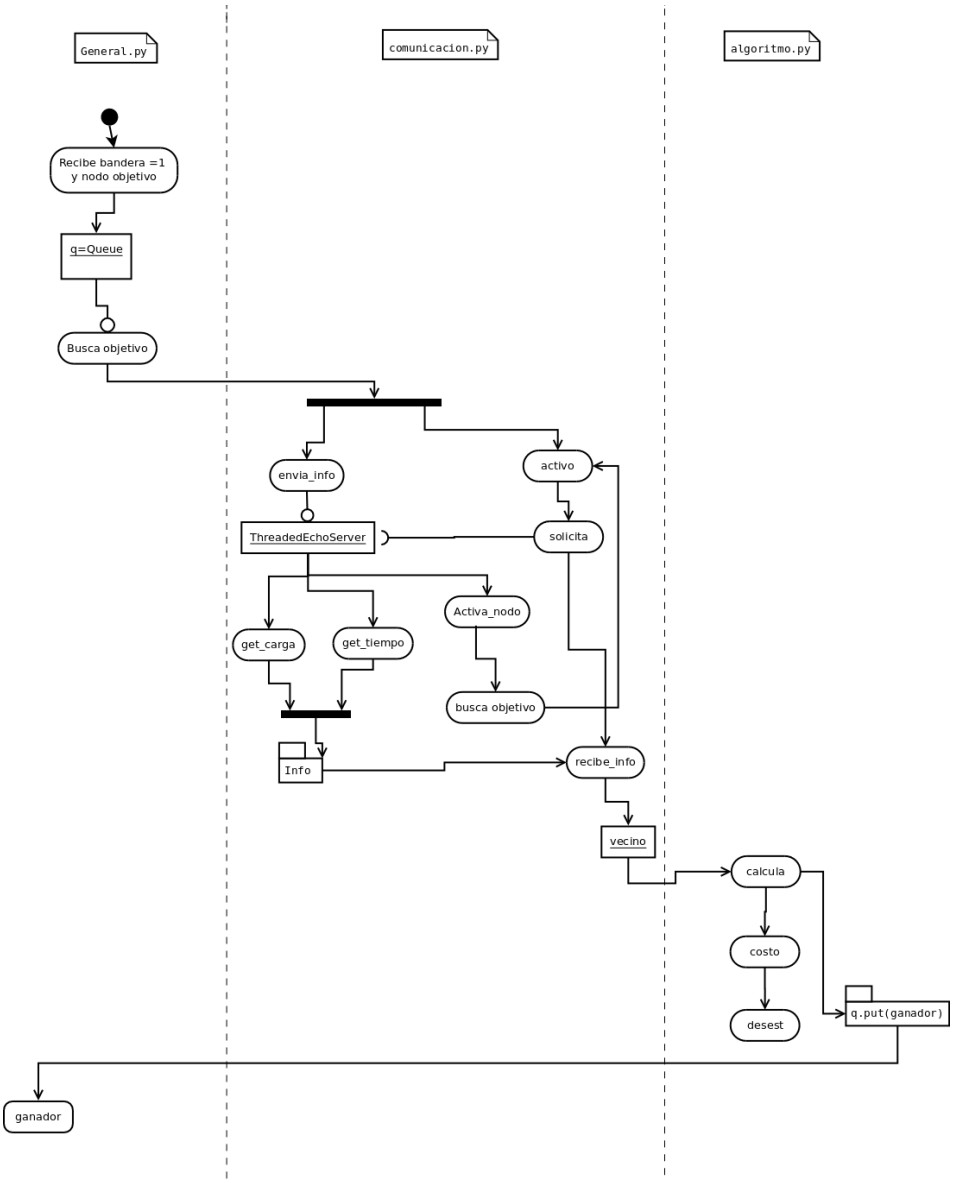


Figura 4.10: Diagrama de actividad del estado del Nodo Fuente del caso de uso.

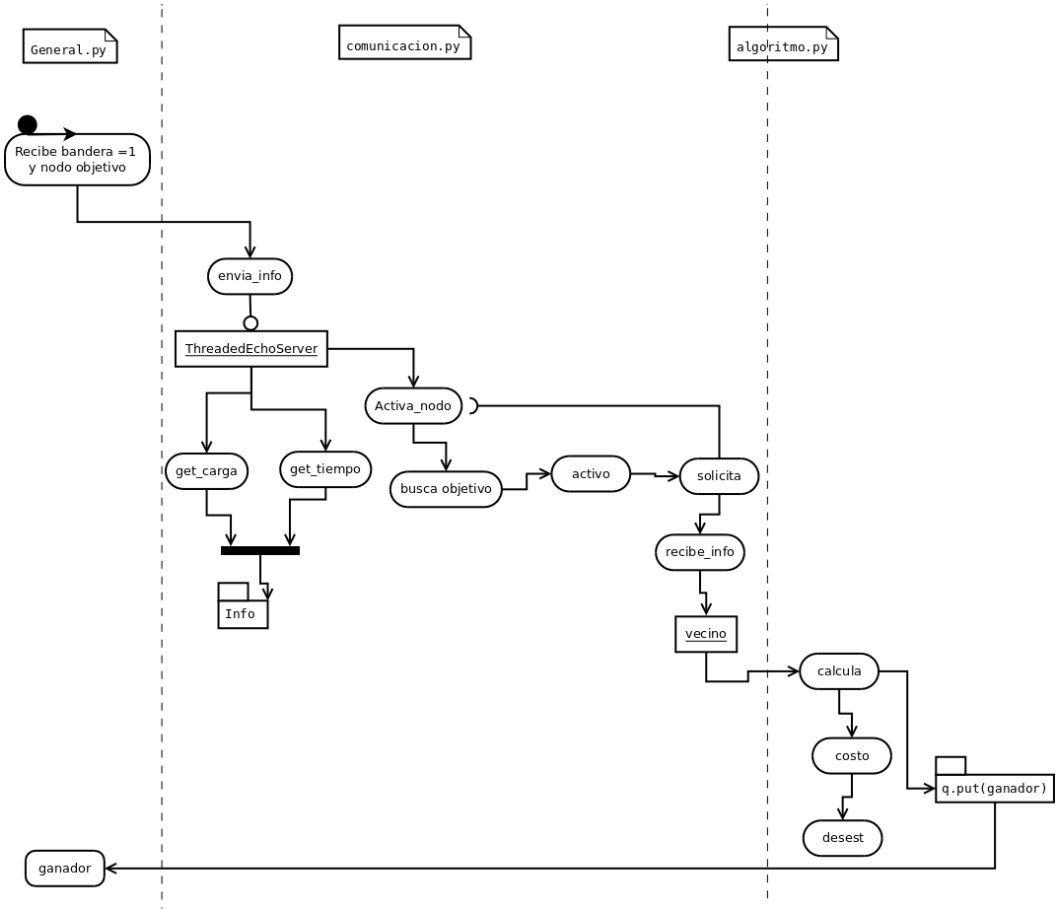


Figura 4.11: Diagrama de actividad del estado del Nodo Vecino del caso de uso.

En el diagrama de actividad mostrado en el diagrama 4.11 se muestra el funcionamiento del caso de uso Nodo Vecino.

Los casos de uso descritos fueron implementados en una red compuesta por 15 dispositivos Arduino ™Yun, con la siguiente topología (4.12).

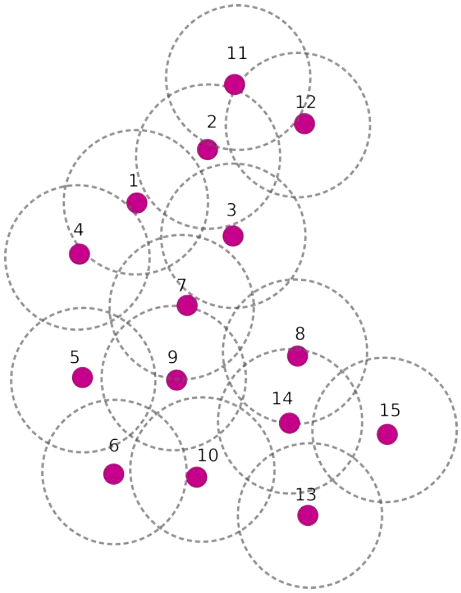


Figura 4.12: Esquema real con 15 Arduinos Yun ®.

Los números corresponden a la siguiente tabla 4.8

4.7.1 Programa principal

El programa principal hace uso de funciones especificas de la biblioteca de comunicación, como se muestra en 4.3.

Código 4.3: Funciones importadas

```
1 # -*- coding: utf-8 -*-
2 import sys
3 import threading
4 from comunicacion import activo
5 from comunicacion import envia_info
6 from comunicacion import busca
```

Cada dispositivo contiene todo el código descrito, la función principal contiene un diccionario que indica el identificador de cada dispositivo que lo relaciona con su tabla de vecinos, correspondientes a 4.4.

ID	Nombre	MAC	IP
1	Andromeda	90:A2:DA:FA:16:95	192.168.10.109
2	Cygnus	90:A2:DA:F3:04:04	192.168.10.115
3	Auriga	90:A2:DA:F5:0B:F3	192.168.10.113
4	Lebreles	90:A2:DA:F3:02:FA	192.168.10.110
5	Camelopalis	90:A2:DA:F1:17:4C	192.168.10.111
6	Casiopea	90:A2:DA:F5:05:22	192.168.10.112
7	Cetus	90:A2:DA:F5:10:BC	192.168.10.108
8	Chamaeleon	90:A2:DA:F2:16:29	192.168.10.116
9	Columba	90:A2:DA:F1:16:9F	192.168.10.105
10	Crux	90:A2:DA:F5:11:8B	192.168.10.103
11	Draco	90:A2:DA:F0:2C:A3	192.168.10.104
12	Hydra	90:A2:DA:F7:03:88	192.168.10.106
13	Lepus	90:A2:DA:F5:0B:15	192.168.10.123
14	Lyra	90:A2:DA:F5:0A:FB	192.168.10.107
15	Phoenix	90:A2:DA:F5:10:1E	192.168.10.122

Tabla 4.8: Lista de dispositivos Arduino ™Yun.

Código 4.4: Diccionario

```
1 ids={"andromeda":0,"cygnus":1,"auriga":2,"lebreles":3,"
      camelopardalis":4,
2 "casiopea":5,"cetus":6,"chamaeleon":7,"columba":8,"crux":9,"
      draco":10,
3 "hydra":11,"lepus":12,"lyra":13,"phoenix":14,"prueba":15}
```

El programa se ejecuta en linea de comandos y recibe como parámetros el identificador (nombre) del dispositivo local, el identificador del nodo objetivo y la bandera que indica que se busca un objetivo 4.5.

Código 4.5: parámetros del programa principal

```
1 if (len(sys.argv)!=4):
2     print "error de parametros"
```

```
3       print "uso:",sys.argv[0], "nombre del programa nombre del
            dispositivo objetivo bandera"
4       sys.exit(-1)
```

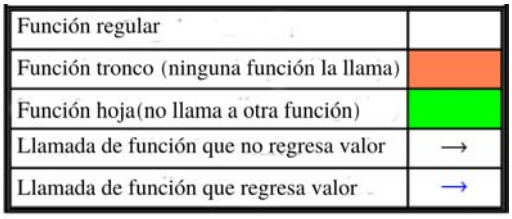


Figura 4.13: Simbología de los diagramas de funciones

El programa principal inicia la búsqueda del objetivo y si no lo encuentra en su vecindario comienza a pedir información a sus vecinos, iniciando 2 hilos que llaman a la función `activo(flag,id1)` e inicializa un servidor de sockets con `envia_info(id1)`.

La relación de las funciones usadas en el programa `general.py`, se muestra en los diagramas mostrados en las secciones siguientes, en la Figura 4.13 se describe la simbología usada en estos diagramas UML. Estos diagramas de funciones muestran la interacción de las funciones entre si.

4.7.2 Clase Vecino

La estructura de la clase `Vecino` se muestra en 4.14. En La función `activo` perteneciente a la biblioteca de comunicación implementa un arreglo de instancias de la clase `Vecino` que se nombra `vecinos`.

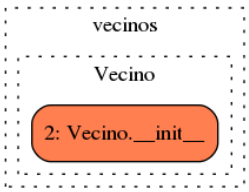


Figura 4.14: Diagrama de la clase Vecino.

Biblioteca de comunicación

La relación de las funciones de la biblioteca Comunicación se muestra en la figura 4.15.

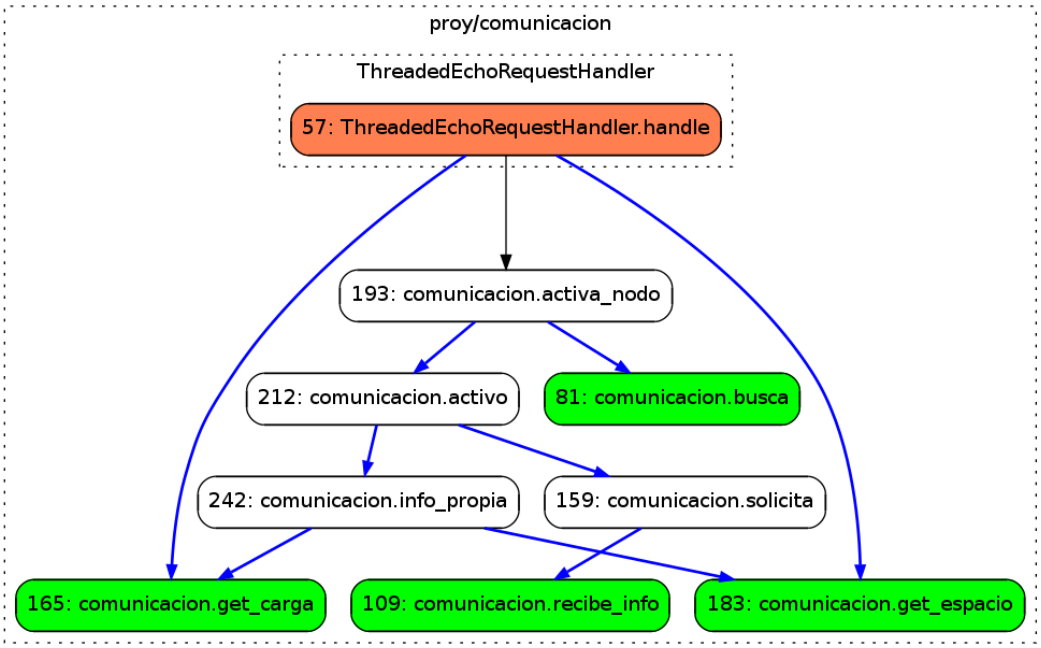


Figura 4.15: Diagrama de la relación de las funciones de la biblioteca Comunicación.

La biblioteca de comunicación contiene la clase que se encarga del manejo de los sockets que sirven para comunicar los dispositivos. Para la implementación de los sockets se hace uso de la biblioteca `socketserver` de python. En 4.6 se muestra la declaración de las clases `ThreadedEchoRequestHandler` y `ThreadedEchoServer` (4.6). La función `handle(self)` debe hacer todo el trabajo necesario para atender una solicitud. La implementación por defecto no hace nada.

Varios atributos de instancia están a su disposición; la solicitud está disponible como `self.request`; la dirección del cliente como `self.client_address`; y la instancia de servidor como `self.server`, en caso de que necesita tener acceso a la información por servidor. En este caso se hacen llamados a las funciones encargadas de obtener los valores que determinan el estado de disponibilidad de un nodo, la carga y el tiempo disponible en el planificador del nodo.

Código 4.6: Uso de socketserver

```
1  import SocketServer
2
3  class ThreadedEchoRequestHandler(SocketServer.BaseRequestHandler):
4
5      def handle( self ):
6          data1= str( get_carga ('localhost'))+',' +str(get_espacio())
7          response = '%s:' % (data1)
8          self . request .send(response)
9          return
10
11 class ThreadedEchoServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer
       ):
12     pass
```

Función activo( )

Esta función (4.7), se encarga de implementar el arreglo de instancias de objetos Vecino, solicitando los identificadores de los vecinos de un nodo en especifico y calcular el consenso entre ellos, recibir el valor ganador.

Código 4.7: activo()

```
1  """
2  Funcion correspondiente al estado de busqueda de un nodo
       objetivo
3  INPUT: flag y su id
4  OUTPUT: flag
5  """
6  def activo ( flag ,id1 ):
7      #while 1:
8      if int( flag ) == 1:
9          vecinos=[]
10         for i in range(len( vecindarios [id ]) ):
11             info= solicita ( ids [id1 ], vecindarios [id1 ][ i ])
12             a = Vecino( info [0], float ( info [1]) , float ( info [2]) )
```

```
13                  vecinos.append(a)
14                  ganador= calcula(vecinos)
15                  print vecindarios[id1][ganador]
16                  print 'el ganador esta en la posicion '+str(ganador)+
                        ' con valores '+str(vecindarios[id][ganador])
17                  if id == ganador:
18                      flag = 1
19                  else:
20                      flag = 0
```

Función solicita( )

Esta función se encarga de obtener los valores de estado de cada instancia Vecino, llamando a la función `recibe_info` (4.8) para cada uno de ellos, recibe la información y la envía a la subrutina que la invocó.

Código 4.8: solicita()

```
1  """
2  Funcion solicita: solicita informacion a un nodo remoto
3  INPUT: id local, id remota
4  OUTPUT: tupla con la informacion de estado de id2
5  """
6  def solicita(id1,id2):
7      info=recibe_info(id1,id2)
8      return info
```

Función recibe_info()

La función `recibe_info()` (4.9)se encarga de crear un socket cliente, hacer la petición de información a cada nodo vecino y reenviarla al programa que lo solicitó.

Código 4.9: recibe_info()

```
1  """
2  Funcion se conecta a una ip y recibe informacion
```

```
3   INPUT: id local e id remoto
4   OUTPUT: un paquete de datos correspondiente a carga y
        espacio de un vecino dado
5   """
6   def recibe_info (id1,id2):
7       puerto = 5555
8       miSocket = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
9       miSocket.connect( (id2, puerto))
10      while True:
11          miSocket.send(id1)
12          data, server = miSocket.recvfrom( 100 )
13          print data
14          break
15      miSocket.close ()
16      datos= data.split (',')
17      if len(datos) == 2:
18          vecino = id2
19          carga = datos[0]
20          espacio = datos[1]
21          return(vecino,carga,espacio)
```

Función busca( )

La función `busca()` 4.10 recibe como parámetro de entrada un identificador y un objetivo. La función compara el objetivo con los identificadores del vecindario correspondiente al identificador dado, si el objetivo se encuentra en el vecindario se devuelve una bandera con valor 2.

Código 4.10: busca()

```
1   """
2   Funcion que busca el nodo objetivo en un vecindario dado,
3   INPUT: id del nodo y la direccion IP del nodo objetivo
4   OUTPUT: 2 si es encontrado, 0 si no
5   """
6   def busca(id1, obj):
```

```
7       for i in len[ vecindarios [id1 ]]:
8           if vecindarios [id1 ][ i]==obj:
9               return 2
10          else :
11              return 0
```

Función get_espacio( )

La función `get_espacio` (4.11)

Código 4.11: get_espacio()

```
1   """
2   Funcion espacio: obtiene la informacion de espacio en el
       planificador de tareas
3   INPUT:
4   OUTPUT: valor flotante correspondiente al espacio
5   """
6   def get_espacio () :
7
8       p=os.popen("mpstat | tail -1 | cut -f 44-46 -d' '",'r')
9       while 1:
10          line = p. readline ()
11          if not line : break
12          load=line
13          j=float (load . split ('\n')[0])
14          return j
```

Función get_carga( )

Esta función (4.12) obtiene el retardo entre un dispositivo y otro mediante el uso de la función de sistema `ping`.

Código 4.12: get_carga()

```
1   """
```

```
2   Funcion carga: obtiene la informacion de carga en el canal
        de un nodo dado
3   INPUT: IP
4   OUTPUT: valor flotante
5   """
6   def get_carga(ip):
7
8       p=os.popen("ping -c 1 %s| head -2 | tail -1 | cut -f4 -d
            '=' | cut -f1 -d' '" % ip,'r')
9       while 1:
10          line = p.readline()
11          if not line: break
12          load=line
13          j=float(load.split('\n')[0])
14          return j
```

4.7.3 Biblioteca del algoritmo

La relación existente entre las funciones de la biblioteca Algoritmo se muestra en la figura 4.16.

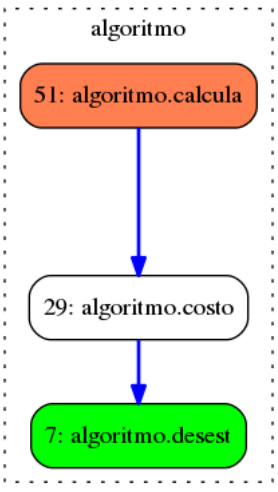


Figura 4.16: Diagrama de la relación de las funciones de la biblioteca Algoritmo.

Esta biblioteca está compuesta por 3 funciones que se encargan del cálculo del consenso, la principal es `calcula()` que recibe un arreglo con la información correspondiente al estado de los nodos del vecindario, esta función llama a la función costo (que a su vez llama la función `desest` que calcula de desviación estándar [1] que se usa para calcular el costo por nodo, ver el Apéndice A, Figura A.4. Para ver de forma detallada el procedimiento seguido para el cálculo del consenso ver el Apéndice A, Figura A.2.

En general el sistema se comporta como en la Figura 4.17.

4.8 Resumen

Las simulaciones que se realizaron abarcan los casos de estudio descritos en el capítulo anterior, pero también con propósitos de comparación se realizaron las mismas simulaciones dejando fijo el parámetro de carga para así poder observar el comportamiento del*AEC*.

En las pruebas de la versión con topología móvil se midió la influencia de la movilidad para la toma de decisiones. También se realizaron comparativas de*AEC* contra el algoritmo de inundación clásico y el algoritmo de Dikjstra.

Para las pruebas en hardware se especificaron 2 implementaciones, el caso centralizado y el caso distribuido.

[1] No se usó ninguna biblioteca de cálculo numérico como Numpy porque es incompatible con el sistema operativo de los dispositivos Arduino ™Yun, por lo que se optó por programar en Python estándar la desviación estándar

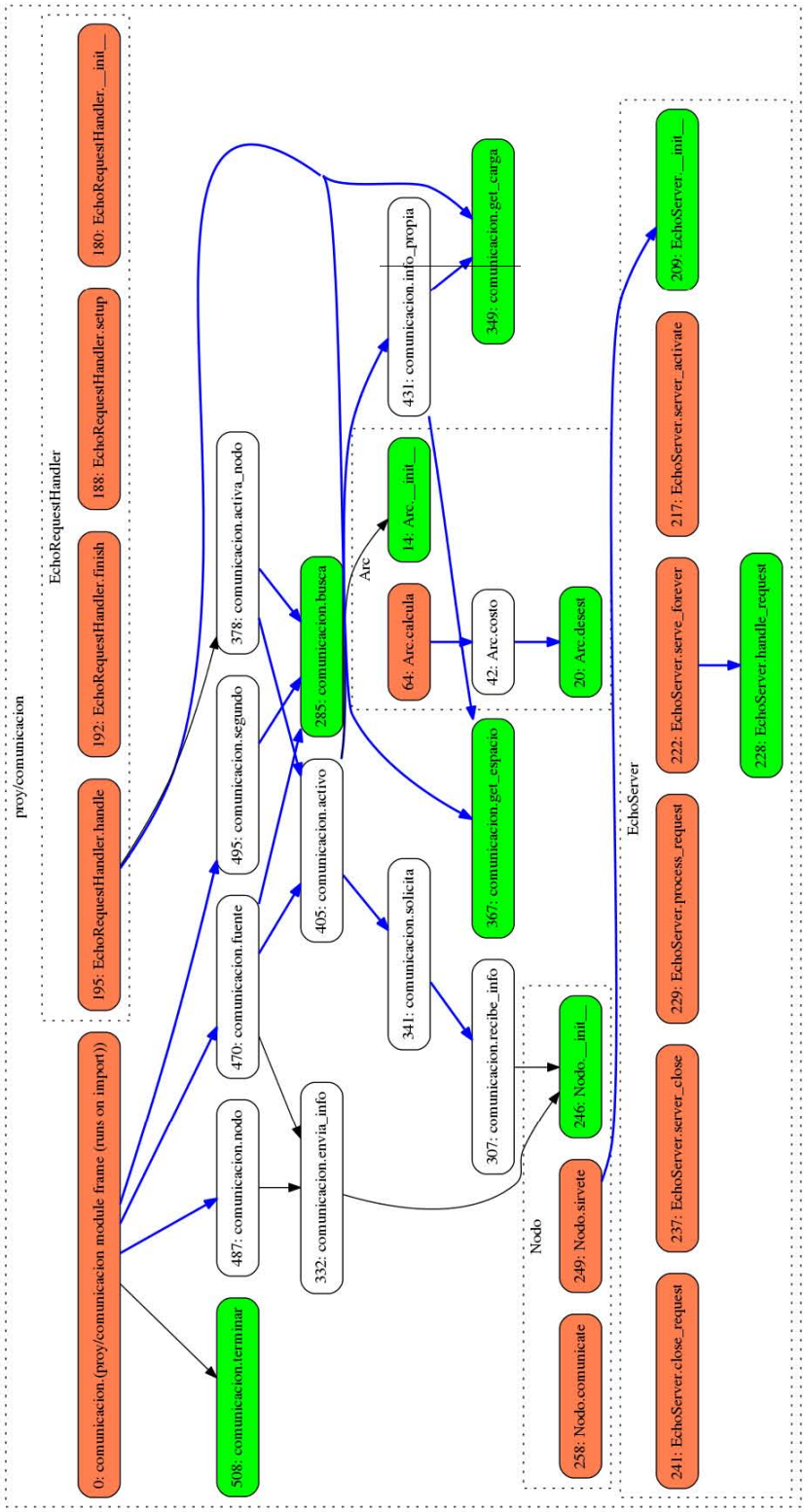


Figura 4.17: Diagrama general.

La ciencia son hechos; de la misma manera que las casas están hechas de piedras, la ciencia está hecha de hechos; pero un montón de piedras no es una casa y una colección de hechos no es necesariamente ciencia.

Henri Poincaré.

CAPÍTULO 5

Resultados

En esta sección se presentaran los resultados de la experimentación descrita en el capitulo 4. Se realizaron varios conjuntos de simulaciones, se han escogido casos representativos de estas para presentar los resultados. Se realizaron 3 versiones de estas simulaciones, cada versión cambiando el rango de alcance del vecindario (Capitulo 3 sección 3.1):

1. Vecindario de un salto.
2. Vecindario de dos saltos.
3. Vecindario de un uno y dos saltos.

5.1 Simulación con matrices de valores fijos

A continuación se presentan los resultados obtenidos al utilizar matrices con valores que fueron fijados, se presentan los resultados de este simulación para establecer un marco de referencia y poder mostrar posteriormente los resultados obtenidos al aplicar el algoritmo en las mismas topologías cambiando valores de carga.

Se presentan las simulaciones realizadas con matrices de carga fija. En esta simulación todos los valores (carga y tiempo disponible en el planificador del nodo)

permanecen fijos de principio a fin. De esta manera, es posible evaluar la influencia de la variación de carga en las simulaciones posteriores.

5.1.1 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto

En este primer ejemplo se muestra la ejecución completa del algoritmo y se detallan numéricamente las operaciones realizadas a cada paso del algoritmo. Ver apéndice B. En los siguientes ejemplos no se mostrarán los valores obtenidos a cada paso de manera numérica. En cambio se muestra el árbol de expansión que se forma a cada paso.

5.1.2 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de dos saltos

Consideremos el grafo mostrado en la figura 5.1 como el sistema distribuido con el que se desarrollará el ejemplo.

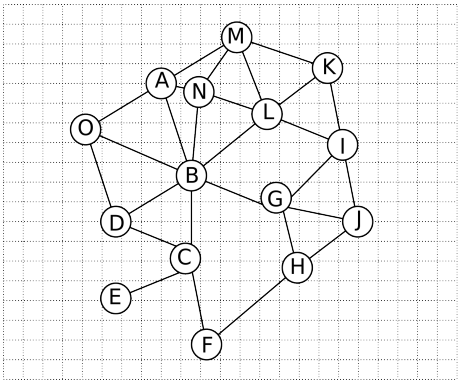


Figura 5.1: Grafo utilizado para el ejemplo de la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de dos saltos.

Sea α la matriz de adyacencia que representa este grafo.

$$\alpha = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.1}$$

Sea λ la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea γ el tiempo máximo que el planificador puede asignar a una tarea.

$$\lambda = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 8 \\ 5 & 0 & 10 & 2 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 17 & 0 & 4 & 1 \\ 0 & 10 & 0 & 14 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 11 & 5 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 11 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 10 & 18 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 9 & 8 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 0 & 16 & 11 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 16 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 3 & 0 & 0 \\ 8 & 1 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.2}$$

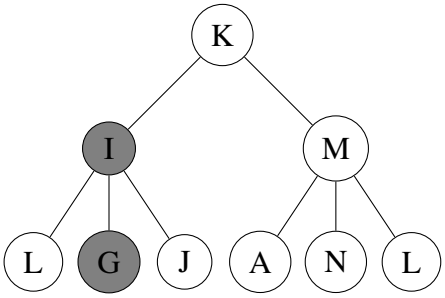
Sea β la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea τ la capacidad del canal de comunicación.

Consideremos para este ejemplo en particular que el nodo que inicia la búsqueda es el nodo K, entonces $n_s = K$ y el nodo destino es E, i.e. $n_d = E$. Se definen los valores de $\gamma = 20kbps$ y $\tau = 1000\mu s$. Se define β^1 como:

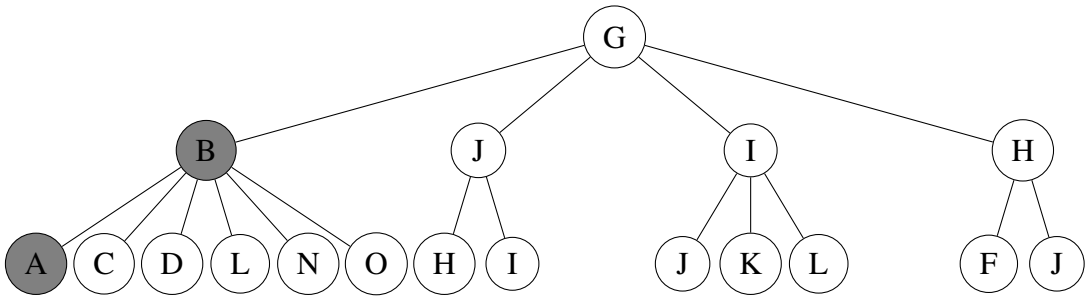
$$\beta^{\mathbf{1}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 875 & 355 & 103 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 875 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 355 & 0 & 0 & 108 & 18 & 92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 103 & 61 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 254 & 416 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 92 & 0 & 254 & 0 & 266 & 101 & 0 \\ 647 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 416 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 101 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.3}$$

La siguiente representación gráfica del algoritmo será descrita por medio de diagramas de árbol, debido a que expresar las operaciones y relaciones entre nodos para esta versión puede resultar confuso.

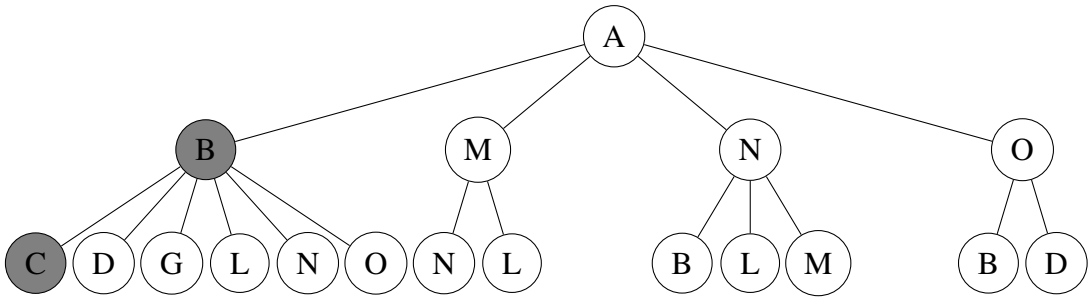
Por lo tanto, comenzando la búsqueda del nodo destino n_d a partir del nodo K se obtiene el siguiente diagrama de árbol, con el nodo K como raíz.



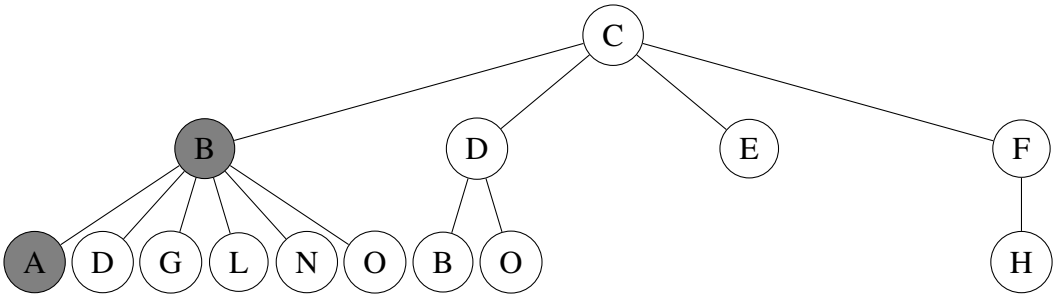
Al terminar todos los cálculos el algoritmo, el nodo K escoge la 3-tupla $K \rightarrow I \rightarrow G$ como la ruta ganadora, entonces la siguiente búsqueda comienza con el nodo G como raíz.



Después de la segunda iteración, la 3-tupla ganadora del consenso es $G \rightarrow B \rightarrow A$, para la siguiente iteración el nodo A será la raíz del árbol.



En la tercera iteración, la 3-tupla ganadora del consenso es $A \rightarrow B \rightarrow C$, para la siguiente iteración el nodo C será la raíz del árbol.



En esta iteración, ya está presente el nodo $n_d = E$, sin embargo, está en el primer nivel del árbol, por lo tanto se ignora y se escoge como 3-tupla ganadora $C \rightarrow B \rightarrow A$ formando un ciclo. La ruta resultante y fallida es $K \rightarrow I \rightarrow G \rightarrow B \rightarrow A \rightarrow B \rightarrow C \rightarrow B \rightarrow A$ En la figura 5.2 se muestra esta ruta.

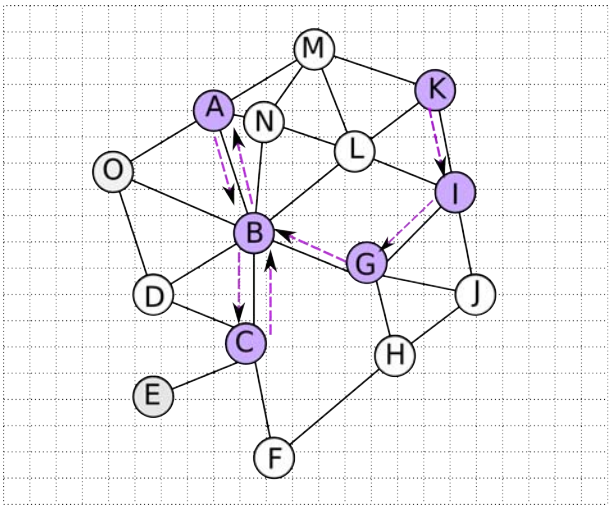


Figura 5.2: Ruta obtenida por la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de dos saltos.

5.1.3 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de uno y dos saltos

Consideremos el grafo mostrado en la figura 5.3 como el sistema distribuido con el que se desarrollará el ejemplo.

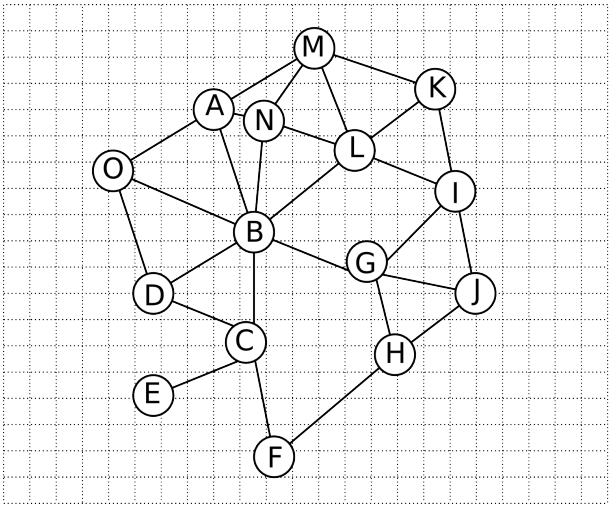


Figura 5.3: Grafo utilizado como ejemplo para la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de uno y dos saltos.

Sea α la matriz de adyacencia que representa este grafo.

$$\alpha = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.4}$$

Sea λ la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea γ el tiempo máximo que el planificador puede asignar a una tarea.

$$\lambda = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 8 \\ 5 & 0 & 10 & 2 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 17 & 0 & 4 & 1 \\ 0 & 10 & 0 & 14 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 11 & 5 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 11 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 10 & 18 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 9 & 8 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 0 & 16 & 11 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 16 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 3 & 0 & 0 \\ 8 & 1 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.5}$$

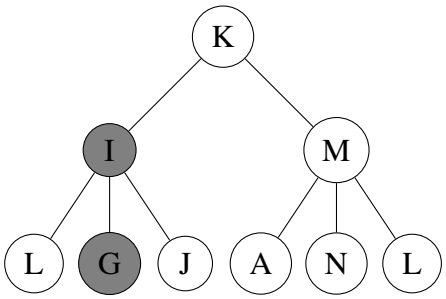
Sea β la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea τ la capacidad del canal de comunicación.

Consideremos para este ejemplo en particular que el nodo que inicia la búsqueda es el nodo K, entonces $n_s = K$ y el nodo destino es E, i.e. $n_d = E$. Se definen los valores de $\gamma = 20kbps$ y $\tau = 1000\mu s$. Se define β^1 como:

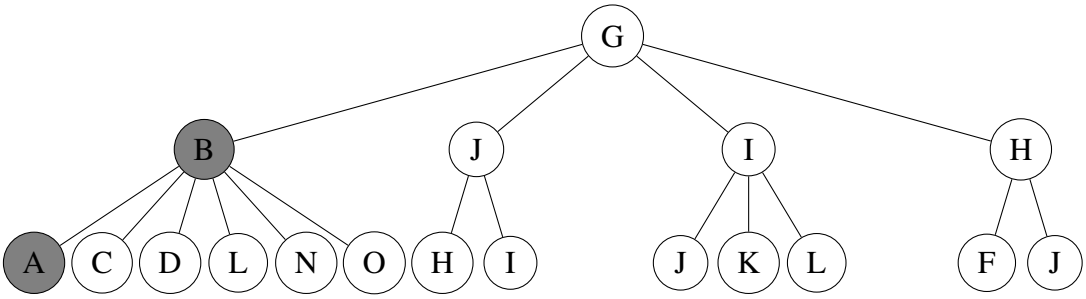
$$\beta^{\mathbf{1}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 875 & 355 & 103 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 875 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 355 & 0 & 0 & 108 & 18 & 92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 103 & 61 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 254 & 416 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 92 & 0 & 254 & 0 & 266 & 101 & 0 \\ 647 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 416 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 101 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.6}$$

La siguiente representación gráfica del algoritmo será descrita por medio de diagramas de árbol, debido a que expresar las operaciones y relaciones entre nodos para esta versión puede resultar confuso.

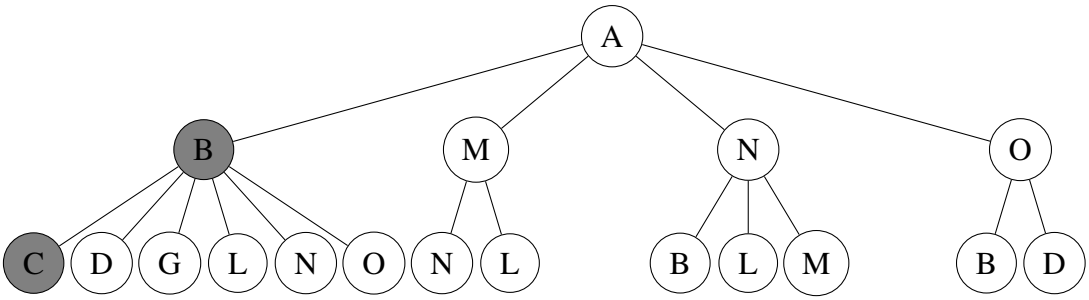
Por lo tanto, comenzando la búsqueda del nodo destino n_d a partir del nodo K se obtiene el siguiente diagrama de árbol, con el nodo K como raíz.



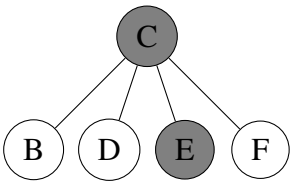
Al terminar todos los cálculos el algoritmo, el nodo K escoge la 3-tupla $K \rightarrow I \rightarrow G$ como la ruta ganadora, entonces la siguiente búsqueda comienza con el nodo G como raíz.



Después de la segunda iteración, la 3-tupla ganadora del consenso es $G \rightarrow B \rightarrow A$, para la siguiente iteración el nodo A será la raíz del árbol.



En la tercera iteración, la 3-tupla ganadora del consenso es $A \rightarrow B \rightarrow C$, para la siguiente iteración el nodo C será la raíz del árbol.



En la siguiente iteración se encuentra al nodo $n_d = E$ como vecino inmediato y no sé continua buscando al siguiente nivel, no se hacen cálculos ni consenso. La ruta completa y exitosa es $K \rightarrow I \rightarrow G \rightarrow B \rightarrow A \rightarrow B \rightarrow C \rightarrow E$

En la figura 5.4 se muestra esta ruta.

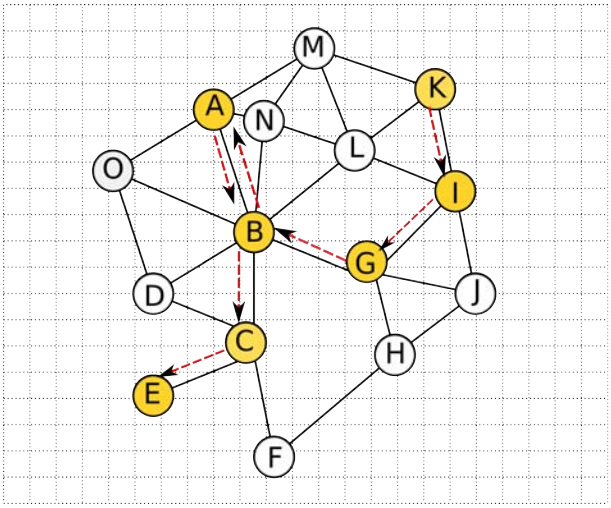


Figura 5.4: Ruta obtenida por la simulación con matriz de valores fijos para búsqueda utilizando un vecindario de uno y dos saltos.

Como se puede notar en los resultados de los ejemplos presentados, es posible que el algoritmo de encaminamiento por consenso tome decisiones que lleven a permanecer en un ciclo y no sea posible romperlo, dado que al permanecer fijas las condiciones de la red no existe ningún factor que permita cambiar la decisión del consenso. En el siguiente ejemplo se muestra como se disminuye la ocurrencia de este problema cambiando los valores de la matriz de carga periódicamente.

5.2 Simulación con una matriz de carga variable

En esta sección se muestran los ejemplos representativos de las simulaciones en donde la matriz de carga es variada constantemente. Esta variación en la matriz de carga permite influir en la elección del nodo ganador del consenso. Los valores de carga se varían con cierta frecuencia y modifica los valores a evaluar y por lo tanto pueden cambiar la decisión resultante del consenso.

5.2.1 Simulación con la matriz de carga variable para búsqueda de un salto

Este primer ejemplo se presentan todos los cálculos realizados por el algoritmo de consenso de manera numérica. Ver apéndice B

Los siguientes ejemplos representativos no se presentarán numéricamente, se mostrará la evolución de los árboles de expansión en cada paso de la formación de la ruta.

5.2.2 Simulación con una matriz de carga variable para búsqueda de dos saltos

Consideremos el grafo mostrado en la figura 5.5 como el sistema distribuido con el que se desarrollará el ejemplo.

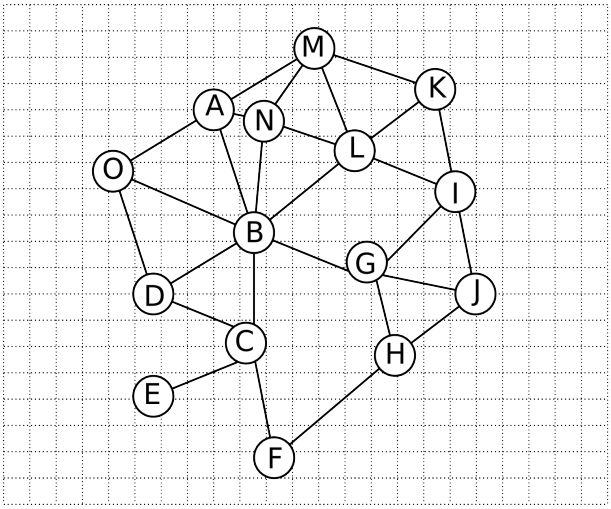


Figura 5.5: Grafo utilizado para el ejemplo de la simulación con la matriz de carga variable para búsqueda de dos saltos.

Sea α la matriz de adyacencia que representa este grafo.

$$\alpha = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.7}$$

Sea λ la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea γ el tiempo máximo que el planificador puede asignar a

una tarea.

$$\lambda = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 8 \\ 5 & 0 & 10 & 2 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 17 & 0 & 4 & 1 \\ 0 & 10 & 0 & 14 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 11 & 5 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 11 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 10 & 18 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 9 & 8 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 0 & 16 & 11 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 16 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 3 & 0 & 0 \\ 8 & 1 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.8}$$

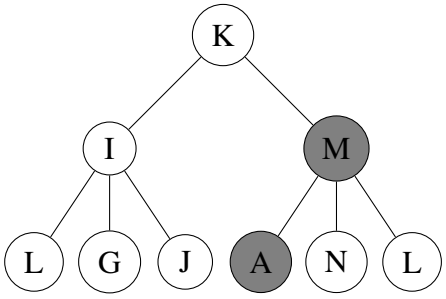
Sea β la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea τ la capacidad del canal de comunicación.

Consideremos para este ejemplo en particular que el nodo que inicia la búsqueda es el nodo K, entonces $n_s = K$ y el nodo destino es E, i.e. $n_d = E$. Se definen los valores de $\gamma = 20$ y $\tau = 1000$. Se define β^1 como:

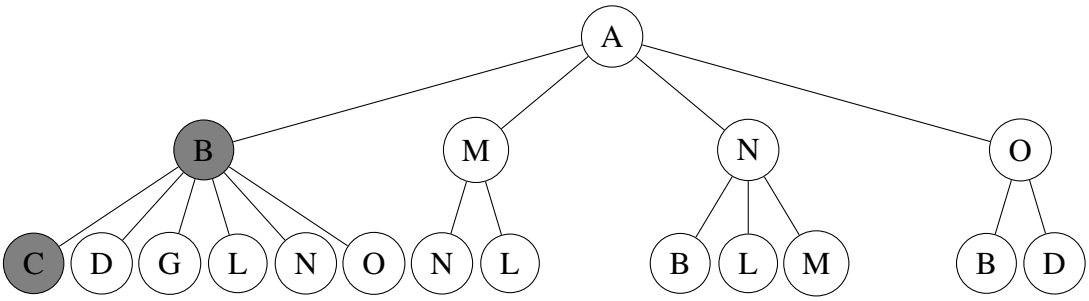
$$\beta^{\mathbf{1}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 875 & 355 & 103 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 875 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 355 & 0 & 0 & 108 & 18 & 92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 103 & 61 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 254 & 416 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 92 & 0 & 254 & 0 & 266 & 101 & 0 \\ 647 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 416 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 101 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.9}$$

La siguiente representación gráfica del algoritmo será descrita por medio de diagramas de árbol, debido a que expresar las operaciones y relaciones entre nodos para esta versión puede resultar confuso.

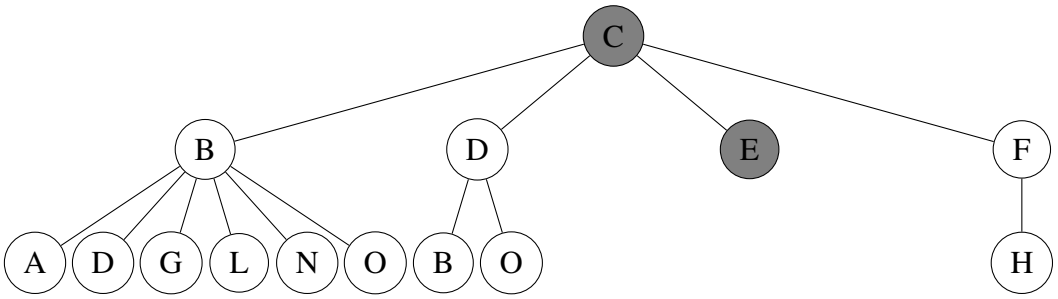
Por lo tanto, comenzando la búsqueda del nodo destino n_d a partir del nodo K se obtiene el siguiente diagrama de árbol, con el nodo K como raíz.



Al terminar todos los cálculos el algoritmo, el nodo K escoge la 3-tupla $K \rightarrow M \rightarrow A$ como la ruta ganadora, entonces la siguiente búsqueda comienza con el nodo A como raíz.



En la segunda iteración, la 3-tupla ganadora del consenso es $A \rightarrow B \rightarrow C$, para la siguiente iteración el nodo C será la raíz del árbol.



En esta iteración, ya está presente el nodo $n_d = E$, sin embargo, está en el primer nivel del árbol, por lo tanto se ignora y se escoge como 3-tupla ganadora $C \rightarrow B \rightarrow A$ formando un ciclo. La ruta resultante y fallida es $K \rightarrow M \rightarrow A \rightarrow B \rightarrow C \rightarrow B \rightarrow A$ En la figura 5.6 se muestra esta ruta.

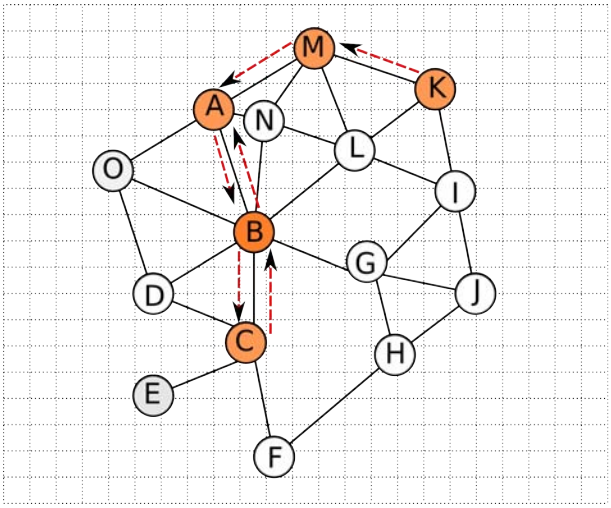


Figura 5.6: Ruta obtenida de la simulación con la matriz de carga variable para búsqueda de dos saltos.

5.2.3 Simulación con una matriz de carga variable para búsqueda de uno y dos saltos

Consideremos el grafo mostrado en la figura 5.7 como el sistema distribuido con el que se desarrollará el ejemplo.

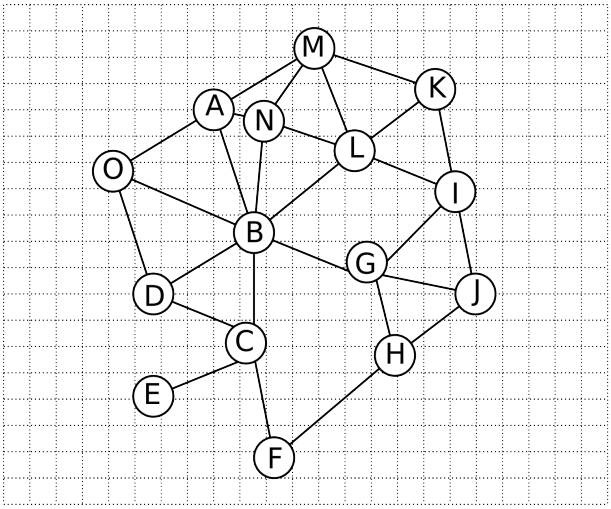


Figura 5.7: Grafo del ejemplo de la simulación con la matriz de carga variable para búsqueda de uno y dos saltos.

Sea α la matriz de adyacencia que representa este grafo.

$$\alpha = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.10}$$

Sea λ la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea γ el tiempo máximo que el planificador puede asignar a una tarea.

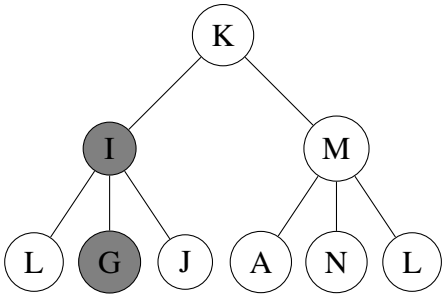
$$\lambda = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 8 \\ 5 & 0 & 10 & 2 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 17 & 0 & 4 & 1 \\ 0 & 10 & 0 & 14 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 11 & 5 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 11 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 10 & 18 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 9 & 8 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 0 & 16 & 11 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 16 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 3 & 0 & 0 \\ 8 & 1 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.11}$$

Sea β la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea τ la capacidad del canal de comunicación. Consideremos para este ejemplo en particular que el nodo que inicia la búsqueda es el nodo K, entonces $n_s = K$ y el nodo destino es E, i.e, $n_d = E$. Se definen los valores de $\gamma = 20$ y $\tau = 1000$. Se define β^1 como:

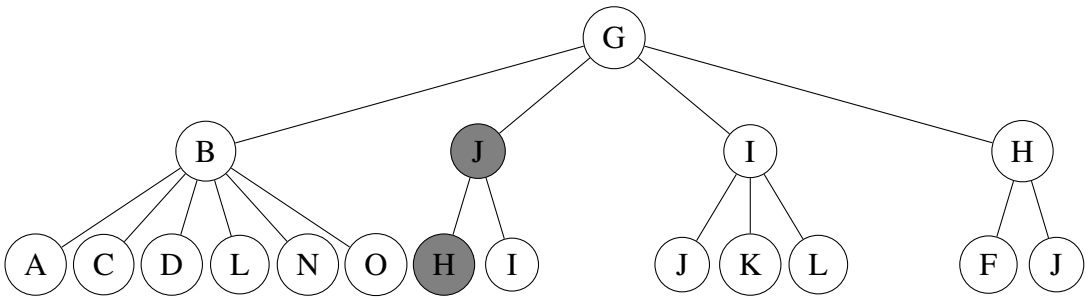
$$\beta^{\mathbf{1}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 875 & 355 & 103 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 875 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 355 & 0 & 0 & 108 & 18 & 92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 103 & 61 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 254 & 416 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 92 & 0 & 254 & 0 & 266 & 101 & 0 \\ 647 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 416 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 101 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.12}$$

La siguiente representación gráfica del algoritmo será descrita por medio de diagramas de árbol, debido a que expresar las operaciones y relaciones entre nodos para esta versión puede resultar confuso.

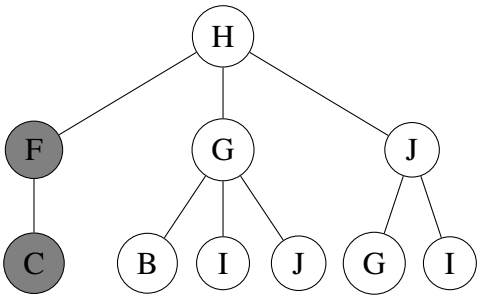
Por lo tanto, comenzando la búsqueda del nodo destino n_d a partir del nodo K se obtiene el siguiente diagrama de árbol, con el nodo K como raíz.



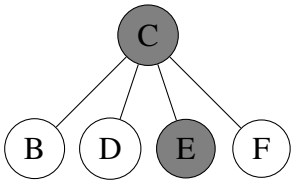
Al terminar todos los cálculos el algoritmo, el nodo K escoge la 3-tupla $K \rightarrow I \rightarrow G$ como la ruta ganadora, entonces la siguiente búsqueda comienza con el nodo G como raíz.



Después de la segunda iteración, la 3-tupla ganadora del consenso es $G \rightarrow J \rightarrow H$, para la siguiente iteración el nodo A será la raíz del árbol.



En la tercera iteración, la 3-tupla ganadora del consenso es $H \rightarrow F \rightarrow C$, para la siguiente iteración el nodo C será la raíz del árbol.



En la siguiente iteración se encuentra al nodo $n_d = E$ como vecino inmediato y no se continua buscando al siguiente nivel, no se hacen cálculos ni consenso. La ruta completa y exitosa es $K \rightarrow I \rightarrow G \rightarrow G \rightarrow J \rightarrow H \rightarrow C \rightarrow E$

En la figura 5.8 se muestra esta ruta.

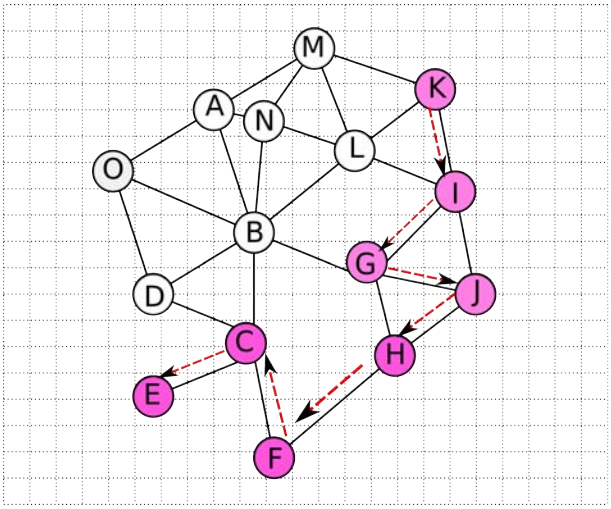


Figura 5.8: Ruta obtenida de la simulación simulación con la matriz de carga variable para búsqueda de uno y dos saltos.

5.3 Ejemplo general de la versión de una matriz con valores fijos

Este ejemplo se realizó con el fin de hacer una comparativa entre las versiones de los algoritmos así como la ventajas y desventajas que tienen cada uno. Se utilizó una matriz de adyacencia de 500 nodos, la matriz de carga y de tiempo disponible en el planificador del nodo tienen las mismas dimensiones que la de adyacencia. La matriz de adyacencia se obtiene de distribuir los nodos con una distribución normal en una malla de 500×500, cada nodo considera como vecinos a los nodos que se encuentren a 50 unidades a la redonda.

Utilizando estas 3 matrices se realizaron pruebas con los tres algoritmos en sus versiones con carga variable. Las tres versiones de los algoritmos buscaran desde el mismo origen al mismo destino, en este caso el nodo que inicia la búsqueda es el 85 y el nodo destino es el 234.

Los resultados de la versión de un salto con carga variable se muestran en la tabla 5.1.

La ruta resultante de el experimento de un salto es : $85 \rightarrow 298 \rightarrow 444 \rightarrow 380 \rightarrow 228 \rightarrow 291 \rightarrow 109 \rightarrow 351 \rightarrow 471 \rightarrow 185 \rightarrow 354 \rightarrow 102 \rightarrow 283 \rightarrow 483 \rightarrow 317 \rightarrow 152 \rightarrow 487 \rightarrow 383 \rightarrow 204 \rightarrow 96 \rightarrow 469 \rightarrow 31 \rightarrow 486 \rightarrow 260 \rightarrow 229 \rightarrow 234$, el costo total es 24.1948 y el tiempo de búsqueda fue de 5.98 s.

Origen	Destino	Costo	Tiempo total
85	298	0.97	
298	444	0.97	
444	380	0.96	
380	228	0.97	
228	291	0.97	
291	109	0.94	
109	351	0.93	
351	471	0.97	
471	185	0.95	
185	354	0.97	
354	102	0.97	
102	283	0.96	
283	483	0.96	
483	317	0.96	
317	152	0.97	
152	487	0.96	
487	383	0.96	
383	204	0.95	
204	96	0.96	
96	469	0.97	
469	31	0.97	
31	486	0.96	
486	260	0.96	
260	229	0.97	
229	234	0.99	
		24.19	5.98

Tabla 5.1: Resultados de la versión de un salto para encontrar la ruta entre el nodo 85 y el 234.

En el experimento realizado con la versión de dos saltos con carga variable, se obtuvo la ruta mostrada en la tabla 5.2.

Ruta			Costo	Tiempo total
85	495	234	.73	37.89

Tabla 5.2: Resultados de la versión de dos saltos para encontrar la ruta entre el nodo 85 y el 234.

En el experimento realizado con la versión de uno y dos saltos con carga variable, se obtuvo la ruta mostrada en la tabla 5.3.

Ruta			Costo	Tiempo total
85	495	234	.73	57.26

Tabla 5.3: Resultados de la versión de uno y dos saltos para encontrar la ruta entre el nodo 85 y el 234.

En las versiones de dos y uno y dos saltos con carga la ruta que se obtuvo es la misma, si embargo el costo total y el tiempo de cálculo difieren. El tiempo que toman estas dos versiones en encontrar la ruta es muy largo en comparación a la versión de un salto.

En el experimento de un salto, se realizaron 25 saltos hasta encontrar la ruta y el tiempo es más corto con respecto a los otros dos ejemplos, sin embargo el costo asociado es mucho mayor.

5.4 Simulación de la versión móvil.

En este caso, se crea para cada nodo una estructura con los siguientes datos:

1. ID.
2. La ubicación espacial en la malla (coordenadas x, y).
3. Tiempo disponible en su planificador de tareas (tiempo).
4. Lista de sus vecinos (conjunto J_i)en un rango de transmisión ρ .
5. Carga entre cada uno de sus enlaces con sus vecinos (carga).

6. Una bandera que indica si el nodo es móvil (flag).

El número de nodos para este ejemplo será 30, el $10\,\%$ de estos nodos serán móviles y se moverán de manera independiente de los cálculos del algoritmo de consenso. En este caso no se tiene una matriz global con la posición de los nodos, cada nodo conoce su posición en el espacio físico y determina que nodos son sus vecinos por la proximidad, el radio al que se considera que un nodo es vecino es de 10 unidades. El nodo que solicita enviar un mensaje crea una tabla de sus vecinos, que incluyen todos los datos de la estructura, el nodo solicitante hace los cálculos del algoritmo y decide que nodo esta apto para recibir el mensaje a la vez transmitirlo, en la siguiente Figura 5.9, el nodo rodeado por un un cuadrado es el nodo origen (nodo D) y el señalado con un rombo es el destino alcanzado (nodo F). Los nodos marcados con un asterisco ($*$) son los vecinos de todas las iteraciones.

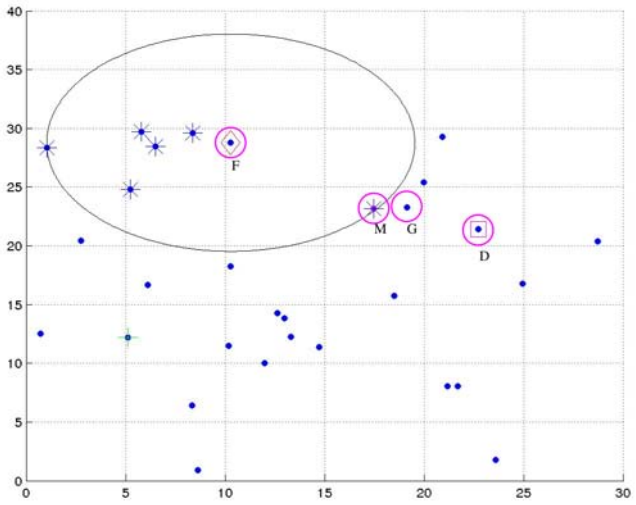


Figura 5.9: Ejemplo de la versión móvil.

El nodo se alcanzó en tres iteraciones del algoritmo, en la Tabla 5.4 se detalla que decisiones se tomaron en cada iteración:

Iteración	Origen	Ganador	Costo	Flag	Móvil	posición en el eje x	posición en el eje y
1	D	G	0.3679	0	0	19.1473	23.2600
2	G	M	0.3679	0	0	17.4729	23.1662
3	M	F	0.3679	1	0	10.2861	28.7814

Tabla 5.4: Resultados de la versión móvil del algoritmo.

Siendo la ruta final $D \rightarrow G \rightarrow M \rightarrow F$.

5.4.1 Algoritmo de inundación.

Como se ha mencionado anteriormente, la mayoría de los algoritmos de ruteo para redes Ad Hoc utilizan el "algoritmo de inundación" (inundación de red) [SK04], en mayor o menor medida como mecanismo de búsqueda de rutas. Consiste en un un algoritmo simple de encaminamiento en el cuál se envían todos los paquetes entrantes por cada interfaz de salida, excepto por la que se ha recibido. Debido a como funciona el algoritmo de encaminamiento se garantiza que un paquete es entregado (si éste puede ser entregado).

Este algoritmo de encaminamiento es muy fácil de implementar, aunque con es un enfoque de fuerza bruta e ineficiente, en el sentido de que se saturan los canales de comunicación.

Se aplica en las tramas de descubrimiento, en los puentes de red (*bridges*) por encaminamiento desde el origen y los transparentes, cuando la dirección de destino es desconocida [PSJ99].

Usenet y P2P (peer-to-peer) utilizan las inundaciones, así como los protocolos de encaminamiento como OSPF (*Open Shortest Path First*), DVMRP (*Distance Vector Multicast Routing Protocol*) y redes ad-hoc inalámbricas [PSJ99].

Debido a que los paquetes son enviados a través de cada enlace de salida se desaprovecha el ancho de banda. La inundación puede aumentar el tráfico exponencialmente, sobre todo sí existen ciclos en la topología, ya que en ese caso se envían paquetes duplicados.

Es posible prevenir esto, limitando el número de saltos o en el tiempo de vida, ya que las copias duplicadas podrán circular dentro de la red sin parar. Otra

posibilidad es identificar los paquetes, por ejemplo numerándolos, para que cada router mantenga una lista de los paquetes enviados y así puede evitar reenviarlos de nuevo, asegurándose de que un paquete pasa a través de él una sola vez. También puede usarse " inundación " selectivo en el que el paquete se envía sólo por las líneas que aproximadamente apuntan en la dirección correcta [PSJ99].

El algoritmo de inundación tiene las siguientes ventajas. En primer lugar, no es necesario conocer el estado del enlace, las distribuciones de información, ni realizar cálculos complejos de ruta, por lo tanto se reduce los gastos generales de funcionamiento y complejidad de la aplicación.

En segundo lugar, los routers no están obligados a mantener la base de datos de la información de estado del enlace, con el consiguiente ahorro espacio de almacenamiento.

En tercer lugar, se acorta la ruta de búsqueda la conexión tiempo de configuración y puede seleccionar la mejor ruta disponible basada en múltiples criterios de calidad de servicio. Desde la última, y por lo tanto a más precisa, la información del estado del enlace se utiliza para la determinación de una nueva conexión, este puede encontrar una ruta cualificada [PSJ99].

El algoritmo de ruteo por consenso se comparó con el algoritmo de " inundación ". Al implementar el algoritmo de " inundación " se limitó a cuatro saltos, a continuación se muestran los resultados obtenidos Se utilizó una matriz de adyacencia de 500 nodos, la matriz de carga y de tiempo disponible en el planificador del nodo tienen las mismas dimensiones que la de adyacencia. La matriz de adyacencia se obtiene de distribuir los nodos con una distribución normal en una malla de 500×500, cada nodo considera como vecinos a 25 unidades a la redonda, se utilizo el nodo origen (nodo 81) y nodo destino (nodo 75). Los resultados del algoritmo de "inundación" se muestran en la tabla 5.5.

Origen	nodo intermedio	nodo intermedio	Destino
81	77	86	75
81	77	157	75
81	77	181	75
81	77	265	75
81	77	291	75
81	77	358	75
81	77	481	75
81	102	265	75
81	170	86	75
81	170	157	75
81	170	265	75
81	170	341	75
81	170	492	75
81	231	86	75
81	231	157	75
81	231	181	75
81	231	265	75
81	231	291	75
81	231	341	75
81	231	358	75
81	231	481	75
81	231	492	75
81	431	86	75
81	431	157	75
81	431	265	75
81	431	291	75
81	431	358	75
81	431	481	75
81	431	492	75
81	484	86	75
81	484	157	75
81	484	265	75
81	484	291	75

Tabla 5.5: Resultados de la implementación del algoritmo de “inundación ”, rutas obtenidas entre el nodo 81 y el 75.

Aunque se obtuvieron 34 rutas distintas para el par de nodos, por la naturaleza del algoritmo, no se toma en cuenta ninguna condición que nos permita saber que tan costosa es cada ruta, en términos de carga u otro factor cuantificable, además de haberse utilizado 68 enlaces de comunicación para obtener la tabla de rutas posibles.

5.4.2 Comparativa con el algoritmo de inundación.

Se realizaron una comparativa de *AEC* contra el algoritmo clásico de inundación, las condiciones experimentales fueron las siguientes:

1. Se utilizó un sistema distribuido móvil de 100 nodos.

2. Se considera como vecinos a los nodos que se encuentren a 10 unidades a la redonda.

3. Se realizaron pruebas con *AEC* su versión de 1 y 2 salto con topología móvil contra el algoritmo de inundación, utilizando como medida de comparación la función de disponibilidad.

4. Se muestran 2 ejemplos, donde ambos algoritmos buscan el mismo destino desde el mismo nodo de inicio.

Prueba 1. Inundación vs *AEC*.

En la Figuras 5.10, se muestra el resultado de ejecutar en la misma topología inicial al algoritmo de inundación y a *AEC*, respectivamente. En la Figura el inciso a) es el resultado de la búsqueda del algoritmo de inundación de una ruta entre el nodo 25 y el 32, se muestra la ruta calculada por el algoritmo de inundación, el cual no encuentra el camino al nodo destino. En el siguiente inciso se muestra el resultado del cálculo de ruta que obtuvo *AEC*.

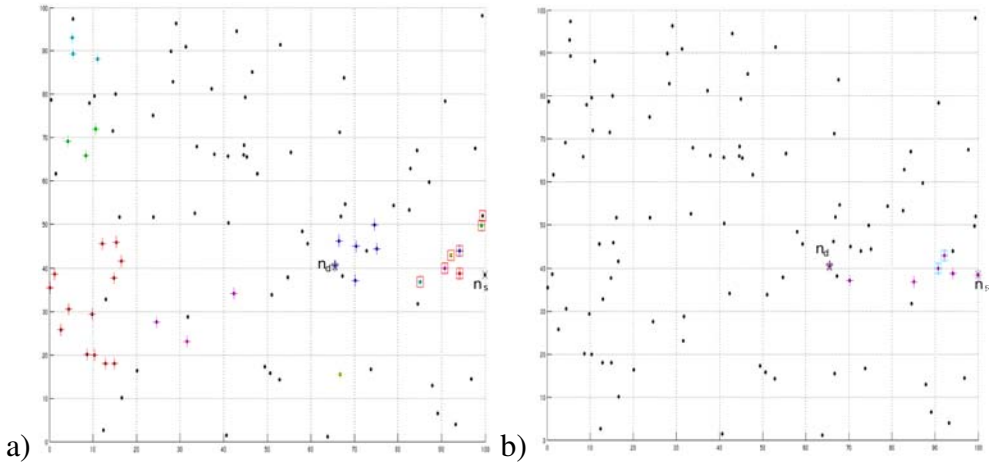


Figura 5.10: Comparativa de las rutas obtenidas por el algoritmo de a) Inundación y b) *AEC*, para la ruta $(25, 32)$.

El resultado de esta prueba se muestra en la tabla 5.6 , donde se nota que el costo (evaluado por la función de disponibilidad) y la distancia [1] son menores para *AEC*.

	Costo	**Distancia**
Inundación	15.08	401.17
AEC	5.15	36.46

Tabla 5.6: Resultados de la Prueba 1.

Prueba 2. Inundación vs *AEC*.

En lasFiguras 5.11, el inciso a) es el resultado de la búsqueda del algoritmo de inundación de una ruta entre el nodo 11 y el 31, el inciso b) es el resultado de la búsqueda de *AEC*.

[1] La distancia es el espacio recorrido en el plano entre slato y salto

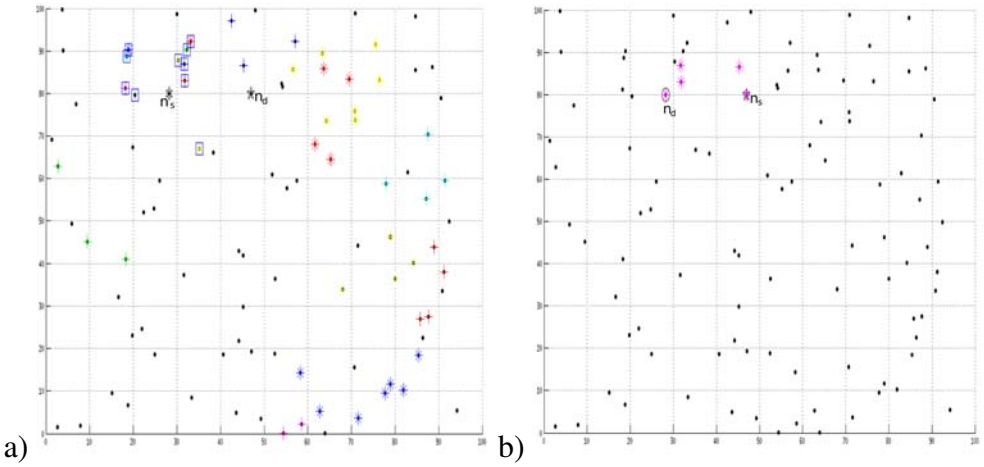


Figura 5.11: Comparativa de las rutas obtenidas por el algoritmo de a) Inundación y b) *AEC*, para la ruta $(11, 31)$, el algoritmo de inundación no encuentra el destino.

El resultado de esta prueba se muestra en la tabla 5.7, donde se nota que el costo (evaluado por la función de disponibilidad) y la distancia son menores para *AEC*, además que por la movilidad el algoritmo de inundación no encontró la ruta antes de *AEC*.

	Costo	**Distancia**
Inundación	20.97	559.21
AEC	1.47	15.16

Tabla 5.7: Resultados de la Prueba 2.

La complejidad de cálculo de *AEC* es considerablemente más grande que la inundación, que solo retransmite, por lo que aunque es más lento para tomar un decisión de la siguiente fuente del consenso. Sin embargo, escoge caminos con costos y distancias menores, disminuyendo la propagación de mensajes y ocupando menos recursos del SDM.

5.5 Experimentación en hardware: Arduino ™Yun

Para propósito de experimentación se contó con una infraestructura de 15 dispositivos Arduino ™Yun (Figura 5.12). El Arduino ™Yun es una placa MIPS basado en

el Atheros AR9331 y el ATmega32u4. El procesador Atheros soporta Linino, una distribución de Linux basada en OpenWRT. La placa incorpora Ethernet, 802.11 g/b/ soporte WiFi, puerto USB-A, ranura para tarjeta micro-SD, 20 entradas digitales / pines de salida (de los cuales 7 se pueden utilizar como salidas PWM y 12 como entradas analógicas), un oscilador de cristal de 16 MHz, una conexión micro USB, una cabecera ICSP, y 3 botones de reinicio.

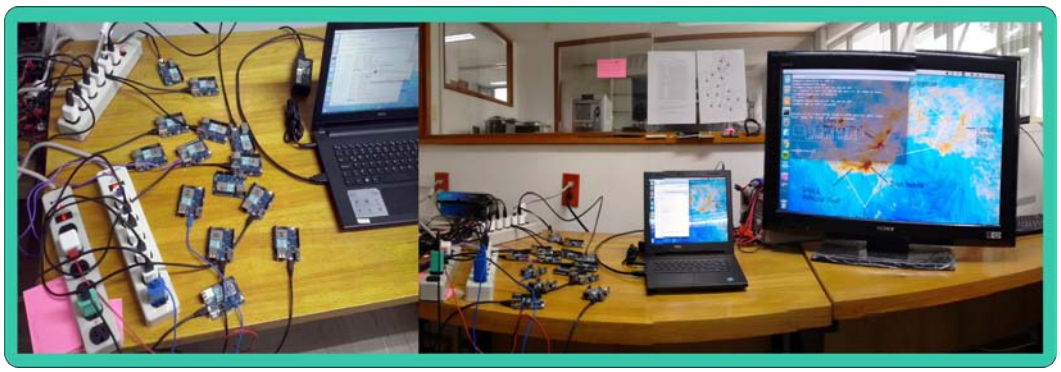


Figura 5.12: A la izquierda los 15 Arduino ™Yun y a la derecha el esquema completo. La pantalla está conectada a la laptop y de esta se controlan todos los arduinos, via ssh.

LininoOS contiene aproximadamente 3000 paquete incorporados y disponibles. Entre estos paquetes se encuentra el interprete de Python, esto fu aprovechado para desarrollar bibliotecas para el funcionamiento de *AEC*. Se desarrollaron las siguientes bibliotecas:

Comunicación: Se desarrolló una biblioteca para el manejo de la comunicación entre dispositivos, basada en la biblioteca de Python ServerSocket. Esta biblioteca contiene los módulos necesarios para enviar, recibir y solicitar información, para que cada nodo pueda ejecutar los 2 casos de uso descritos anteriormente. Adicionalmente existen módulos responsables de obtener la información local de cada nodo y buscar el vecindario inmediato. Para evitar la inundación existe una función que limita la propagación de mensajes por la red a 2 saltos de distancia del nodo activo. Cada nodo opera independientemente, en un estado base en el que permanentemente se ejecuta un servidor de sockets, esperando a que otro nodo le solicite información.

Búsqueda: Cuando un nodo comienza el procesos de búsqueda de otro nodo, primero lo busca en su vecindario inmediato, si no lo encuentra, solicita infor-

mación (tiempo disponible en su planificador y carga en el enlace de comunicación) a su vecinos y al mismo tiempo activa al vecindario para que busquen al nodo destino en sus propios vecindarios.

Cálculo: También se desarrolló una biblioteca encargada de realizar los cálculos de consenso utilizados por *AEC*.

Existen limitaciones técnicas, como el no tener baterías para permitir la movilidad de los dispositivos, por lo tanto se diseñó una topología para los dispositivos fijando artificialmente el vecindario de cada nodo. La topología del SDM se muestra en la Figura 5.13.

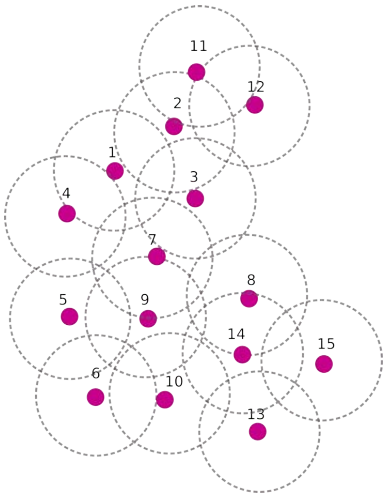


Figura 5.13: Topología experimental para los dispositivos Arduino Yun.

Usando la topología mostrada en la Figura 5.13 se describe el proceso de búsqueda de un objetivo:

- La búsqueda comienza cuando el nodo 4 (nodo fuente) comienza la búsqueda del dispositivo 3.

- El nodo 4 buscará en su vecindario más cercano, al no encontrar una ruta directa con el nodo 3 (Figura 5.14 a)), entonces el algoritmo *AEC* se ejecutará para calcular el próximo paso de la ruta(Figura 5.14 b)).

- Existen dos nodos que pueden ganar su propio consenso y establecer comunicación con el nodo 3 (esto dependerá de los valores locales de cada nodo), en este caso los nodos 1 y 7 ganan su propio consenso y tramiten el mensaje al nodo 3 y lo activan (Figura 5.14 c)). Cabe destacar que un nodo que no gane su propio consenso se desactivará y volverá a su estado base, como es el caso del nodo 4 y el nodo 5.
- Dado que no existe más comunicación que las solicitudes de la información local de los nodos, no hay manera de informar a los nodos que hay más de un ganador, por lo que el mensaje será recibido por el nodo 3 desde dos orígenes distintos.
- El objetivo ha sido alcanzado.

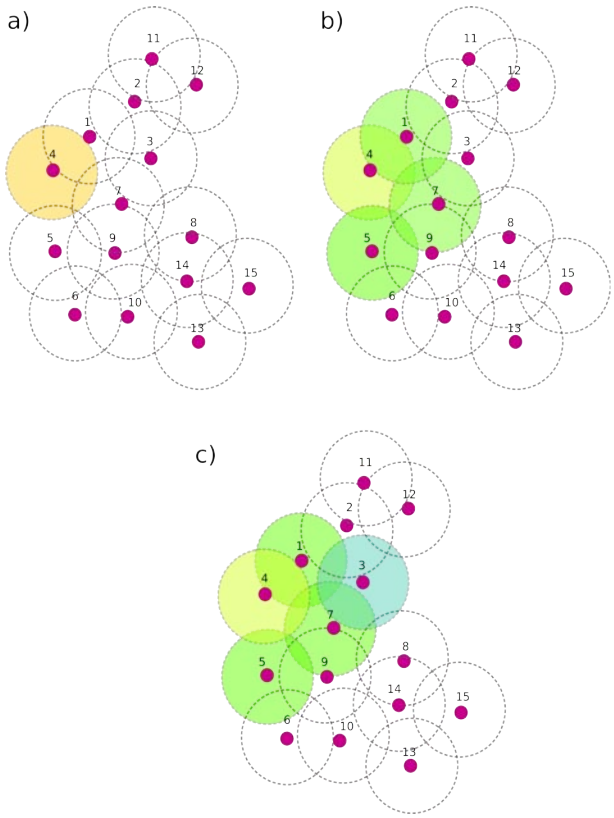


Figura 5.14: Secuencia de búsqueda del nodo 3 desde el nodo 4 utilizando el algoritmo *AEC*.

El tiempo total (costo) en que el nodo 3 encontró al nodo 4 fue de $0{,}31$ s. También se midió el tiempo transcurrido en cada nodo para la obtención de la información el nodo (Figura 5.15) , la búsqueda del nodo destino (Figura 5.16) y la activación del nodo ganador (Figura 5.17) se muestra en las siguientes gráficas. No se cuenta con un reloj global, los tiempos mostrados en las gráficas son locales por nodo.

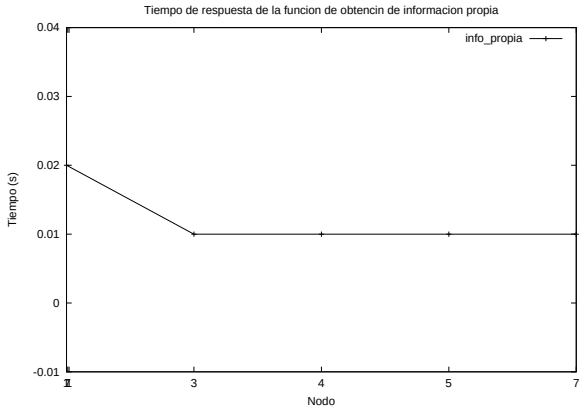


Figura 5.15: Retardo de la obtención de información propia.

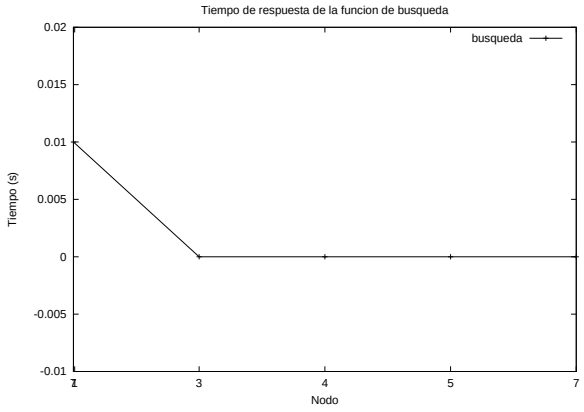


Figura 5.16: Retardo de la búsqueda del nodo objetivo en el vecindario inmediato.

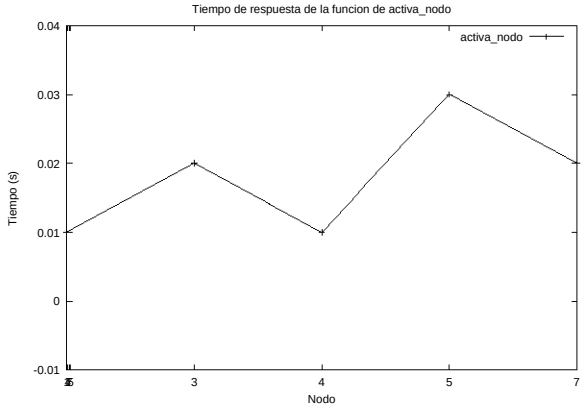


Figura 5.17: Retardo de la activación del nodo ganador.

Se realizó una segunda búsqueda usando el algoritmo *AEC*, en este caso el nodo fuente es el nodo 2 y el nodo objetivo es el nodo 9.

- En $a)$ de la Figura 5.18 se muestra la localización del nodo 2.
- El nodo 2 buscará en sus vecindario al nodo 9, dentro de este se encuentran los nodos 11, 12, 1 y 3.
- Al no encontrarse el nodo 9 en el vecindario inmediato, los nodos 2,11, 12, 1 y 3 iniciaran el *AEC* de manera independiente (b) de la Figura 5.18).
- El nodo 1 ha resultado ganador y comienza otra ronda de búsqueda en su vecindario (c) Figura 5.18), al no encontrarse el nodo 9, se ejecuta nuevamente el *AEC*.
- El nodo 7 tiene al 9 en su vecindario, por lo tanto se asume que el ganador de la ronda es 7 y este hace llegar el mensaje a 9 (d) Figura 5.18).

El tiempo total (costo) en que el nodo 9 encontró al nodo 2 fue de 0,39 s. El tiempo transcurrido en la obtención de la información propia (Figura 5.19), la búsqueda del nodo destino (Figura 5.20) y la activación del nodo ganador (Figura 5.21) se muestra en las siguientes gráficas.

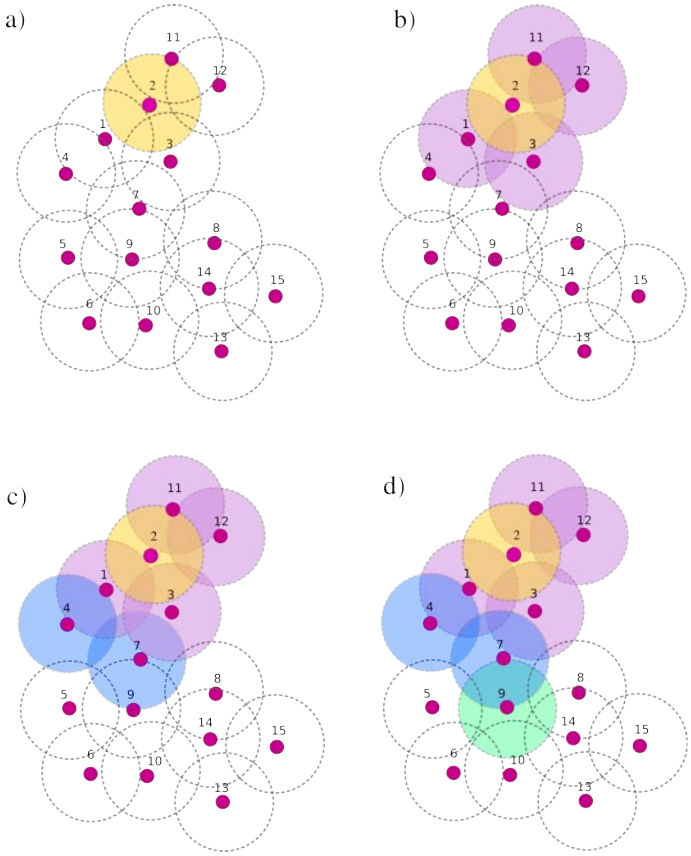


Figura 5.18: Secuencia de búsqueda del nodo 9 desde el nodo 2 utilizando el algoritmo *AEC*.

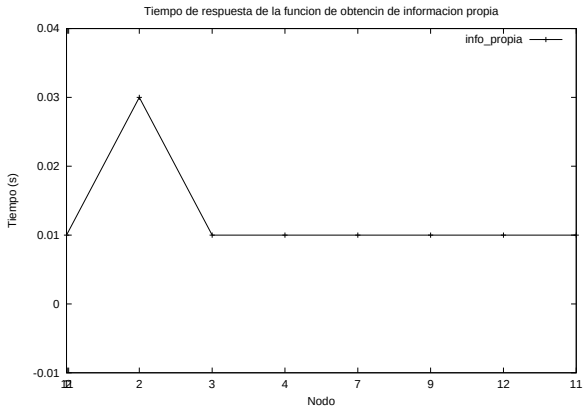


Figura 5.19: Retardo de la obtención de información propia.

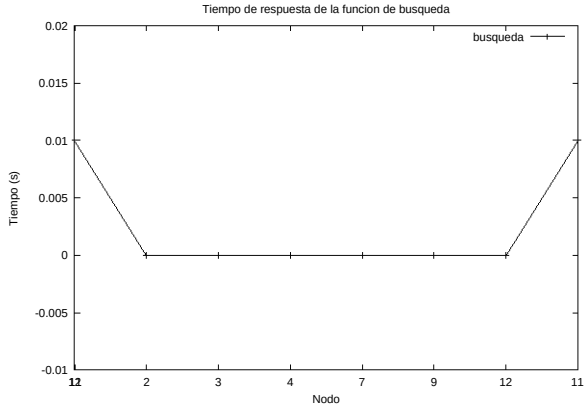


Figura 5.20: Retardo de la búsqueda del nodo objetivo en el vecindario inmediato.

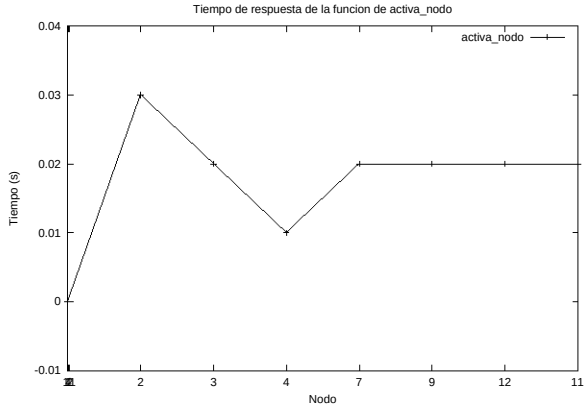


Figura 5.21: Retardo de la activación del nodo ganador.

5.6 Resumen

En este capítulo se presentaron los resultados de las simulaciones y experimentos descritos en el capítulo 4. En la comparativa se puede apreciar la reducción de mensajes de difusión del *AEC* al ser comparado contra el algoritmo de inundación. Así como el funcionamiento del *AEC* en un ambiente de dispositivos móviles reales.

Al final todo saldrá bien, y si no sale bien es que aún no es el final.

Sonny Kapur. El Exótico Hotel Marigold.

CAPÍTULO

6

Conclusiones

El objetivo del presente trabajo fue el diseño de un algoritmo de encaminamiento para un SDM basado en un algoritmo de consenso. Dicho algoritmo tomó como parámetros la carga del canal de comunicación entre nodos y el estado del planificador de tareas de cada nodo. En este capítulo se presentan las conclusiones sobre los resultados obtenidos en el capítulo anterior y el trabajo futuro que se desprende luego de analizar los resultados obtenidos.

De manera general se concluye que :

1. La ruta obtenida por el *AEC* es una secuencia de nodos con las mejores condiciones para transmitir, reduciendo la probabilidad de falla o ruptura de la ruta.

2. Al utilizar un vecindario acotado, el número de comunicaciones establecida se reduce en comparación del broadcast utilizado por el algoritmo de inundación.

3. Aunque la ruta obtenida por inundación es más corta y se obtiene en menor tiempo es más frágil que la calculada por *AEC*.

Esquema distribuido

El caso ideal para que el *AEC* funcione de la mejor forma, sería que la topología sea un grafo completo, es decir que se encuentre totalmente conectado, además de todos los nodos deben estar ubicados a poco distancia entre ellos, además de uniformemente distribuidos, aun así las orillas serían descartadas.

De otra forma el consenso descartaría a los segmentos del SDM que no se encuentren fuertemente conexos e incrementa la probabilidad de que se presenten ciclos de encaminamiento. El esquema distribuido presenta un dilema que es la formación de un bosque, como se menciona en el capítulo 4 sección 4.6.2. Se plantean 2 soluciones posibles para resolver este dilema:

1. Permitir que los nodos usen nodos intermedios para cumplir el requisito de estar conectados con todos los nodos participantes en el consenso. Esto implica que el costo del enlace se incremente, puesto que es necesario usar un nodo como intermediario, por lo que el costo de este enlace extra se suma al costo total; al incrementar el costo se reduce su probabilidad de ser electo durante el proceso de consenso.

2. Descartar a los nodos que no están fuertemente conexos. Esta opción implica descartar la mayoría de los nodos e incrementa la probabilidad de la formación de un bosque y la aparición de ciclos de encaminamiento.

Se optó por relajar la condición de que para que un nodo sea considerado para el consenso es necesario que sea fuertemente conectado con el resto; dependiendo del número de participantes se establecerá un mínimo número de enlaces hacia los nodos participantes y los faltantes serán suplidos utilizando un nodo intermediario (que incrementará el costo de esa conexión). De esta forma se incrementa el número de nodos participantes en el consenso y se reduce la probabilidad de la formación de un bosque y la aparición de ciclos de encaminamiento.

El algoritmo resultó ser eficiente con redes de menos de 100 nodos ya que para una mayor población del nodos el número de cálculos aumenta al tener vecindarios más poblados.

Logros

Con el diseño del algoritmo, simulaciones y experimentos realizados se logró reducir la inundación por medio de utilizar información local de cada nodo para el cálculo de cada paso de la ruta.

La movilidad de algunos nodos de la red influye en la dinámica de la transmisión de mensajes, ya que estos nodos móviles hacen posible romper ciclos de encaminamiento.

Los cambios en algunas secciones de la red no perturbaron la comunicación de la red en su totalidad.

Trabajo Futuro

El algoritmo propuesto (*AEC*) aportan robustez y resistencia ante los cambios frecuentes en la topología de un SDM, es necesario destacar que queda trabajo a ser realizado dentro del mismo, sea en características no desarrolladas o limitaciones del algoritmo actual, al momento de escribir el trabajo presente son:

1. La simulación del algoritmo en un simulador de redes como Opnet o Ns-3.

2. La implementación del algoritmo en otros dispositivos móviles.

APÉNDICE

A

Diagramas de Flujo

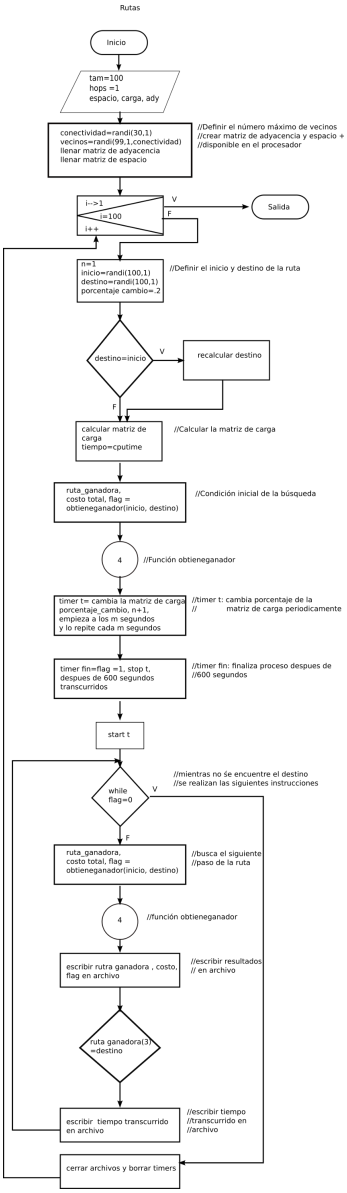


Figura A.1: Diagrama de flujo del programa principal. Este módulo manda a llamar el resto de las funciones.

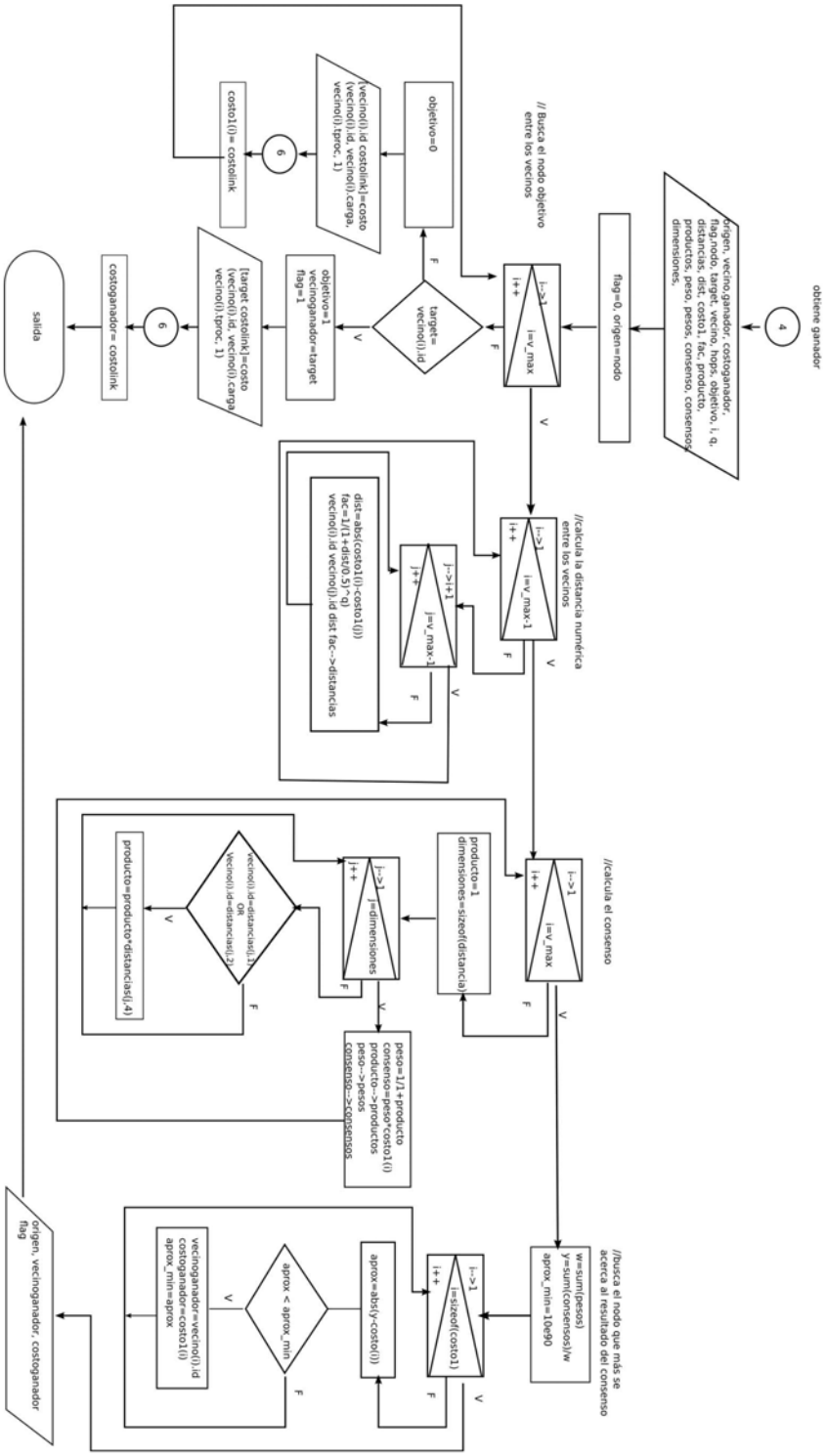


Figura A.2: Diagrama de flujo de la función obtiene_ganador donde se obtiene el resultado del proceso de consenso.

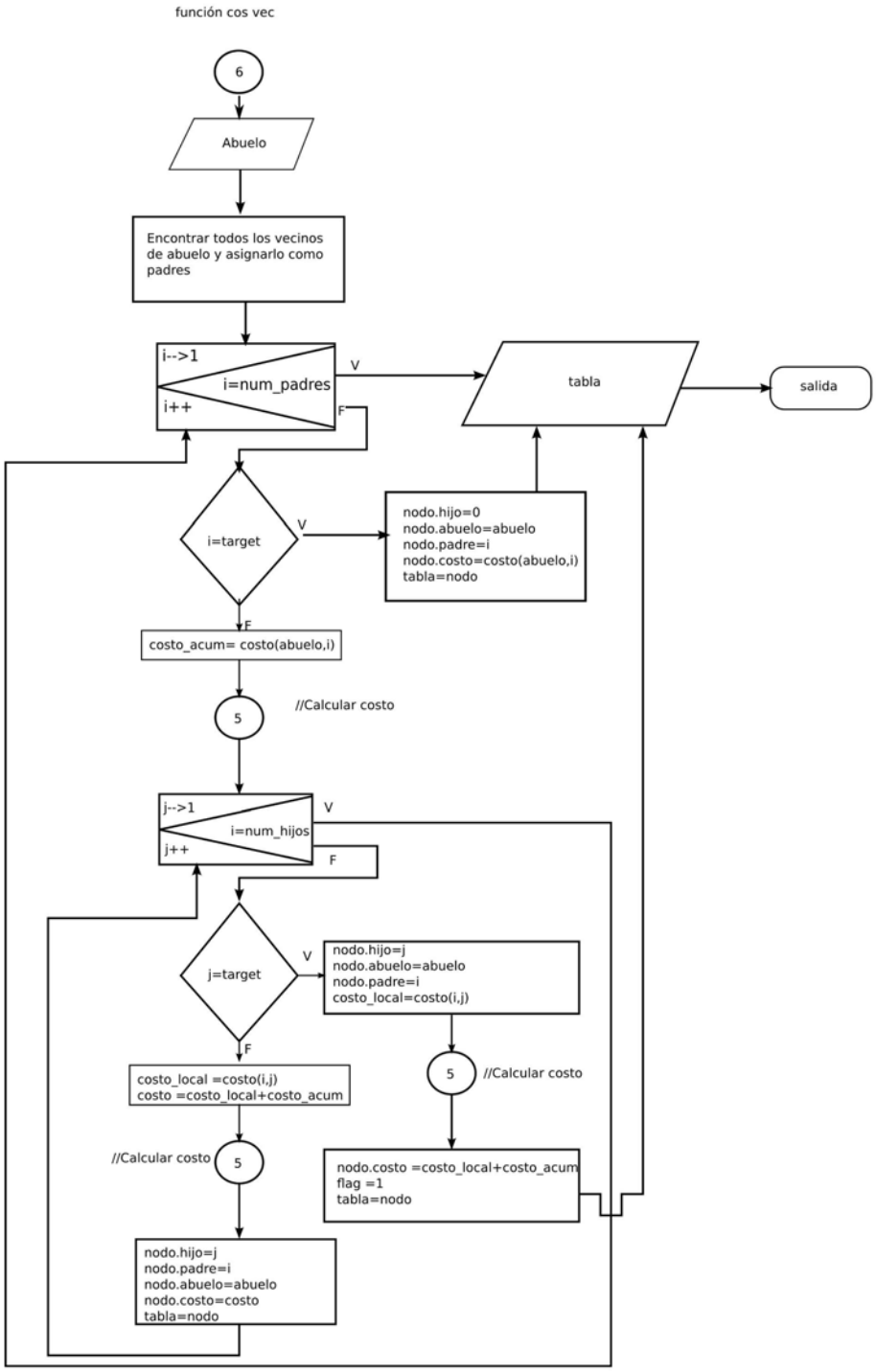


Figura A.3: Diagrama de flujo de la función cosvec donde se implementa la búsqueda de vecinos.

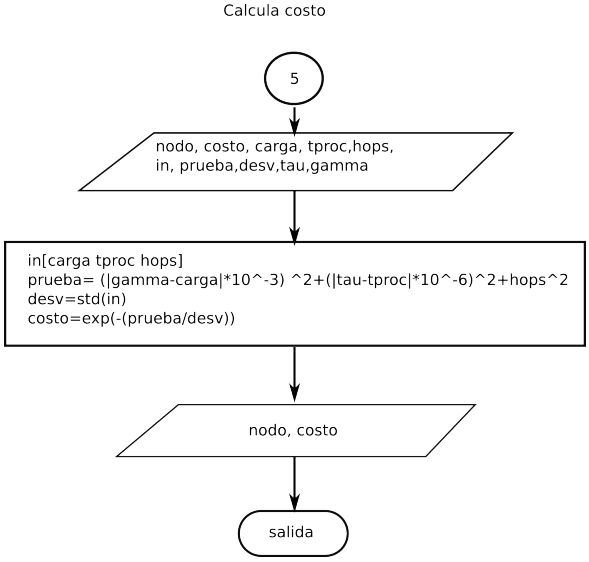


Figura A.4: Diagrama de flujo de la función costo, usada para calcular el costo de cada enlace.

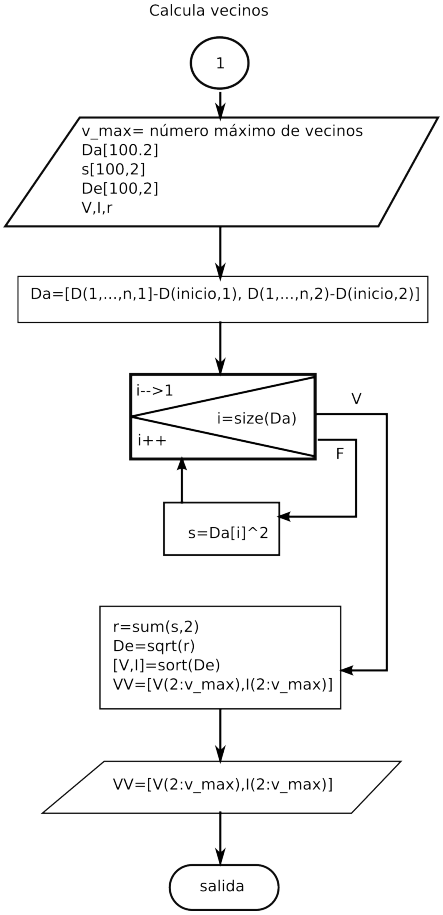


Figura A.5: Diagrama de flujo de la función calcula_vecinos.

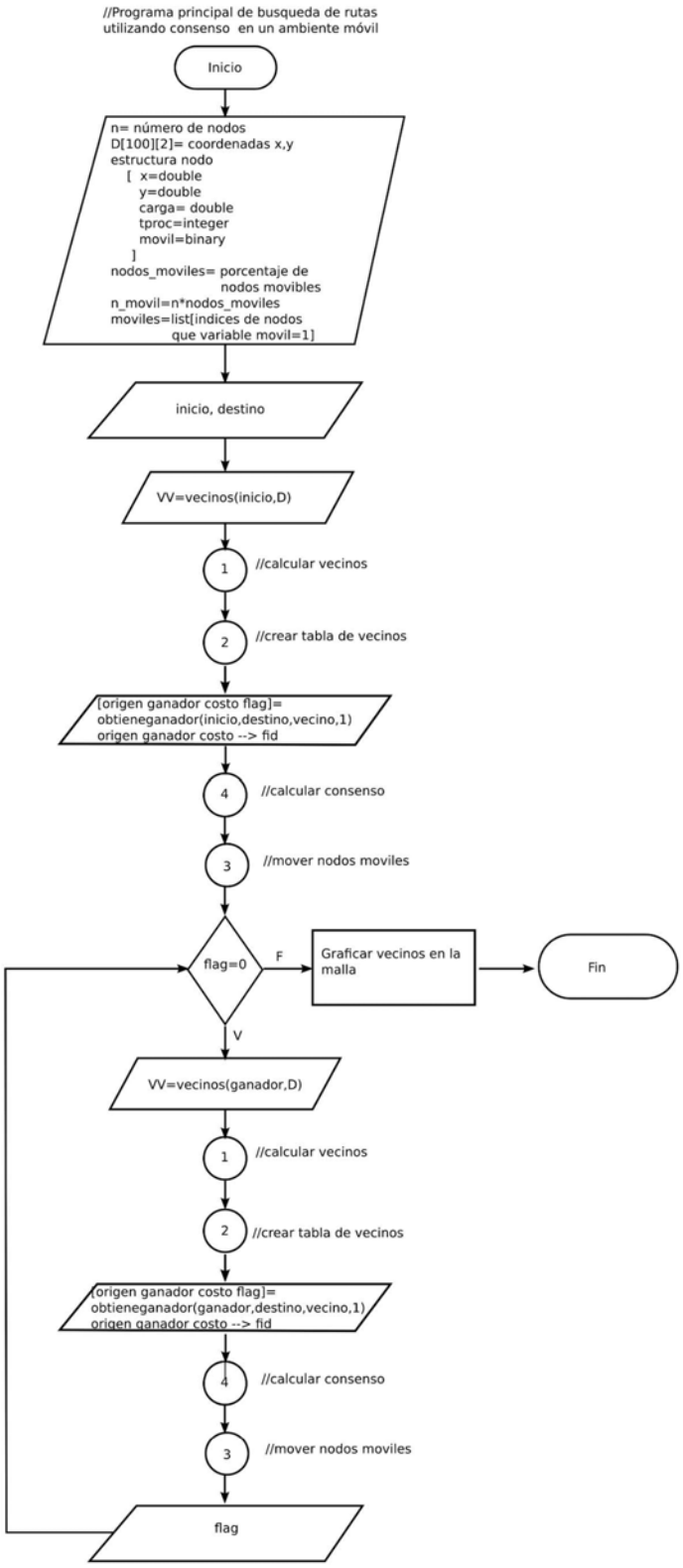


Figura A.6: Diagrama de flujo del programa principal de la versión móvil.

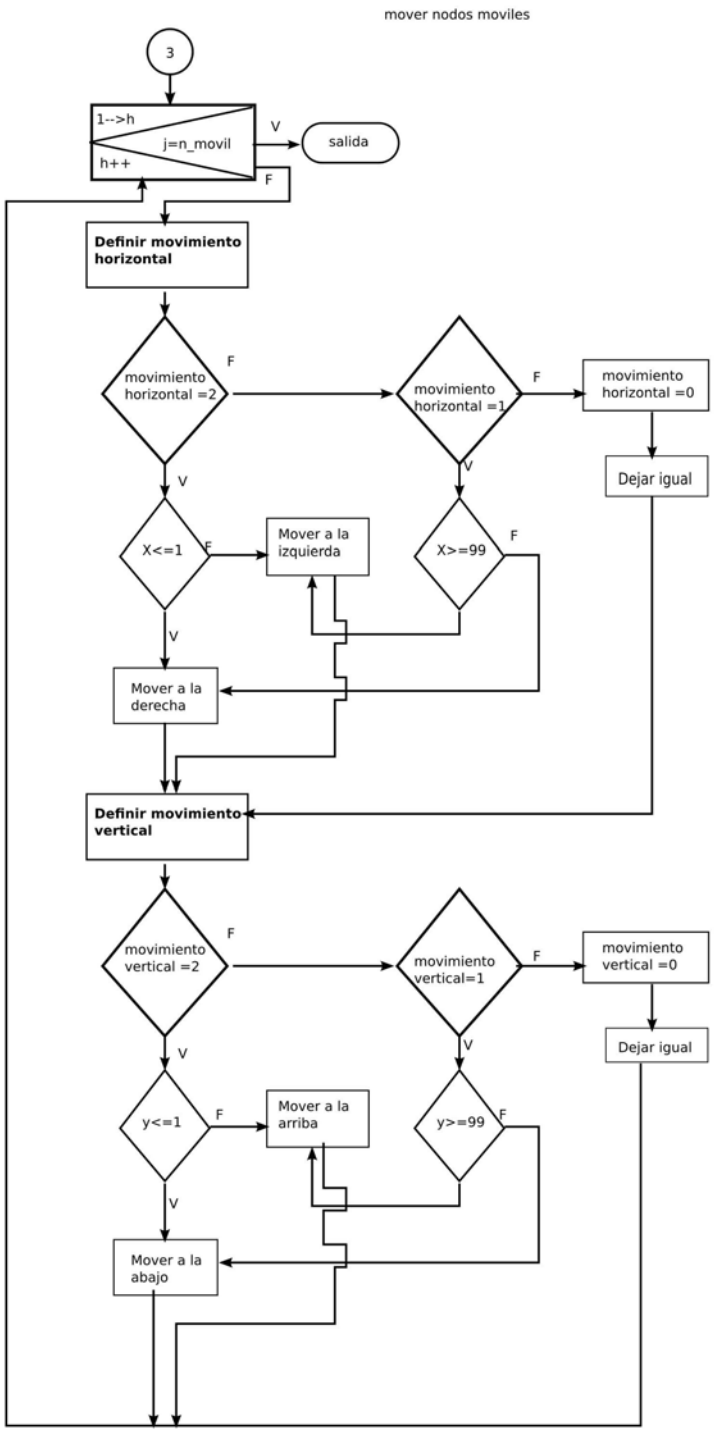


Figura A.7: Diagrama de flujo de la función mueve nodos, donde determina la dirección a la que se moverán los nodos móviles.

APÉNDICE

B

Ejemplos númericos

B.1 Simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto

En este primer ejemplo se muestra la ejecución completa del algoritmo y se detallan numéricamente las operaciones realizadas a cada paso del algoritmo.

En esta sección se mostrará numéricamente el funcionamiento del *AEC*.

Consideremos el grafo mostrado en la figura B.1 como el sistema distribuido con el que se desarrollará el ejemplo.

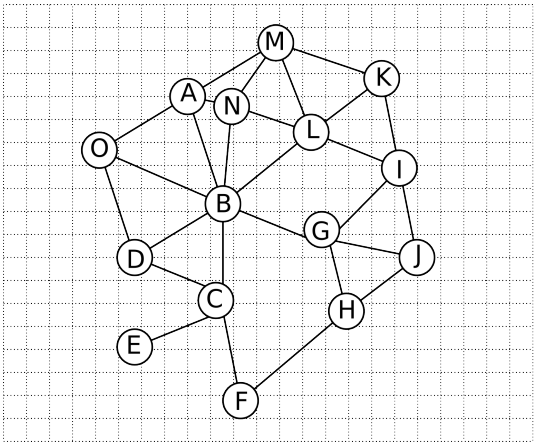


Figura B.1: Grafo utilizado en la simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto.

Sea α la matriz de adyacencia que representa este grafo.

$$\alpha = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.1}$$

Sea λ la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea γ el tiempo máximo que el planificador puede asignar a una tarea.

$$\lambda = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 8 \\ 5 & 0 & 10 & 2 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 17 & 0 & 4 & 1 \\ 0 & 10 & 0 & 14 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 11 & 5 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 11 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 10 & 18 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 9 & 8 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 0 & 16 & 11 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 16 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 3 & 0 & 0 \\ 8 & 1 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.2}$$

Sea β la matriz de tiempo disponible en el planificador de tareas de cada nodo

que representa este grafo. Sea τ la capacidad del canal de comunicación.

$$\beta = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 875 & 355 & 103 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 875 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 355 & 0 & 0 & 108 & 18 & 92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 103 & 61 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 254 & 416 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 92 & 0 & 254 & 0 & 266 & 101 & 0 \\ 647 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 416 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 101 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.3}$$

Consideremos para este ejemplo en particular que el nodo que inicia la búsqueda es el nodo K, entonces $n_s = K$ y el nodo destino es E, i.e. $n_d = E$. Se definen los valores de $\gamma = 20kbps$ y $\tau = 1000\mu s$.

El conjunto $J_K = \{I, L, M\}$, como el nodo E no esta en el conjunto de los vecinos inmediatos del nodo K entonces se calculan los valores de las función de disponibilidad de los pares de entradas:

$$\begin{aligned} f_{(K,I)} &= e^{-\left(\frac{(\lambda(K,I)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(K,I))^2}{\sigma_3^2}\right)} = ,3678 \\ f_{(K,L)} &= e^{-\left(\frac{(\lambda(K,L)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(K,L))^2}{\sigma_3^2}\right)} = ,3678 \\ f_{(K,M)} &= e^{-\left(\frac{(\lambda(K,M)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(K,M))^2}{\sigma_3^2}\right)} = ,3678. \end{aligned}$$

Posteriormente se calculan las distancias numéricas entre los pares de entradas:

$$\begin{aligned} d_{I,L} &= |f_{(K,I)} - f_{(K,L)}| = 0 \\ d_{I,M} &= |f_{(K,I)} - f_{(K,M)}| = 4{,}7084E-013 \\ d_{L,M} &= |f_{(K,L)} - f_{(K,M)}| = 4{,}7084E-013. \end{aligned}$$

Después se calcula el nivel de acuerdo entre los pares de entradas, se establecen lo valores de $p = ,5$ y $q = 2$.

$$\begin{aligned} s_{K(I,L)} &= \frac{1}{1+pd_{I,L}^q} = 1{,}00E+000 \\ s_{K(I,M)} &= \frac{1}{1+pd_{I,M}^q} = 1{,}00E+000 \\ s_{K(L,M)} &= \frac{1}{1+pd_{L,M}^q} = 1{,}00E+000. \end{aligned}$$

Se asigna un peso a cada entrada

$$w_{(K,I)} = \frac{1}{1 + (s_{K(I,L)} \quad s_{K(I,M)})} = ,5$$
$$w_{(K,L)} = \frac{1}{1 + (s_{K(I,L)} \quad s_{K(L,M)})} = ,5$$
$$w_{(L,M)} = \frac{1}{1 + (s_{K(I,M)} \quad s_{K(L,M)})} = ,5$$

Se obtienen los valores propuestos para el consenso para cada entrada:

$$y_K^1 = \frac{(w_{(K,I)} \quad f_{(K,I)}) + (w_{(I,L)} \quad f_{(I,L)}) + (w_{(L,M)} \quad f_{(L,M)})}{w_{(K,I)} + w_{(I,L)} + w_{(L,M)}} = 0{,}4905.$$

Entonces, comparando este valor con los valores de entrada el nodo ganador será :

$$\text{mín}(|y_K^1 - f_{(K,I)}|, |y_K^1 - f_{(K,L)}|, |y_K^1 - f_{(K,M)}|) = 0{,}1226.$$

este valor numérico esta asociado a la entrada $F_{K,M}$, por lo tanto el nodo ganador del consenso $G_K^1 = M$.

A continuación se mostraran solo los cálculos de los siguientes pasos hasta alcanzar el nodo objetivo.

$$k = 2$$
$$n_s = M$$
$$n_d = E$$
$$J_M = A, K, L, N$$
$$\gamma = 20$$
$$\tau = 1000$$
$$f_{M,A} = e^{-\left(\frac{(\lambda(M,A)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(M,A))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{M,K} = e^{-\left(\frac{(\lambda(M,K)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(M,K))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{M,L} = e^{-\left(\frac{(\lambda(M,L)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(M,L))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{M,A} = e^{-\left(\frac{(\lambda(M,A)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(M,A))^2}{\sigma_3^2}\right)} = 0{,}3678$$

$$
\begin{aligned}
d_{A,K} &= |f_{(M,A)} - f_{(M,K)}| = 4{,}8538 - 013 \\
d_{A,L} &= |f_{(M,A)} - f_{(M,L)}| = 4{,}8538 - 013 \\
d_{A,N} &= |f_{(M,A)} - f_{(M,N)}| = 4{,}8538 - 013 \\
d_{K,L} &= |f_{(M,K)} - f_{(M,L)}| = 0 \\
d_{K,N} &= |f_{(M,K)} - f_{(M,N)}| = 0 \\
d_{L,N} &= |f_{(M,L)} - f_{(M,N)}| = 0
\end{aligned}
$$

$$
\begin{aligned}
s_{M(A,K)} &= \frac{1}{1 + pd_{A,K}^{q}} = 1{,}00E + 000 \\
s_{M(A,L)} &= \frac{1}{1 + pd_{A,L}^{q}} = 1{,}00E + 000 \\
s_{M(A,N)} &= \frac{1}{1 + pd_{A,N}^{q}} = 1{,}00E + 000 \\
s_{M(K,L)} &= \frac{1}{1 + pd_{K,L}^{q}} = 1{,}00E + 000 \\
s_{M(K,N)} &= \frac{1}{1 + pd_{K,N}^{q}} = 1{,}00E + 000 \\
s_{M(L,N)} &= \frac{1}{1 + pd_{L,N}^{q}} = 1{,}00E + 000
\end{aligned}
$$

$$
\begin{aligned}
w_{(M,A)} &= \frac{1}{1 + (s_{M(A,K)} \quad s_{M(A,L)} \quad s_{M(A,N)})} = ,5 \\
w_{(M,K)} &= \frac{1}{1 + (s_{M(A,K)} \quad s_{M(K,L)} \quad s_{M(K,N)})} = ,5 \\
w_{(M,L)} &= \frac{1}{1 + (s_{M(A,L)} \quad s_{M(K,L)} \quad s_{M(K,N)})} = ,5 \\
w_{(M,N)} &= \frac{1}{1 + (s_{M(A,N)} \quad s_{M(K,N)} \quad s_{M(L,N)})} = ,5
\end{aligned}
$$

$$
y_M^2 = \frac{(w_{(M,A)} \quad f_{(M,A)}) + (w_{(M,K)} \quad f_{(M,K)}) + (w_{(M,L)} \quad f_{(M,L)}) + (w_{(M,N)} \quad f_{(M,N)})}{w_{(M,A)} + w_{(M,K)} + w_{(M,L)} + w_{(M,N)}} = 0{,}64378.
$$

$$min(|y_M^2 - f_{(M,A)}|, |y_M^2 - f_{(M,K)}|, |y_M^2 - f_{(M,L)}|, |y_M^2 - f_{(M,N)}|) = 0{,}2759.$$

este valor numérico esta asociado a la entrada $F_{M,A}$, por lo tanto el nodo ganador del consenso $G_M^2 = A$.

Cuando existen valores numéricos iguales se elige al primero de ellos.

$$\begin{aligned}
k &= 3 \\
n_s &= A \\
n_d &= E \\
J_M &= B, M, N, O \\
\gamma &= 20 \\
\tau &= 1000 \\
f_{A,B} &= e^{-\left(\frac{(\lambda(A,B)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(A,B))^2}{\sigma_3^2}\right)} = 0{,}3678 \\
f_{A,M} &= e^{-\left(\frac{(\lambda(A,M)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(A,M))^2}{\sigma_3^2}\right)} = 0{,}3678 \\
f_{A,N} &= e^{-\left(\frac{(\lambda(A,N)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(A,N))^2}{\sigma_3^2}\right)} = 0{,}36787 \\
f_{A,O} &= e^{-\left(\frac{(\lambda(A,O)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(A,O))^2}{\sigma_3^2}\right)} = 0{,}3678
\end{aligned}$$

$$\begin{aligned}
d_{B,M} &= |f_{(A,B)} - f_{(A,M)}| = 2{,}6427E-012 \\
d_{B,N} &= |f_{(A,B)} - f_{(A,N)}| = 1{,}7231E-012 \\
d_{B,O} &= |f_{(A,B)} - f_{(A,O)}| = 2{,}6427E-012 \\
d_{M,N} &= |f_{(A,M)} - f_{(A,N)}| = 9{,}1965-013 \\
d_{M,O} &= |f_{(A,M)} - f_{(A,O)}| = 0 \\
d_{N,O} &= |f_{(A,N)} - f_{(A,O)}| = 9{,}1965-013
\end{aligned}$$

$$
\begin{aligned}
s_{A(B,M)} &= \frac{1}{1+pd^{q}_{B,M}} = 1{,}00E+000 \\
s_{A(B,N)} &= \frac{1}{1+pd^{q}_{B,N}} = 1{,}00E+000 \\
s_{A(B,O)} &= \frac{1}{1+pd^{q}_{B,O}} = 1{,}00E+000 \\
s_{A(M,N)} &= \frac{1}{1+pd^{q}_{M,N}} = 1{,}00E+000 \\
s_{A(M,O)} &= \frac{1}{1+pd^{q}_{M,O}} = 1{,}00E+000 \\
s_{A(N,O)} &= \frac{1}{1+pd^{q}_{N,O}} = 1{,}00E+000
\end{aligned}
$$

$$
\begin{aligned}
w_{(A,B)} &= \frac{1}{1+(s_{A(B,M)} \quad s_{A(B,N)} \quad s_{A(B,O)})} = ,5 \\
w_{(A,M)} &= \frac{1}{1+(s_{A(B,M)} \quad s_{A(M,N)} \quad s_{A(M,O)})} = ,5 \\
w_{(A,N)} &= \frac{1}{1+(s_{A(B,N)} \quad s_{A(M,N)} \quad s_{A(N,O)})} = ,5 \\
w_{(A,O)} &= \frac{1}{1+(s_{A(B,O)} \quad s_{A(M,O)} \quad s_{A(N,O)})} = ,5
\end{aligned}
$$

$$
y^{3}_{M} = \frac{(w_{(A,B)} \quad f_{(A,B)}) + (w_{(A,M)} \quad f_{(A,M)}) + (w_{(A,N)} \quad f_{(A,N)}) + (w_{(A,O)} \quad f_{(A,O)})}{w_{(A,B)} + w_{(A,M)} + w_{(A,N)+w_{A,O}}} = 0{,}6437.
$$

$$
min(|y^{3}_{M} - f_{(A,B)}|, |y^{3}_{M} - f_{(A,M)}|, |y^{3}_{M} - f_{(A,N)}|, |y^{3}_{M} - f_{(A,O)}|) = 0{,}2759.
$$

este valor numérico esta asociado a la entrada $F_{A,O}$, por lo tanto el nodo ganador del consenso $G^{3}_{M} = O$.

$$k = 4$$
$$n_s = O$$
$$n_d = E$$
$$J_M = A, B, D$$
$$\gamma = 20$$
$$\tau = 1000$$
$$f_{O,A} = e^{-\left(\frac{(\lambda(O,A)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(O,A))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{O,B} = e^{-\left(\frac{(\lambda(O,B)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(O,B))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{O,D} = e^{-\left(\frac{(\lambda(O,D)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(O,D))^2}{\sigma_3^2}\right)} = 0{,}3678$$

$$d_{A,B} = |f_{(O,A)} - f_{(O,B)}| = 4{,}7095E-013$$
$$d_{A,D} = |f_{(O,A)} - f_{(O,D)}| = 4{,}70951E-013$$
$$d_{B,D} = |f_{(O,B)} - f_{(O,D)}| = 0$$

$$s_{O(A,B)} = \frac{1}{1+pd_{A,B}^q} = 1{,}00E+000$$
$$s_{O(A,D)} = \frac{1}{1+pd_{A,D}^q} = 1{,}00E+000$$
$$s_{O(B,D)} = \frac{1}{1+pd_{B,D}^q} = 1{,}00E+000$$

$$w_{(O,A)} = \frac{1}{1+(s_{O(A,B)} \quad s_{O(A,D)}} = ,5$$
$$w_{(O,B)} = \frac{1}{1+(s_{O(A,B)} \quad s_{O(B,D)}} = ,5$$
$$w_{(O,D)} = \frac{1}{1+(s_{O(A,D)} \quad s_{O(B,D)}} = ,5$$

$$y_M^4 = \frac{(w_{(O,A)} \quad f_{(O,A)}) + (w_{(O,B)} \quad f_{(O,B)}) + (w_{(O,D)} \quad f_{(O,D)})}{w_{(O,A)} + w_{(O,B)} + w_{(O,D)}} = 0{,}4905.$$

$$min(|y_M^4 - f_{(O,A)}|, |y_M^4 - f_{(O,B)}|, |y_M^4 - f_{(O,D)}|) = 0{,}1226.$$

este valor numérico esta asociado a la entrada $f_{O,A}$, por lo tanto el nodo ganador del consenso $g_M^4 = A$.

En este ejemplo como la información de las matrices no varía el resultado cambiará en el transcurso del tiempo, por lo que se tiene un ciclo entre en nodo O y A. No se logró establecer una ruta y esta ruta trunca quedo de esta manera: $K \rightarrow g^1 \rightarrow g^2 \rightarrow g^3 \rightarrow g^4 = K \rightarrow M \rightarrow A \rightarrow O \rightarrow A$. En la figura B.2 se muestra esta ruta.

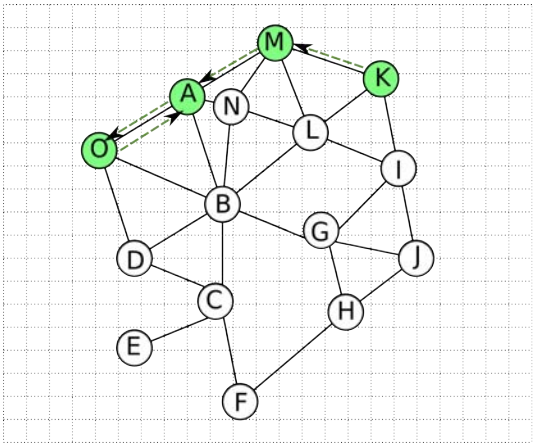


Figura B.2: Ruta obtenida en la simulación con una matriz de valores fijos para búsqueda utilizando un vecindario de un salto.

B.2 Simulación con una matriz de carga variable para búsqueda de un salto

Este primer ejemplo se presentan todos los cálculos realizados por el algoritmo de consenso de manera numérica.

Consideremos el grafo mostrado en la figura B.3 como el sistema distribuido con el que se desarrollará el ejemplo.

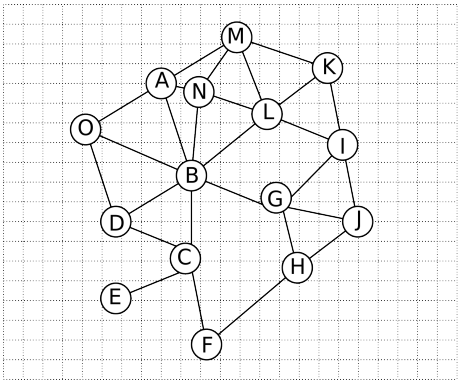


Figura B.3: Grafo utilizado en simulación con una matriz de carga variable para búsqueda de un salto.

Sea α la matriz de adyacencia que representa este grafo.

$$\alpha = H \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.4}$$

Sea λ la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea γ el tiempo máximo que el planificador puede asignar a una tarea.

$$\lambda = H \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 8 \\ 5 & 0 & 10 & 2 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 17 & 0 & 4 & 1 \\ 0 & 10 & 0 & 14 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 11 & 5 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 11 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 10 & 18 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 9 & 8 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 0 & 16 & 11 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 16 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 3 & 0 & 0 \\ 8 & 1 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.5}$$

Sea β la matriz de tiempo disponible en el planificador de tareas de cada nodo que representa este grafo. Sea τ la capacidad del canal de comunicación.

Consideremos para este ejemplo en particular que el nodo que inicia la búsqueda es el nodo K, entonces $n_s = K$ y el nodo destino es E, i.e. $n_d = E$. Se definen

los valores de $\gamma = 20$ y $\tau = 1000$. Se define β^1 como:

$$\beta^1 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 875 & 355 & 103 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 875 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 355 & 0 & 0 & 108 & 18 & 92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 103 & 61 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 254 & 416 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 92 & 0 & 254 & 0 & 266 & 101 & 0 \\ 647 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 416 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 101 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.6}$$

El conjunto $J_K = \{I, L, M\}$, como el nodo E no esta en el conjunto de los vecinos inmediatos del nodo K entonces se calculan los valores de las función de disponibilidad de los pares de entradas:

$$f_{(K,I)} = e^{-\left(\frac{(\lambda(K,I)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(K,I))^2}{\sigma_3^2}\right)} = ,3678$$
$$f_{(K,L)} = e^{-\left(\frac{(\lambda(K,L)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(K,L))^2}{\sigma_3^2}\right)} = ,3678$$
$$f_{(K,M)} = e^{-\left(\frac{(\lambda(K,M)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(K,M))^2}{\sigma_3^2}\right)} = ,3678$$

Posteriormente se calculan las distancias numéricas entre los pares de entradas

$$d_{I,L} = |f_{(K,I)} - f_{(K,L)}| = 0$$
$$d_{I,M} = |f_{(K,I)} - f_{(K,M)}| = 4{,}7084E - 013$$
$$d_{L,M} = |f_{(K,L)} - f_{(K,M)}| = 4{,}7084E - 013$$

Después se calcula el nivel de acuerdo entre los pares de entradas, se establecen lo valores de $p = ,5$ y $q = 2$.

$$s_{K(I,L)} = \frac{1}{1 + pd_{I,L}^q} = 1{,}00E + 000$$
$$s_{K(I,M)} = \frac{1}{1 + pd_{I,M}^q} = 1{,}00E + 000$$
$$s_{K(L,M)} = \frac{1}{1 + pd_{L,M}^q} = 1{,}00E + 000$$

Se asigna un peso a cada entrada

$$w_{(K,I)} = \frac{1}{1 + (s_{K(I,L)} \quad s_{K(I,M)})} = ,5$$
$$w_{(K,L)} = \frac{1}{1 + (s_{K(I,L)} \quad s_{K(L,M)})} = ,5$$
$$w_{(L,M)} = \frac{1}{1 + (s_{K(I,M)} \quad s_{K(L,M)})} = ,5$$

Se obtienen los valores propuestos para el consenso para cada entrada:

$$y_K^1 = \frac{(w_{(K,I)} \quad f_{(K,I)}) + (w_{(I,L)} \quad f_{(I,L)}) + (w_{(L,M)} \quad f_{(L,M)})}{w_{(K,I)} + w_{(I,L)} + w_{(L,M)}} = 0{,}4905.$$

Entonces, comparando este valor con los valores de entrada el nodo ganador será :

$$\text{mín}(|y_K^1 - f_{(K,I)}|, |y_K^1 - f_{(K,L)}|, |y_K^1 - f_{(K,M)}|) = 0{,}12262.$$

este valor numérico esta asociado a la entrada $F_{K,M}$, por lo tanto el nodo ganador del consenso $G_K^1 = M$

A continuación se mostraran solo los cálculos de los siguientes pasos hasta alcanzar el nodo objetivo.

$$\beta^{\mathbf{2}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 647 & 272 & 801 \\ 548 & 0 & 743 & 277 & 0 & 0 & 707 & 0 & 0 & 0 & 0 & 388 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 293 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 293 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 707 & 0 & 0 & 0 & 0 & 0 & 100 & 79 & 700 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 100 & 0 & 0 & 250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 79 & 0 & 0 & 108 & 550 & 505 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 700 & 250 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 550 & 0 & 0 & 700 & 788 & 0 & 0 \\ 0 & 388 & 0 & 0 & 0 & 0 & 0 & 0 & 505 & 0 & 700 & 0 & 266 & 408 & 0 \\ 467 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 788 & 266 & 0 & 39 & 0 \\ 272 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 408 & 39 & 0 & 0 \\ 801 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.7}$$

$$k = 2$$
$$n_s = M$$
$$n_d = E$$
$$J_M = A, K, L, N$$
$$\gamma = 20$$
$$\tau = 1000$$
$$f_{M,A} = e^{-\left(\frac{(\lambda(M,A)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(M,A))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{M,K} = e^{-\left(\frac{(\lambda(M,K)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(M,K))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{M,L} = e^{-\left(\frac{(\lambda(M,L)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(M,L))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{M,A} = e^{-\left(\frac{(\lambda(M,A)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(M,A))^2}{\sigma_3^2}\right)} = 0{,}3678$$

$$d_{A,K} = |f_{(M,A)} - f_{(M,K)}| = 4{,}8538E-013$$
$$d_{A,L} = |f_{(M,A)} - f_{(M,L)}| = 4{,}8538E-013$$
$$d_{A,N} = |f_{(M,A)} - f_{(M,N)}| = 4{,}8538E-013$$
$$d_{K,L} = |f_{(M,K)} - f_{(M,L)}| = 0$$
$$d_{K,N} = |f_{(M,K)} - f_{(M,N)}| = 0$$
$$d_{L,N} = |f_{(M,L)} - f_{(M,N)}| = 0$$

$$s_{M(A,K)} = \frac{1}{1 + pd^q_{A,K}} = 1{,}00E + 000$$
$$s_{M(A,L)} = \frac{1}{1 + pd^q_{A,L}} = 1{,}00E + 000$$
$$s_{M(A,N)} = \frac{1}{1 + pd^q_{A,N}} = 1{,}00E + 000$$
$$s_{M(K,L)} = \frac{1}{1 + pd^q_{K,L}} = 1{,}00E + 000$$
$$s_{M(K,N)} = \frac{1}{1 + pd^q_{K,N}} = 1{,}00E + 000$$
$$s_{M(L,N)} = \frac{1}{1 + pd^q_{L,N}} = 1{,}00E + 000$$

$$w_{(M,A)} = \frac{1}{1 + \left(s_{M(A,K)} \quad s_{M(A,L)} \quad s_{M(A,N)}\right)} = ,5$$
$$w_{(M,K)} = \frac{1}{1 + \left(s_{M(A,K)} \quad s_{M(K,L)} \quad s_{M(K,N)}\right)} = ,5$$
$$w_{(M,L)} = \frac{1}{1 + \left(s_{M(A,L)} \quad s_{M(K,L)} \quad s_{M(K,N)}\right)} = ,5$$
$$w_{(M,N)} = \frac{1}{1 + \left(s_{M(A,N)} \quad s_{M(K,N)} \quad s_{M(L,N)}\right)} = ,5$$

$$y^2_M = \frac{\left(w_{(M,A)} \quad f_{(M,A)}\right) + \left(w_{(M,K)} \quad f_{(M,K)}\right) + \left(w_{(M,L)} \quad f_{(M,L)}\right) + \left(w_{(M,N)} \quad f_{(M,N)}\right)}{w_{(M,A)} + w_{(M,K)} + w_{(M,L)} + w_{(M,N)}} = 0{,}6437.$$

$$\text{mín}(|y^2_M - f_{(M,A)}|, |y^2_M - f_{(M,K)}|, |y^2_M - f_{(M,L)}|, |y^2_M - f_{(M,N)}|) = 0{,}2759.$$

este valor numérico esta asociado a la entrada $F_{M,A}$, por lo tanto el nodo ganador del consenso $G^2_M = A$

$$\beta^{\mathbf{3}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 548 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 798 & 650 & 208 \\ 548 & 0 & 743 & 277 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 100 & 0 & 40 & 568 \\ 0 & 743 & 0 & 104 & 632 & 56 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 277 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 56 & 0 & 0 & 0 & 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 70 & 0 & 0 & 0 & 0 & 0 & 100 & 79 & 700 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 252 & 100 & 0 & 0 & 250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 79 & 0 & 0 & 108 & 304 & 63 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 700 & 250 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 304 & 0 & 0 & 305 & 788 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 63 & 0 & 305 & 0 & 369 & 874 & 0 \\ 798 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 788 & 369 & 0 & 39 & 0 \\ 650 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 874 & 39 & 0 & 0 \\ 208 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.8}$$

$$k = 3$$
$$n_s = A$$
$$n_d = E$$
$$J_M = B, M, N, O$$
$$\gamma = 20$$
$$\tau = 1000$$
$$f_{A,B} = e^{-\left(\frac{(\lambda(A,B)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(A,B))^2}{\sigma_3^2}\right)} = 0{,}3678.$$
$$f_{A,M} = e^{-\left(\frac{(\lambda(A,M)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(A,M))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{A,N} = e^{-\left(\frac{(\lambda(A,N)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(A,N))^2}{\sigma_3^2}\right)} = 0{,}3678$$
$$f_{A,O} = e^{-\left(\frac{(\lambda(A,O)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(A,O))^2}{\sigma_3^2}\right)} = 0{,}3678$$

$$d_{B,M} = |f_{(A,B)} - f_{(A,M)}| = 2{,}6427E-012$$
$$d_{B,N} = |f_{(A,B)} - f_{(A,N)}| = 1{,}3347E-012$$
$$d_{B,O} = |f_{(A,B)} - f_{(A,O)}| = 2{,}6427E-012$$
$$d_{M,N} = |f_{(A,M)} - f_{(A,N)}| = 1{,}3080E-012$$
$$d_{M,O} = |f_{(A,M)} - f_{(A,O)}| = 0$$
$$d_{N,O} = |f_{(A,N)} - f_{(A,O)}| = 1{,}3080E-012$$

$$s_{A(B,M)} = \frac{1}{1 + pd^q_{B,M}} = 1{,}00E + 000$$
$$s_{A(B,N)} = \frac{1}{1 + pd^q_{B,N}} = 1{,}00E + 000$$
$$s_{A(B,O)} = \frac{1}{1 + pd^q_{B,O}} = 1{,}00E + 000$$
$$s_{A(M,N)} = \frac{1}{1 + pd^q_{M,N}} = 1{,}00E + 000$$
$$s_{A(M,O)} = \frac{1}{1 + pd^q_{M,O}} = 1{,}00E + 000$$
$$s_{A(N,O)} = \frac{1}{1 + pd^q_{N,O}} = 1{,}00E + 000$$

$$w_{(A,B)} = \frac{1}{1 + \begin{pmatrix} s_{A(B,M)} & s_{A(B,N)} & s_{A(B,O)} \end{pmatrix}} = ,5$$
$$w_{(A,M)} = \frac{1}{1 + \begin{pmatrix} s_{A(B,M)} & s_{A(M,N)} & s_{A(M,O)} \end{pmatrix}} = ,5$$
$$w_{(A,N)} = \frac{1}{1 + \begin{pmatrix} s_{A(B,N)} & s_{A(M,N)} & s_{A(N,O)} \end{pmatrix}} = ,5$$
$$w_{(A,O)} = \frac{1}{1 + \begin{pmatrix} s_{A(B,O)} & s_{A(M,O)} & s_{A(N,O)} \end{pmatrix}} = ,5$$

$$y^3_M = \frac{\begin{pmatrix} w_{(A,B)} & f_{(A,B)} \end{pmatrix} + \begin{pmatrix} w_{(A,M)} & f_{(A,M)} \end{pmatrix} + \begin{pmatrix} w_{(A,N)} & f_{(A,N)} \end{pmatrix} + \begin{pmatrix} w_{(A,O)} & f_{(A,O)} \end{pmatrix}}{w_{(A,B)} + w_{(A,M)} + w_{(A,N)+w_{A,O}}} = 0{,}6437.$$

$$\text{mín}(|y^3_M - f_{(A,B)}|, |y^3_M - f_{(A,M)}|, |y^3_M - f_{(A,N)}|, |y^3_M - f_{(A,O)}|) = 0{,}2759.$$

este valor numérico esta asociado a la entrada $F_{A,O}$, por lo tanto el nodo ganador del consenso $G^3_A = O$

$$\beta^{\mathbf{4}} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 350 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 359 & 407 & 908 \\ 350 & 0 & 87 & 500 & 0 & 0 & 500 & 0 & 0 & 0 & 0 & 100 & 0 & 40 & 568 \\ 0 & 87 & 0 & 300 & 632 & 203 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 300 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 203 & 0 & 0 & 0 & 0 & 900 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 & 0 & 171 & 79 & 42 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 900 & 171 & 0 & 0 & 250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 79 & 0 & 0 & 108 & 304 & 250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 42 & 250 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 304 & 0 & 0 & 305 & 305 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 250 & 0 & 305 & 0 & 607 & 100 & 0 \\ 359 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 788 & 607 & 0 & 39 & 0 \\ 407 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 39 & 0 & 0 \\ 908 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.9}$$

$$
\begin{aligned}
k &= 4 \\
n_s &= O \\
n_d &= E \\
J_M &= A, B, D \\
\gamma &= 20 \\
\tau &= 1000 \\
f_{O,A} &= e^{-\left(\frac{(\lambda(O,A)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(O,A))^2}{\sigma_3^2}\right)} = 0{,}3678 \\
f_{O,B} &= e^{-\left(\frac{(\lambda(O,B)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(O,B))^2}{\sigma_3^2}\right)} = 0{,}3678 \\
f_{O,D} &= e^{-\left(\frac{(\lambda(O,D)-\Gamma)^2}{\sigma_1^2}+\frac{hops^2}{\sigma_2^2}+\frac{(\tau-\beta(O,D))^2}{\sigma_3^2}\right)} = 0{,}3678
\end{aligned}
$$

$$
\begin{aligned}
d_{A,B} &= |f_{(O,A)} - f_{(O,B)}| = 0 \\
d_{A,D} &= |f_{(O,A)} - f_{(O,D)}| = 0 \\
d_{B,D} &= |f_{(O,B)} - f_{(O,D)}| = 0
\end{aligned}
$$

$$
\begin{aligned}
s_{O(A,B)} &= \frac{1}{1+pd_{A,B}^q} = 1{,}00E+000 \\
s_{O(A,D)} &= \frac{1}{1+pd_{A,D}^q} = 1{,}00E+000 \\
s_{O(B,D)} &= \frac{1}{1+pd_{B,D}^q} = 1{,}00E+000
\end{aligned}
$$

$$
\begin{aligned}
w_{(O,A)} &= \frac{1}{1+(s_{O(A,B)} \quad s_{O(A,D)}} = ,5 \\
w_{(O,B)} &= \frac{1}{1+(s_{O(A,B)} \quad s_{O(B,D)}} = ,5 \\
w_{(O,D)} &= \frac{1}{1+(s_{O(A,D)} \quad s_{O(B,D)}} = ,5
\end{aligned}
$$

$$
y_M^4 = \frac{(w_{(O,A)} \quad f_{(O,A)}) + (w_{(O,B)} \quad f_{(O,B)}) + (w_{(O,D)} \quad f_{(O,D)})}{w_{(O,A)} + w_{(O,B)} + w_{(O,D)}} = 0{,}4905.
$$

$$\text{mín}(|y_M^4 - f_{(O,A)}|, |y_M^4 - f_{(O,B)}|, |y_M^4 - f_{(O,D)}|) = 0{,}1226.$$

este valor numérico esta asociado a la entrada $f_{O,B}$, por lo tanto el nodo ganador del consenso $G_M^4 = B$

$$\beta^5 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O \end{array} \\ \begin{pmatrix} 0 & 350 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 359 & 407 & 908 \\ 350 & 0 & 87 & 650 & 0 & 0 & 680 & 0 & 0 & 0 & 0 & 50 & 0 & 40 & 568 \\ 0 & 87 & 0 & 300 & 632 & 203 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 35 & 300 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 632 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 203 & 0 & 0 & 0 & 0 & 900 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 680 & 0 & 0 & 0 & 0 & 0 & 171 & 79 & 42 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 900 & 171 & 0 & 0 & 250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 79 & 0 & 0 & 108 & 304 & 250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 42 & 250 & 108 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 304 & 0 & 0 & 305 & 788 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 & 250 & 0 & 305 & 0 & 607 & 100 & 0 \\ 359 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 788 & 607 & 0 & 39 & 0 \\ 407 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 39 & 0 & 0 \\ 908 & 568 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{B.10}$$

$$\begin{aligned} k &= 5 \\ n_s &= B \\ n_d &= E \\ J_M &= A, C, D, G, L, N, O \\ \gamma &= 20 \\ \tau &= 1000 \end{aligned}$$

$$\begin{aligned} f_{B,A} &= e^{-\left(\frac{(\lambda(B,A)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,A))^2}{\sigma_3^2}\right)} = 0{,}3678 \\ f_{B,C} &= e^{-\left(\frac{(\lambda(B,C)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,C))^2}{\sigma_3^2}\right)} = 0{,}3678 \\ f_{B,D} &= e^{-\left(\frac{(\lambda(B,D)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,D))^2}{\sigma_3^2}\right)} = 0{,}3678 \\ f_{B,G} &= e^{-\left(\frac{(\lambda(B,G)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,G))^2}{\sigma_3^2}\right)} = 0{,}3678 \\ f_{B,L} &= e^{-\left(\frac{(\lambda(B,L)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,L))^2}{\sigma_3^2}\right)} = 0{,}3678 \\ f_{B,N} &= e^{-\left(\frac{(\lambda(B,N)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,N))^2}{\sigma_3^2}\right)} = 0{,}3678 \\ f_{B,O} &= e^{-\left(\frac{(\lambda(B,O)-\Gamma)^2}{\sigma_1^2} + \frac{hops^2}{\sigma_2^2} + \frac{(\tau-\beta(B,O))^2}{\sigma_3^2}\right)} = 0{,}3678 \end{aligned}$$

$$d_{A,C} = |f_{(B,A)} - f_{(B,C)}| = 0$$
$$d_{A,D} = |f_{(B,A)} - f_{(B,D)}| = 0$$
$$d_{A,G} = |f_{(B,A)} - f_{(B,G)}| = 0$$
$$d_{A,L} = |f_{(B,A)} - f_{(B,L)}| = 3{,}1964E-011$$
$$d_{A,N} = |f_{(B,A)} - f_{(B,N)}| = 0$$
$$d_{A,O} = |f_{(B,A)} - f_{(B,O)}| = 0$$
$$d_{C,D} = |f_{(B,C)} - f_{(B,D)}| = 0$$
$$d_{C,G} = |f_{(B,C)} - f_{(B,G)}| = 0$$
$$d_{C,L} = |f_{(B,C)} - f_{(B,L)}| = 3{,}1964E-011$$
$$d_{C,N} = |f_{(B,C)} - f_{(B,N)}| = 0$$
$$d_{C,O} = |f_{(B,C)} - f_{(B,O)}| = 0$$

$$d_{D,G} = |f_{(B,D)} - f_{(B,G)}| = 0$$
$$d_{D,L} = |f_{(B,D)} - f_{(B,L)}| = 3{,}1964E-011$$
$$d_{D,N} = |f_{(B,D)} - f_{(B,N)}| = 0$$
$$d_{D,O} = |f_{(B,D)} - f_{(B,O)}| = 0$$
$$d_{G,L} = |f_{(B,G)} - f_{(B,L)}| = 3{,}1964E-011$$
$$d_{G,N} = |f_{(B,G)} - f_{(B,N)}| = 0$$
$$d_{G,O} = |f_{(B,G)} - f_{(B,O)}| = 0$$
$$d_{L,N} = |f_{(B,L)} - f_{(B,N)}| = 3{,}1964E-011$$
$$d_{L,O} = |f_{(B,L)} - f_{(B,O)}| = 3{,}1964E-011$$
$$d_{N,O} = |f_{(B,N)} - f_{(B,O)}| = 0$$

$$s_{B(A,C)} = \frac{1}{1 + pd^q_{A,C}} = 1{,}00E + 000$$
$$s_{B(A,D)} = \frac{1}{1 + pd^q_{A,D}} = 1{,}00E + 000$$
$$s_{B(A,G)} = \frac{1}{1 + pd^q_{A,G}} = 1{,}00E + 000$$
$$s_{B(A,L)} = \frac{1}{1 + pd^q_{A,L}} = 1{,}00E + 000$$
$$s_{B(A,N)} = \frac{1}{1 + pd^q_{A,N}} = 1{,}00E + 000$$
$$s_{B(A,O)} = \frac{1}{1 + pd^q_{A,O}} = 1{,}00E + 000$$
$$s_{B(C,D)} = \frac{1}{1 + pd^q_{C,D}} = 1{,}00E + 000$$
$$s_{B(C,G)} = \frac{1}{1 + pd^q_{C,G}} = 1{,}00E + 000$$
$$s_{B(C,L)} = \frac{1}{1 + pd^q_{C,L}} = 1{,}00E + 000$$
$$s_{B(C,N)} = \frac{1}{1 + pd^q_{C,N}} = 1{,}00E + 000$$
$$s_{B(C,O)} = \frac{1}{1 + pd^q_{C,O}} = 1{,}00E + 000$$

$$s_{B(D,G)} = \frac{1}{1 + pd^{q}_{D,G}} = 1{,}00E + 000$$
$$s_{B(D,L)} = \frac{1}{1 + pd^{q}_{D,L}} = 1{,}00E + 000$$
$$s_{B(D,N)} = \frac{1}{1 + pd^{q}_{D,N}} = 1{,}00E + 000$$
$$s_{B(D,O)} = \frac{1}{1 + pd^{q}_{D,O}} = 1{,}00E + 000$$
$$s_{B(G,L)} = \frac{1}{1 + pd^{q}_{G,L}} = 1{,}00E + 000$$
$$s_{B(G,N)} = \frac{1}{1 + pd^{q}_{G,N}} = 1{,}00E + 000$$
$$s_{B(G,O)} = \frac{1}{1 + pd^{q}_{G,O}} = 1{,}00E + 000$$
$$s_{B(L,N)} = \frac{1}{1 + pd^{q}_{L,N}} = 1{,}00E + 000$$
$$s_{B(L,O)} = \frac{1}{1 + pd^{q}_{L,O}} = 1{,}00E + 000$$
$$s_{B(N,O)} = \frac{1}{1 + pd^{q}_{N,O}} = 1{,}00E + 000$$

$$w_{(B,A)} = \frac{1}{1 + (s_{B(A,C)} \quad s_{B(A,D)} \quad s_{B(A,G)} \quad s_{B(A,L)} \quad s_{B(A,N)} \quad s_{B(A,O)}} = ,5$$
$$w_{(B,C)} = \frac{1}{1 + (s_{B(A,C)} \quad s_{B(C,D)} \quad s_{B(C,G)} \quad s_{B(C,L)} \quad s_{B(C,N)} \quad s_{B(C,O)}} = ,5$$
$$w_{(B,D)} = \frac{1}{1 + (s_{B(A,D)} \quad s_{B(C,D)} \quad s_{B(D,G)} \quad s_{B(D,L)} \quad s_{B(D,N)} \quad s_{B(D,O)}} = ,5$$
$$w_{(B,G)} = \frac{1}{1 + (s_{B(A,G)} \quad s_{B(C,G)} \quad s_{B(D,G)} \quad s_{B(G,L)} \quad s_{B(G,N)} \quad s_{B(G,O)}} = ,5$$
$$w_{(B,L)} = \frac{1}{1 + (s_{B(A,L)} \quad s_{B(C,L)} \quad s_{B(D,L)} \quad s_{B(G,L)} \quad s_{B(L,N)} \quad s_{B(L,O)}} = ,5$$
$$w_{(B,N)} = \frac{1}{1 + (s_{B(A,N)} \quad s_{B(C,N)} \quad s_{B(D,N)} \quad s_{B(G,N)} \quad s_{B(L,N)} \quad s_{B(L,O)}} = ,5$$
$$w_{(B,O)} = \frac{1}{1 + (s_{B(A,O)} \quad s_{B(C,O)} \quad s_{B(D,O)} \quad s_{B(G,O)} \quad s_{B(L,O)} \quad s_{B(N,O)}} = ,5$$

$$y_M^5 = \frac{(w_{(B,A)}f_{(B,A)}) + (w_{(B,C)}f_{(B,C)}) + (w_{(B,D)}f_{(B,D)}) + (w_{(B,G)}f_{(B,G)}) + (w_{(B,L)}f_{(B,L)}) + (w_{(B,N)}f_{(B,N)}) + (w_{(B,O)}f_{(B,O)})}{w_{(B,A)} + w_{(B,C)} + w_{(B,D)} + w_{(B,G)} + w_{(B,L)} + w_{(B,N)} + w_{(B,O)}}$$
$$= 1{,}1649.$$

$$\text{mín}(|y_M^5 - f_{(B,A)}|, |y_M^5 - f_{(B,C)}|, |y_M^5 - f_{(B,D)}|, |y_M^5 - f_{(B,G)}|, |y_M^5 - f_{(B,L)}|, |y_M^5 - f_{(B,N)}|, |y_M^5 - f_{(B,O)}|) = 0{,}7970.$$

este valor numérico esta asociado a la entrada $f_{B,C}$, por lo tanto el nodo ganador del consenso $G_M^5 = C$ Para el paso $k = 6$, se tienen los valores $n_s = C, n_d = E$, el conjunto $J = D, E, F$, como $n_d \in J$, el algoritmo ya no hace más cálculos y construye la ruta. Se logró establecer una ruta y quedando: $K \rightarrow G^1 \rightarrow G^2 \rightarrow G^3 \rightarrow G^4 \rightarrow G^5 \rightarrow E = K \rightarrow M \rightarrow A \rightarrow O \rightarrow B \rightarrow C \rightarrow E$. En la figura B.4 se muestra esta ruta.

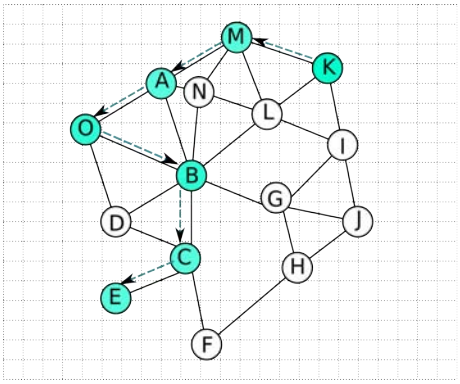


Figura B.4: Ruta obtenida en la simulación con una matriz de carga variable para búsqueda de un salto.

En este ejemplo la información de la matriz de carga varía, por lo que el resultado del consenso cambia en el transcurso del tiempo.

Referencias

[AGG00] Ionut D. Aron, Sandeep K. S. Gupta, and Eep K. S. Gupta. Analytical comparison of local and end-to-end error recovery in reactive routing protocols for mobile ad hoc networks. In *Proc. ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 69–76, 2000. 7

[Agn76] Carson E. Agnew. On quadratic adaptive routing algorithms. *Commun. ACM*, 19(1):18–22, 1976. 41, 42, 43

[AV11] Magali Arellano-Vázquez. Estudio de algoritmos de ruteo considerando restricciones de tiempo real. Master's thesis, Posgrado en ciencia e ingeniería de la computación. Universidad Nacional Autónoma de México, 2011. 10, 66

[AVBPOA11] Magali Arellano-Vázquez, Héctor Benítez-Pérez, and Jorge L. Ortega-Arjona. Study of routing algorithms considering real time restrictions using a connectivity function. In *Proceedings of the 12th Annual Conference on Towards Autonomous Robotic Systems*, TAROS'11, pages 412–413, Berlin, Heidelberg, 2011. Springer-Verlag. 69

[AW00] J.M. Aldous and R.J. Wilson. *Graphs and Applications: An Introductory Approach*. Springer, 2000. 14

[BF08] Ioannis Broustis and Michalis Faloutsos. Review article routing in vehicular networks: Feasibility, modeling, and security, 2008.

[BLG15] Abdulkader Benchi, Pascale Launay, and Frédéric Guidec. Solving consensus in opportunistic networks. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, ICDCN '15, pages 1:1–1:10, New York, NY, USA, 2015. ACM. 58

[BR00] R. Balakrishnan and K. Ranganathan. *A Textbook of Graph Theory*. Universitext (1979). Springer, 2000. 14

[CCN06] Matthew Caesar, Miguel Castro, and Edmund B. Nightingale. Virtual ring routing: network routing inspired by dhts. In *In Proc. of ACM SIGCOMM*, pages 351–362, 2006. xiii, 8, 53, 54

[CM02] Benjie Chen and Robert Morris. Scalable landmark routing and address lookup for multi-hop wireless networks, 2002. 8

[Cou01] George. Dollimore Jean. Kindberg Tim Couloris. *Distributed Systems, Concepts and Design*. Queen Mary and Westfield College,University of London, London, 2001. 16

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001. 2, 24, 26, 28

[DPH05] Saumitra M. Das, Himabindu Pucha, and Y. Charlie Hu. Performance comparison of scalable location services for geographic ad hoc routing, 2005. 8, 54, 55, 56

[DYN97] Jose Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1997. 30, 31, 32

[eB08] Wei en and Randal W. Beard. *Distributed consensus in multi-vehicle cooperative control : theory and applications*. Communications and control engineering. Springer, London, 2008. 16

[FB13] Adriano Fagiolini and Antonio Bicchi. On the robust synthesis of logical consensus algorithms for distributed intrusion detection. *Automatica*, 49(8):2339–2350, 2013. 9, 10

[FDB11] Adriano Fagiolini, Nevio Dubbini, and Antonio Bicchi. Distributed consensus on set-valued information. *CoRR*, abs/1101.2275, 2011. 9

[FFR+04] Rodrigo Fonseca, Rodrigo Fonseca, Sylvia Ratnasamy, Sylvia Ratnasamy, David Culler, David Culler, Scott Shenker, Scott Shenker, Ion Stoica, and Ion Stoica. Beacon vector routing: Scalable point-to-point in wireless sensornets. 2004. 8

[FMBB10] Adriano Fagiolini, Simone Martini, Davide Di Baccio, and Antonio Bicchi. A self-routing protocol for distributed consensus on logical information. In *IROS*, pages 5151–5156. IEEE, 2010. 9

[GR06] Rachid Guerraoui and Luís Rodrigues. *Introduction to Reliable Distributed Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 17, 18, 19, 20, 21, 22

[HMMD02] Urs Hengartner, Sue Moon, Richard Mortier, and Christophe Diot. Detection and analysis of routing loops in packet traces. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 107–112, New York, NY, USA, 2002. ACM. xiii, 36, 37, 38

[HPS02] Zygmunt J Haas, Marc R Pearlman, and Prince Samar. The zone routing protocol (zrp) for ad hoc networks. *draft-ietf-manet-zone-zrp-04. txt*, 2002. 8

[JDWB10] Anne Jorstad, Daniel DeMenthon, I-Jeng Wang, and Philippe Burlina. Distributed consensus on camera pose. *IEEE Trans Image Process*, 19(9):2396–407, 2010.

[JHM07] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), February 2007. 56, 58

[JKBK+08] John P. John, Ethan Katz-Bassett, Arvind Krishnamurthy, Thomas Anderson, and Arun Venkataramani. Consensus routing: The internet as a distributed system. In *Proceedings of the 5th USENIX*

Symposium on Networked Systems Design and Implementation, NSDI'08, pages 351–364, Berkeley, CA, USA, 2008. USENIX Association. 10, 59, 60, 61

[JM96a] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996. 2, 3, 6

[JM96b] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996. 7

[JMC+01] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. pages 62–68, 2001. 2, 7, 50, 51, 58

[JT87] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, jan 1987. 5

[KK00] Brad Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 243–254, New York, NY, USA, 2000. ACM. 8

[KW05] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005. 14

[LKA+10] Jung Woo Lee, Branislav Kusy, Tahir Azim, Basem Shihada, and Philip Levis. Whirlpool routing for mobility. In *Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '10, pages 131–140, New York, NY, USA, 2010. ACM. 57

[LKLK08] Zhipeng Liu, Jeongmi Kim, Bokman Lee, and ChongGun Kim. A routing protocol based on adjacency matrix in ad hoc mobile networks. *Advanced Language Processing and Web Information Technology, International Conference on*, 0:430–436, 2008. 8, 9

[LSBB01] G. Latif-Shabgahi, Julian M. Bass, and Stuart Bennett. History-based weighted average voter: A novel software voting algorithm for fault-tolerant computer systems. In *PDP*, pages 402–409. IEEE Computer Society, 2001. 10, 29, 61, 62, 67, 69

[Mac05] James Macfarlane. *Network Routing Basics*. John Wiley & Sons, Inc., New York, NY, USA, 2005. 28, 29

[MD05] Mahesh K. Marina and Samir R. Das. Routing in mobile ad hoc networks. In Prasant Mohapatra and Srikanth V. Krishnamurthy, editors, *Ad Hoc Networks*, pages 63–90. Springer US, 2005. 8

[MGLA96] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.*, 1(2):183–197, oct 1996. 7

[Mie11] Piet Van Mieghem. *Graph Spectra for Complex Networks*. Cambridge University Press, New York, NY, USA, 2011. 14

[Moh91] Bojan Mohar. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, pages 871–898. Wiley, 1991.

[Moh98] Prasant Mohapatra. Wormhole routing techniques for directly connected multicomputer systems. *ACM Comput. Surv.*, 30(3):374–410, 1998. xiii, 30

[MR07] Deepankar Medhi and Karthikeyan Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007. 2, 24, 25, 26, 27, 28, 32, 33, 34, 40

[OSFM07] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. In *Proceedings of the IEEE*, page 2007, 2007. 17

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, oct 1994. 6, 7

[PC97] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks, 1997. 2, 7, 51, 52

[PD07] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach, Fourth Edition (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann, 4 edition, March 2007. 25, 47, 48, 49, 50, 67

[PGH00] Guangyu Pei, Mario Gerla, and Xiaoyan Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc '00, pages 11–18, Piscataway, NJ, USA, 2000. IEEE Press. 8

[PH06] M. R. Pearlman and Z. J. Haas. Determining the optimal configuration for the zone routing protocol. *IEEE J.Sel. A. Commun.*, 17(8):1395–1414, September 2006. 7

[PSA06] A. Patooghy and H. Sarbazi-Azad. Performance comparison of partially adaptive routing algorithms. *Advanced Information Networking and Applications, International Conference on*, 2:763–767, 2006. 35, 36

[PSJ99] H.K. Pung, J. Song, and L. Jacob. Fast and efficient flooding based qos routing algorithm. In *Computer Communications and Networks, 1999. Proceedings. Eight International Conference on*, pages 298–303, 1999. 44, 45, 136, 137

[RHS03] Venugopalan Ramasubramanian, Zygmunt J. Haas, and Emin Gün Sirer. Sharp: a hybrid adaptive routing protocol for mobile ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '03, pages 303–314, New York, NY, USA, 2003. ACM. 7, 8

[Ros00] Kenneth H. Rosen. *Handbook of discrete and combinatorial mathematics*. AT&T Laboratories, 2000. 23, 24

[Sei85] Charles L. Seitz. The cosmic cube. *Commun. ACM*, 28(1):22–33, 1985. 29

[Sey98] Ayse Yasemin Seydim. Wormhole routing in parallel computers, 1998. 2, 30

[SGLA01] Marcelo Spohn and J.J. Garcia-Luna-Aceves. Neighborhood aware source routing, 2001. 7

[Sib02] Galvin Peter Gagne Greg Siberschartz Abraham. *Sistemas Operativos*. Limusa Wiley, 2002. 34, 35, 40

[SK04] Cigdem Sengul and Robin Kravets. Bypass routing: An on-demand local recovery protocol for ad hoc networks, 2004. 2, 7, 9, 136

[SRM03] Reza Olfati Saber, Saber Richard, and Richard M. Murray. Consensus protocols for networks of dynamic agents, 2003.

[Ste95] Martha Steenstrup, editor. *Routing in Communications Networks*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1995. 6

[Sun97] Charles Perkins Sun. Ad-hoc on-demand distance vector routing. In *In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1997. 2, 7, 48, 49, 58

[Tej10] Cristian Gastón Tejia. Estudio y análisis de mecanismos orientados al robustecimiento de antop, utilizando ruteo proactivo. Master's thesis, Universidad de Buenos Aires. Facultad de Ingeniería, 2010. 54

[Toh97] Chai-Keong Toh. Associativity-based routing for ad hoc mobile networks. *Wirel. Pers. Commun.*, 4(2):103–139, mar 1997. 7

[VB00] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, 2000. 46, 58

[YFG+10] Yağiz Onat Yazir, Roozbeh Farahbod, Adel Guitouni, Sudhakar Ganti, and Yvonne Coady. Adaptive routing in mobile ad hoc networks based on decision aid approach. In *Proceedings of the 8th ACM International Workshop on Mobility Management and Wireless Access*, MobiWac '10, pages 1–10, New York, NY, USA, 2010. ACM. 57

[ZWXS11] Xin Ming Zhang, En Bo Wang, Jing Jing Xia, and Dan Keun Sung. An estimated distance-based routing protocol for mobile ad hoc networks. *Vehicular Technology, IEEE Transactions on*, 60(7):3473 –3484, sept. 2011. 9

Glosario

A

Ad Hoc

Una red ad hoc inalámbrica es un tipo de red inalámbrica descentralizada. La red es ad hoc porque no depende de una infraestructura pre-existente, como routers (en redes cableadas) o de puntos de accesos en redes inalámbricas administradas. En lugar de ello, cada nodo participa en el encaminamiento mediante el reenvío de datos hacia otros nodos, de modo que la determinación de estos nodos hacia la información se hace dinámicamente sobre la base de conectividad de la red. 48

B

Broadcast

El direccionamiento de un paquete a todos los destinos utilizando un código especial en el campo de dirección. Cuando se transmite un paquete con este código, todas las máquinas de la red lo reciben y procesan. Este modo de operación se conoce como difusión (broadcasting). 29

C

Contando al infinito

El problema de la cuenta hasta el infinito es causado por fallos de los enlaces que divide la red en 2 o más segmentos. Cuando se divide la red, los nodos en una parte del segmento no pueden llegar a los nodos en la otra parte del segmento La distancia entre los nodos en diferentes segmentos es infinito. 48

Cubo cósmico

Red de tipo hipercubo 6. Procesadores 8086 con 128 Kb de memoria. Intel iPSC/1 con procesadores Intel 80286 con 212 Kbytes de memoria, 8 puertos de E/S por cada nodo para formar un hipercubo de dimensión 7, el octavo puerto comunica el nodo con el nodo por red Ethernet. 30

E

enrutador

En una red de computadoras existe un nodo llamado enrutador, que se encarga de trazar una ruta entre un nodo origen a un nodo destino. 28

M

MANETS

Acrónimo de Mobile ad hoc network. En algunas ocasiones denominada también como malla de nodos móviles (mobile mesh network), se trata de una red de dispositivos conectados por wireless y que poseen propiedades de autoconfiguración, además de poseer cierta movilidad (es decir se encuentran montados en plataformas móviles). x, 57

Multicast

La transmisión a un subconjunto de máquinas, algo conocido como multidifusión (multicasting). Un esquema posible es la reserva de un bit para indicar la multidifusión. Los bits de dirección n–1 restantes pueden contener un número de grupo. Cada máquina puede ”suscribirse” a alguno o a todos los grupos. Cuando se envía un paquete a cierto grupo, se distribuye a todas las máquinas que se suscriben a ese grupo. 29

T

Teorema de Kuhn-Tucker

condiciones necesarias y suficientes para que la solución de una programación no lineal sea óptima. Es una generalización del método de los Multiplicadores de Lagrange. 41

U

Unicast

La transmisión de punto a punto con un emisor y un receptor se conoce como unidifusión (unicasting). 29

Índice alfabético

Algoritmo de Bellman-Ford, 24, 26
Algoritmo de Dijkstra, 26, 28
Algoritmo de inundación, 43
Algoritmo de votaciones por promedio ponderado, 61
Algoritmos de consenso, 18
Algoritmos de encaminamiento, 39
Algoritmos de encaminamiento adaptativo, 40
Arduino, PcDuino, 92

Ciclos de encaminamiento., 36
Conectividad., 14
Consenso, 16
Consenso aleatorio., 22
Consenso jerárquico, 20
Consenso promedio, 22
Consenso regular, 18
Consenso., 17
Consensus routing, 59
Cubo cósmico, 30

Encaminamiento, 33
Encaminamiento óptimo alternante, 41
Encaminamiento adaptativo, 36
Encaminamiento determinista, 35
Encaminamiento parcialmente adaptativo, 36
Estado global, 15

Grado de un vértice., 14
Grid, 54

Matriz de adyacencia, 23
Matriz de adyacencia., 14
Matriz de incidencia, 24

Problema, 1
Protocolo de encaminamiento de fuente dinámica., 56
Protocolo de encaminamiento Whirlpool., 57
Protocolos de encaminamiento
    Estado de enlace, 49
Protocolos de encaminamiento., 46
    Estado de enlace
        Encaminamiento optimizado de estado de enlace., 50
    Vector de distancia., 47
    Vector de distancia.

Ad hoc On Demand Distance Vector., 48

Recorrido y rutas., 14
Redes de comunicación, 29
Redes Oportunistas, 58
Representación de redes, 23

Switching, 30

Temporally-Ordered Routing Algorithm, 51
Tipos de encaminamiento, 35

Vecindario., 14
Virtual Ring Routing, 53

Wormhole, 30