INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Enrutamiento compacto usando etiquetas basadas en prefijos

Tesis que presenta

Ing. Ismael Quintana Avalos

Para Obtener el Grado de

Maestro en Ciencias de la Computación

Directores

Dr. Rolando Menchaca Méndez
Dr. Ricardo Menchaca Méndez

México D.F. Julio 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 17 del mes de junio de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Enrutamiento compacto usando etiquetas basadas en prefijos"

Presentada por el alumno:

QUINTANA	**AVALOS**	**ISMAEL**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	1	2	0	4	2	1

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA
Directores de Tesis

Dr. Rolando Menchaca Méndez

Dr. Ricardo Menchaca Méndez

Dra. Nareli Cruz Cortés

Dr. Rolando Quintero Téllez

Dr. José Giovanni Guzmán Lugo

M. en C. Sergio Sandoval Reyes

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F. el día 19 del mes de Junio del año 2014, el que suscribe Ismael Quintana Avalos alumno del Programa de Maestría en Ciencias de la Computación, con número de registro A120421, adscrito al Centro de Investigación en Computación, manifiesta que es el autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Rolando Menchaca Méndez y el Dr. Ricardo Menchaca Méndez y cede los derechos del trabajo titulado "Enrutamiento compacto usando etiquetas basadas en prefijos", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a las siguientes direcciones ismael.qa@gmail.com . Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Ismael Quintana

Ismael Quintana Avalos

Nombre y firma del alumno(a)

Resumen

Actualmente, los protocolos de enrutamiento tradicionales basados en direcciones IP que soportan a Internet, se están enfrentando a problemas debido a que no son capaces de adaptar su funcionamiento a la creciente cantidad de usuarios que se añaden a la red global. Para afrontar los problemas de escalabilidad, se ha hecho investigación en lo que se conoce como enrutamiento compacto. Este tipo de enrutamiento presenta la característica que es escalable y su funcionamiento es estable aún cuando trabaje con una gran cantidad de nodos, sin embargo, también presenta ciertos puntos débiles. En concreto, los esquemas de enrutamiento compacto que utilizan etiquetas basadas en prefijos, tienen las desventajas de que pueden llegar a generar rutas de gran longitud a comparación de la ruta óptima y que tienden a concentrar el tráfico en ciertas áreas de la red, específicamente en las regiones cercanas al nodo que se toma como raíz para comenzar el etiquetado.

En la presente tesis se propone como solución a las desventajas observadas del enrutamiento compacto un algoritmo de enrutamiento compacto el cual usa etiquetas basadas en prefijos, cuya principal característica es el uso de dos nodos raíz para comenzar con un doble etiquetado de la red. Esto se realiza con la finalidad de obtener rutas alternas hacia cualquier destino. Elegir al nodo centro promedio de la red y al nodo más alejado a él, de acuerdo a los resultados obtenidos en la experimentación realizada, fue la mejor estrategia de elección ya que mostró la generación de rutas de menor longitud, un *stretch* menor y una reducción en cuanto a la concentración del tráfico de la red. Con la información obtenida en la experimentación, también se concluyó que la inversión de recursos para la elección de los nodos que serán las raíces dentro de la red se traduce en mejoras en las métricas consideradas.

Abstract

Nowadays, traditional routing protocols based on IP addresses that support Internet are facing problems because they are not able to adapt their operations to the increasing number of users joining the global network. To face the scalability issues, research on what is known as a compact routing has been done. This type of routing has the feature that is scalable and its performance is stable even working with a large number of nodes, however it also has some weaknesses. Specifically, compact routing schemes that use prefix labels have the disadvantages that they can generate much longer routes compared to the optimal route and they also tend to concentrate traffic in certain areas of the network, specifically the regions near to the node elected as root to start the labeling.

In this thesis as a solution to the observed disadvantages of compact routing is proposed as a compact routing algorithm that uses prefix labels which main feature is the use of two root nodes to start with a double labeling of the network. This is done in order to obtain alternative routes to any destination. According to the results of the experimentation, choosing the node considered as the average center of the network and the farthest node to it was the best strategy of choosing two nodes because it showed the generation of shorter routes, a lower stretch and a reduction of network traffic concentration. With the information obtained in the experimentation we also concluded that the investment of resources for choosing the nodes that will be the roots in the network gives improvements in the considered metrics.

Agradecimientos

Al Instituto Politécnico Nacional por haberme brindado la oportunidad de ser parte de la comunidad politécnica.

Al Centro de Investigación en Computación por ser mi seguno hogar y darme la oportunidad de desarrollarme académicamente y gracias al CONACYT por su apoyo económico.

Agradezco especialmente a Ismael Quintana Martínez, mi padre y a Elizabeth Avalos Pérez, mi madre, por todo esa confianza, apoyo y amor incondicional que me han brindado durante todos mis años de vida y en especial en estos dos años y medio. Aunque hemos estado separados geográficamente por algunos períodos de tiempo, siguen estando igual de cerca para mí. Gracias por se un ejemplo de sacrificio y esfuerzo, sin ustedes nada de esto sería posible.

Irving e Ilse, gracias por su apoyo y cada una de sus palabras de aliento que me han brindado. Espero ser un ejemplo para ustedes.

A mi familia y a mis amigos en ciudad Juárez que en cada momento tuvieron las palabras para animarme a terminar este reto. Muchas gracias.

Dr. Rolando Menchaca Méndez gracias por guiarme durante este trabajo de tesis, por compartir su conocimiento, por su apoyo, sus enseñanzas, por siempre ser paciente conmigo y sobretodo por ser una gran persona y ejemplo a seguir.

Dr. Ricardo Menchaca Méndez gracias por ser parte de este trabajo de tesis, por su apoyo y sus buenos deseos.

Gracias a los profesores que compartieron sus conocimientos en cada una de sus clases.

A mis compañeros del laboratorio de Comunicaciones y Redes de Computadora los cuales siempre estuvieron dispuestos a ayudarme cuando necesité de su guía y a los amigos del laboratorio de PIIG. Gracias.

Daniela, Enrique, Manuel, Mónica, Joel y Paty gracias por bridarme su amistad durante esta etapa de mi vida y por haber compartido grandes momentos en estos dos años y medio.

¡GRACIAS!

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En nuestros días, las redes de computadoras juegan un papel preponderante en la industria, la ciencia, el entretenimiento y en general, en todos los aspectos de la vida humana. Por lo anterior, cada vez son más los dispositivos que se conectan a la red global haciendo que su escala actual sea de cientos de millones de nodos. Ésto impone una serie de desafíos a los algoritmos que implementan las funciones de las diferentes capas de la pila de protocolos de Internet, pero en particular para aquellos algoritmos cuyo ámbito de acción es la red completa. Entre estos algoritmos están los de enrutamiento, cuya función es determinar la secuencia de saltos de máquina a máquina que un paquete de datos debe seguir para llegar a su destino. Es importante mencionar que el espacio de búsqueda de un algoritmo de enrutamiento es exponencialmente grande con respecto al número de nodos que componen la red y por lo tanto la eficiencia con que el algoritmo determine la mejor ruta para llevar paquetes desde un origen hasta un destino es un factor determinante en el desempeño de la red como un todo. [37].

Actualmente, el funcionamiento de los protocolos de enrutamiento utilizados en Internet está basado en las direcciones proporcionadas por el protocolo IP. Con dichas direcciones, los algoritmos de enrutamiento buscan una ruta entre los dos nodos dentro de la red (nodo origen y nodo destino). La elección de esta ruta se hace tomando en cuenta alguna métrica deseada. Generalmente se busca que la distancia entre el origen y el destino sea la distancia más corta entre ellos y por lo tanto una de las métricas más utilizadas es la distancia en saltos entre el origen y el destino. Esto no quiere decir que los algoritmos no puedan utilizar otras métricas de costo para los caminos que calculan como puede ser el retardo, el consumo de energía, el costo monetario, entre otros.

En los esquemas de enrutamiento tradicional, cada nodo dentro de la red debe almacenar una tabla de enrutamiento, que le permite saber hacia qué puerto dirigir cualquier paquete que le llegue. El proceso consiste en simplemente revisar la entrada correspondiente a la dirección de destino para obtener cuál es el puerto de salida por el que deberá transmitirse el paquete. Como puede verse, esta forma de operación impone que por cada destino dentro de la red, debe haber una entrada en la tabla de enrutamiento. Por lo tanto, si se tiene una red con una gran cantidad de nodos el tamaño de la tabla de enrutamiento será también de gran tamaño y se tendrá la necesidad de ocupar mayor cantidad de recursos del nodo, lo cual hace que estos protocolos presenten la característica de ser poco escalables.

Al decir que los protocolos son poco escalables, se hace referencia a su capacidad de adaptarse al crecimiento de la red. Con un gran número de elementos dentro de la red, los nodos consumirán una mayor cantidad de memoria de la que disponen sólo para mantener la información contenida en sus tablas de enrutamiento. Otro de los recursos que se utilizarán en

mayor medida es el procesador, puesto que al tener gran cantidad de información se requerirá más procesamiento para poder calcular la ruta al nodo destino. Es por estas razones que la convergencia de los protocolos se hace lenta y ésto se ve reflejado en el desempeño de la red que a su vez se manifiesta como una mala experiencia para los usuarios finales.

Para tratar de solucionar este problema, se ha hecho investigación en esquemas de enrutamiento que generen tablas de enrutamiento que crezcan de manera más lenta a lo que crece la red, esto quiere decir, que crezcan de manera sublineal. A este tipo de esquemas se les conoce como *enrutamiento compacto*. El objetivo del enrutamiento compacto es estudiar los límites de la escalabilidad de los protocolos de enrutamiento para diseñar algoritmos de complejidad espacial y temporal baja cuyo desempeño se acerque a dichos límites de escalabilidad [19]. Un concepto importante relacionado con el enrutamiento compacto es el de *estiramiento* (*stretch*), que de acuerdo con [5], es el radio máximo entre la longitud de la ruta calculada por el esquema de enrutamiento compacto entre dos puntos y la misma distancia calculada por un esquema de ruta más corta.

Aunado al problema del tamaño de las tablas de enrutamiento, también se tiene la centralización de servicios dentro de la red, como ejemplo, el DNS (sistema de nombres de dominio por sus siglas en inglés). Este servicio se encuentra implantado en un sólo equipo que es el encargado de proporcionarlo, lo cual genera cuellos de botellas dentro de la red por la concentración de tráfico hacia ese punto.

Otro problema relacionado con el esquema de direccionamiento basado en direcciones IP es que impone una separación entre la identidad de las máquinas y del contenido que alojan. Dada esta separación, se requiere de un subsistema que se encargue de almacenar las relaciones entre las direcciones IP de las máquinas y los nombres o identificadores de los objetos de datos que están almacenados en dichas máquinas. En la arquitectura actual de Internet, el servicio de nombres de dominio (DNS por sus siglas en inglés) es el responsable de implementar un sistema de almacenamiento jerárquico que aloja dichas relaciones. De esta forma, para acceder a un objeto de datos es necesario realizar una consulta al DNS sobre la dirección IP de la máquina que aloja a un objeto en particular. En el mejor de los casos esta consulta se resuelve localmente por medio de alguna de las copias almacenadas en los servidores DNS cercanos a la máquina cliente, pero en el peor de los casos requiere de una consulta hasta el servidor raíz del servicio de resolución de nombres. Lo anterior tiene la desventaja de que para establecer una ruta, antes se tuvieron que establecer otras rutas hacia los servidores que implementan los distintos niveles del servicio de nombres de dominio. La Fig. 1.1 muestra un ejemplo de la interacción de los nodos con un servidor que implementa el servicio de nombres de dominio.

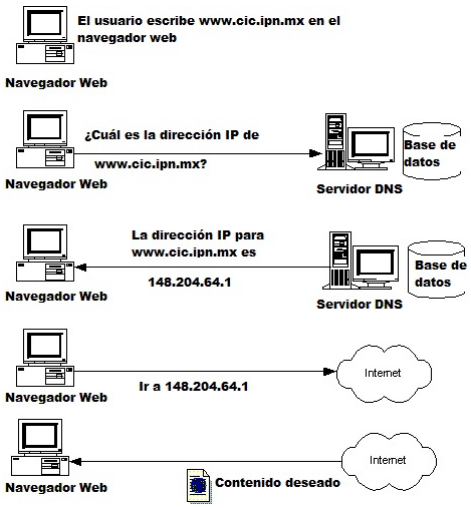


Figura 1.1: Proceso de consulta a un servidor de nombres de dominio (DNS).

Estos múltiples viajes dentro de la red, así como la centralización de servicios afectan de manera considerable el rendimiento de la red y a su vez afectan la escalabilidad de la misma, para finalmente traducir esas desventajas a una mala experiencia para los usuarios.

Al pensar en estrategias para mitigar esos problemas, se ha analizado el tráfico generado en Internet y se ha visto que la mayoría es tráfico orientado a contenido [17]. Esto quiere decir que lo que le importa a los usuarios es el contenido y no la máquina que lo contenga, es así que se comenzó a hablar de las *redes orientadas a contenido*. En esta clase de redes, los servicios de resolución de nombres se encuentran distribuidos en la red, lo cual significa, que la red no cuenta con nodos que realicen tareas especiales y cualquier nodo es capaz de realizar la búsqueda del contenido deseado. En el presente trabajo se analizará experimentalmente el desempeño de un nuevo esquema de enrutamiento orientado a contenido que está basado en ordenamientos múltiples basados en prefijos.

1.1. Antecedentes

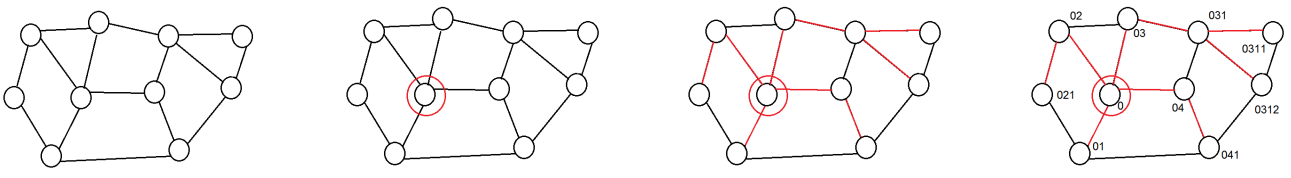
Considerando todos los problemas anteriores, se ha realizado investigación en el campo del enrutamiento compacto y de acuerdo con [35] los algoritmos de solución utilizados en estos esquemas se pueden clasificar en dos categorías.

- Solución explícita: Los nombres y etiquetas asignados a los nodos son arbitrarios y en cada nodo se mantiene información de enrutamiento detallada para cada destino como lose realiza en [14] y [24].
- Solución implícita: No se mantiene información detallada, en cambio, los nombres y las etiquetas son asignadas a los nodos de tal manera que esa información implícita en las etiquetas pueda ser utilizada al momento de tomar las decisiones de enrutamiento. Este tipo de algoritmos son estáticos y descentralizados.

Este trabajo se enfocará en los algoritmos con una solución implícita. Dentro de este tipo de esquemas existe una gran variedad de trabajos realizados, como por ejemplo[41],[36],[31], sólo por hacer mención de algunos. En estos trabajos lo que se hace es montar o sobreponer una red *virtual* sobre la infraestructura física de la red, para así aprovecharla y crear estructuras que ayuden en el esquema de enrutamiento. Dichas estructuras pueden ser mallas, árboles, anillos, caminos, etc. Éstas ayudarán en la asignación de las etiquetas o nombres que recibirán cada nodo para identificarse y al mismo tiempo esas etiquetas ayudarán en las decisiones de enrutamiento de los paquetes para llegar a su destino.

La mayor parte de los trabajos que se estudiaron y que serán descritos con mayor profundidad en capítulos posteriores, montan sobre la red un árbol enraizado en un nodo que pudo haber sido elegido por poseer características que ayudarán en el desempeño del algoritmo o también pudo haber sido un nodo elegido arbitrariamente. Todo el proceso del enrutamiento se realiza siguiendo las aristas que forman al árbol que se creó después de hacer una búsqueda por amplitud o profundidad y que también sirvió para la asignación de las etiquetas o nombres de cada uno de los nodos que forman al árbol. Dichas etiquetas se forman a partir de la relación padre e hijo de los nodos que participan; donde la etiqueta del padre es un prefijo de la etiqueta de los hijos.

Este esquema, que se ilustra en la Figura 1.2, en donde un nodo de la red es el encargado de iniciar el ordenamiento (etiquetamiento), será el punto de partida para el desarrollo de la presente tesis. Se analizarán sus desventajas y se propondrán soluciones para la obtención de un esquema nuevo de enrutamiento compacto usando etiquetas basadas en prefijos.



(a) Grafo (b) Nodo raíz (c) Inducción árbol (d) Etiquetado

Figura 1.2: Esquema de un sólo nodo raíz

1.2. Planteamiento del problema

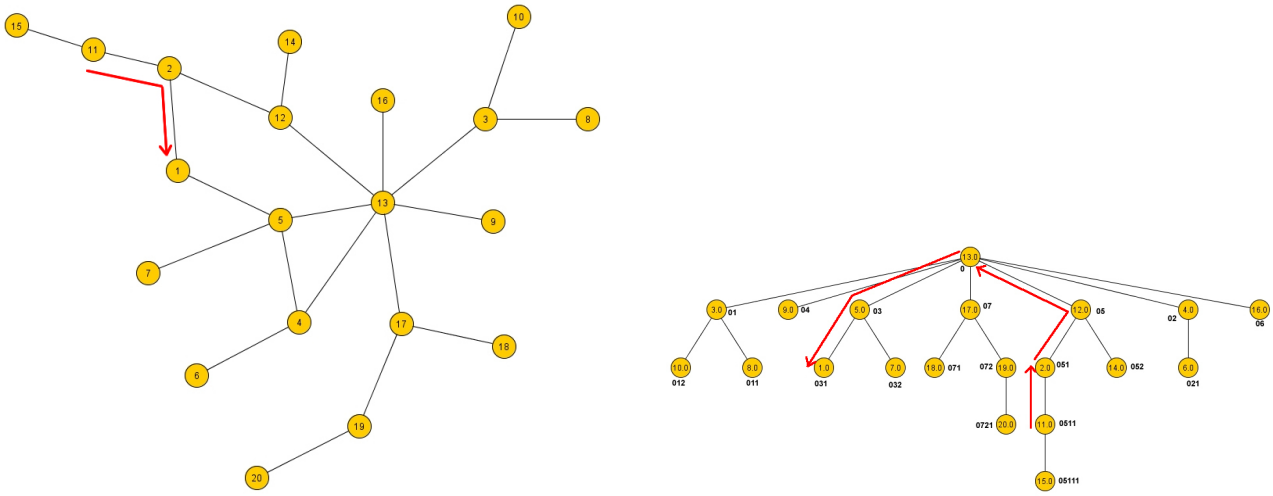
El problema de enrutamiento en una red de comunicaciones, que es modelada con un grafo $G = (V, E)$ donde los nodos $v \in V$ representan computadoras o enrutadores y las aristas $e \in E$ representan enlaces de comunicación bidireccionales entre dos computadoras, consiste en encontrar un conjunto de nodos que formen un camino desde un nodo origen arbitrario a otro nodo destino, también arbitrario. Un camino C válido del nodo s al nodo d es una secuencia de nodos $C = s, n_1, n_2, ..., n_i, d$ tal que para cualquier par de nodos n_j, n_{j+1} consecutivos en el camino C se cumple que $(n_j, n_{j+1}) \in E$. Adicionalmente, se busca que el camino C que conecte a los nodos origen y destino minimice alguna métrica de costo como puede ser la longitud del camino medido en términos del número de nodos que lo componen, el retardo que experimenta un paquete al transitar por dicha ruta, el consumo total de energía requerida para transportar el paquete desde el origen al destino, etc.

Como hemos mencionado, los esquemas de enrutamiento tradicionales basados tanto en vectores de distancia [1], [22], [33] como en el estado de los enlaces [26], [10] tienen la desventaja de que el tamaño de sus tablas de enrutamiento es proporcional al número de nodos en la red, es decir, su complejidad espacial es $\Theta(|V|)$. Lo anterior limita de manera considerable la escalabilidad de los protocolos ya que las consultas a estas tablas pueden ser muy tardadas pero sobre todo por la sobrecarga de control necesaria para mantener actualizadas todas las entradas.

Para subsanar el problema de escalabilidad de los esquemas de enrutamiento tradicional, se han desarrollado esquemas de enrutamiento *compacto* cuyo objetivo es que tanto las tablas de enrutamiento como la sobrecarga de control crezca de manera sublineal con respecto al tamaño de la red, es decir, su complejidad tanto espacial como de mensajes sea $o(|V|)$. En este trabajo abordamos el problema de diseñar, verificar y caracterizar un esquema de enrutamiento compacto que minimice la longitud de las rutas y que balancee la carga a lo largo de la red. El balanceo de carga es importante porque disminuye los problemas de congestion generados cuando múltiples rutas tienden a sobreponerse en ciertas regiones de la red.

En el presente trabajo se propone utilizar un esquema de enrutamiento basado en etiquetas que están compuestas por prefijos. Estos etiquetados inducen un árbol sobre la red que es utilizado para enrutar paquetes desde cualquier origen hasta cualquier destino. A pesar de sus ventajas, esta forma de enrutamiento tiene tres problemas fundamentales que se listan a continuación.

1. Longitud de rutas.
   Aun cuando el *stretch* es algo característico del enrutamiento compacto, al realizarse el enrutamiento sobre el árbol inducido por el etiquetado basado en prefijos, las longitudes de los caminos establecidos entre dos hojas del árbol puede llegar a ser de orden $O(\log(|V|))$ lo cual es relativamente alto. Es importante mencionar que la elección del nodo raíz del árbol de etiquetado es importante, debido a que una mala elección podría generar rutas de tamaño considerable. En la Figura 1.3, se muestra el caso en el que un camino va desde el nodo 11 al nodo 1 en un árbol de etiquetados aun y cuando en el gafo original la ruta más corta entre ese par de nodos es de unos cuantos saltos.



(a) Árbol original (b) Ruta en etiquetado

Figura 1.3: Generacióm de ruta más larga

2. Congestión en los nodos que se encuentran cerca de la raíz del árbol inducido por los etiquetados de prefijos.
   Debido a que los caminos encontrados por el algoritmo de enrutamiento compacto están sobre el árbol inducido por el etiquetado, éstos tienden a concentrarse en regiones cercanas a la raíz. Lo anterior puede propiciar que las regiones de la red que se encuentran cerca del nodo raíz se congestionen con facilidad porque muchos de los caminos calculados por el algoritmo pasarán por ahí. En la Figura 1.4, se muestra que al comunicarse los nodos de las áreas amarillas, los nodos superiores marcados con color verde, serán los nodos que concentrarán ese flujo.

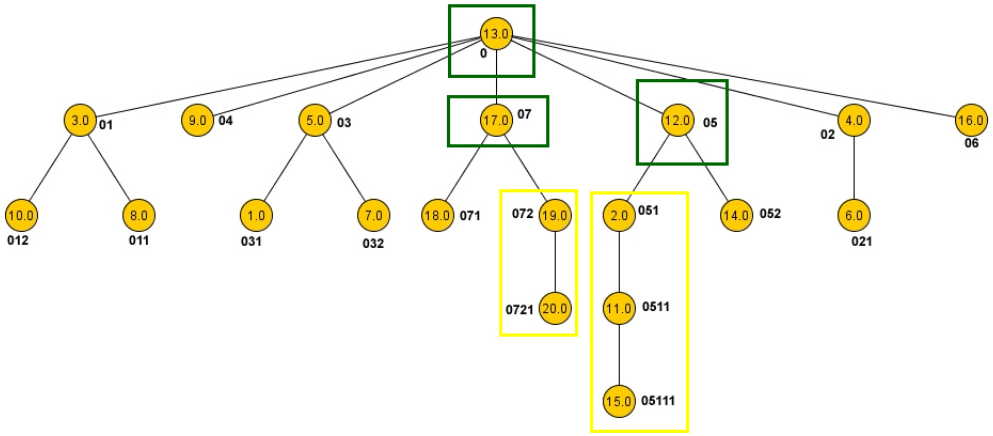


Figura 1.4: Las regiones de la red cercanas a la raíz del etiquetado tienden a congestionarse debido a que muchos de los caminos calculados por el algoritmo de enrutamiento compacto pasan por ahí.

3. Poca tolerancia a fallas.
   Uno de los problemas principales de los esquemas de enrutamiento basados en etiquetados de prefijos es el que se conoce como *tormenta de re-etiquetado*. Una tormenta de re-etiquetado se presenta cuando un nodo cercano a la raíz desaparece de la red, lo que provoca que todos sus descendientes en el árbol tengan que cambiar de etiqueta. En el caso extremo, cuando el nodo raíz desaparece, la red completa tiene que ser re-etiquetada lo cual es costoso en términos de la sobre carga de red. La Figura 1.5 ilustra este problema.

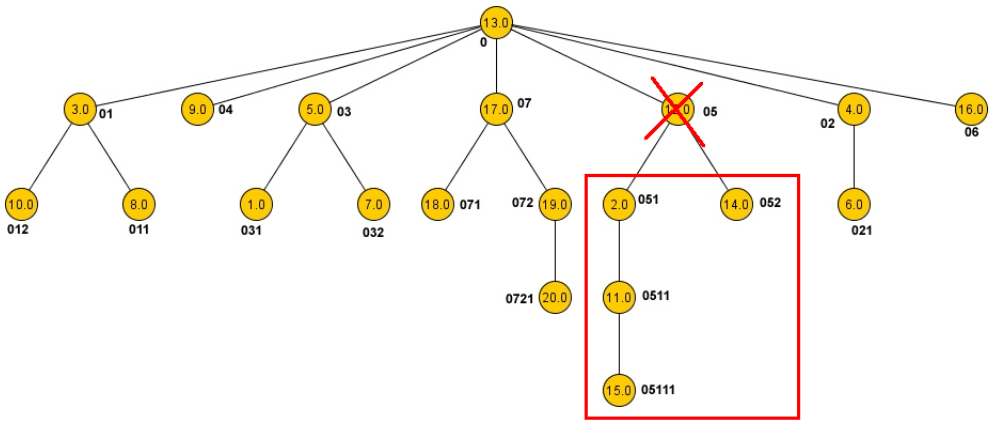


Figura 1.5: Ejemplo de una tormenta de re-etiquetado. Cuando el nodo con la etiqueta "05" desaparece de la red, todos sus descendientes en el árbol de enrutamiento deben obtener nuevas etiquetas.

En la presente tesis se propone desarrollar un nuevo esquema de enrutamiento compacto que sea capaz de reducir el impacto de los problemas antes mencionados, pero manteniendo las características deseables de utilizar tablas de enrutamiento que crezcan de manera sublineal con respecto al tamaño de la red y además que tengan un *stretch* menor que el esquema planteado como base.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar, verificar y caracterizar experimentalmente un algoritmo de enrutamiento compacto usando etiquetas basadas en prefijos.

1.3.2. Objetivos específicos

- Diseñar, verificar y caracterizar una serie de algoritmos de elección de centros de un grafo para los casos de un único centro y dos centros.
- Diseñar, verificar y caracterizar un algoritmo de enrutamiento basado en dos etiquetados de prefijos.
- Realizar un análisis experimental del impacto de la selección de centros en el desempeño de algoritmos de enrutamiento basados en un único etiquetado. Las métricas de desempeño utilizadas son el estiramiento (*stretch*), promedio de las rutas, y el número promedio y máximo de caminos que pasan por un nodo.
- Realizar un análisis experimental del desempeño de los algoritmos de enrutamiento que utilizan múltiples etiquetados.

1.4. Justificación

La presente investigación es relevante desde el punto de vista teórico debido a que se busca experimentar con un esquema novedoso de enrutamiento para redes de computadora de gran escala que está basado en resultados de teoría de grafos y de teoría de algoritmos. Como se ha mencionado, las redes de computadoras han tomado gran importancia gracias a que han cambiado la manera en la cual las personas obtienen información y la manera en cómo se comunican. Lo anterior ha fomentado que cada vez más dispositivos estén conectados a la red global y por lo tanto, que su tamaño crezca aceleradamente por lo que se requieren de nuevos esquemas de enrutamiento específicamente diseñados para redes dinámicas de cientos de miles de nodos.

1.5. Organización de la tesis

El resto del presente documento se organiza de la siguiente manera.

- **Capítulo 1: Introducción.**
  Se hace mención de los objetivos, la justificación y el planteamiento del problema.
- **Capítulo 2: Enrutamiento compacto.**
  Se abordarán los conceptos relacionados al enrutamiento compacto, así como sus características principales y se mostrarán ejemplos de este tipo de enrutamiento.
- **Capítulo 3: Trabajos relacionados**
  Aquí se mostrarán los trabajos estudiados en el estado del arte, así como sus principales aportaciones y características.
- **Capítulo 4: Propuesta**
  Se presenta la solución propuesta y se describirán todos los detalles de la misma.
- **Capítulo 5: Resultados**
  Se exhiben los resultados obtenidos por la implementación de la propuesta.
- **Capítulo 6: Conclusiones**
  Se da un pequeño resumen de la realización del trabajo y se exponen las conclusiones derivadas del trabajo.
- **Bibliografía**

Capítulo 2

Enrutamiento y enrutamiento compacto

Dado que la presente tesis se desarrolla en el contexto de enrutamiento dentro de las redes de computadoras, es necesario contar con fundamentos teóricos dentro de esta área para poder obtener un mayor entendimiento, tanto del problema que se pretende solucionar como de la propuesta de solución.

A continuación, se desarrollan los principales conceptos y definiciones que ayudaron en el desarrollo del trabajo y de los cuales se hablará en capítulos posteriores. Los conceptos están organizados en dos secciones relacionadas con el enrutamiento tradicional y con el enrutamiento compacto.

2.1. Enrutamiento de camino óptimo

En esta sección se presentan los conceptos y definiciones básicas relacionadas con el problema de encontrar una ruta óptima en una red de computadoras. Es importante mencionar, que en este caso el criterio de optimalidad para una ruta es su longitud en saltos, es decir, el número de nodos que tienen que retransmitir un mensaje para que sea entregado al nodo destino.

2.1.1. Red de computadoras

Desde un punto de vista práctico, una red de computadoras puede definirse como un conjunto de elementos de hardware y software que se encuentran conectados entre sí por medio de enlaces de comunicación que pueden ser alambrados o inalámbricos. La finalidad de una red de computadoras es compartir información, recursos y servicios. De acuerdo con [37], existen dos clasificaciones para las aplicaciones de las redes de computadoras:

- Aplicaciones de negocios.
- Aplicaciones domésticas.

En cada una de esas clasificaciones, puede o no haber usuarios móviles, pero sin duda lo que provee a las redes de su funcionalidad son los protocolos de enrutamiento que emplean.

Desde el punto de vista teórico, una red de computadoras puede especificarse por medio de un grafo $G(t) = (V(t), E(t))$ que varía en el tiempo compuesto por un conjunto dinámico de nodos $V(t)$ que están interconectados entre sí por medio de un conjunto también dinámico de aristas $E(t)$ que definen enlaces de comunicación. El conjunto de nodos es dinámico, porque en cualquier momento nuevos nodos pueden unirse o salir de la red. De forma análoga, el conjunto de enlaces de comunicación es dinámico porque en cualquier momento se pueden establecer o perder enlaces entre nodos.

2.1.2. Enrutamiento de camino más corto

Como hemos mencionado, el problema de enrutamiento de camino más corto entre dos nodos $u, v \in V$ consiste en encontrar un camino C compuesto por una secuencia de nodos $u, x_1, ..., x_i, v$ de longitud mínima tal que para cualquier par de nodos x_j y x_{j+1} en V se cumple que el enlace (x_j, x_{j+1}) está en E.

Aun y cuando el enrutamiento de camino más corto es el más comúnmente utilizado en las redes de computadoras, existen otras variantes en las que se establecen funciones de costo sobre los nodos ($c : V \to \mathbb{R}$) o sobre los enlaces ($c : E \to \mathbb{R}$) y donde se busca que el camino $C = u, x_1, ..., x_i, v$ calculado por el algoritmo de enrutamiento sea aquel que minimice $\sum_{\forall x_j \in C} c(x_j)$ para el caso de los nodos y $\sum_{\forall (x_j, x_{j+1}) \in C} c((x_j, x_{j+1}))$ para el caso de las artistas.

2.1.3. Protocolo de enrutamiento

Un protocolo de enrutamiento es aquel que define el esquema de comunicación entre los dispositivos que comunican las redes, es decir, los ruteadores. En otras palabras, un protocolo de enrutamiento define el conjunto de reglas que utilizan los ruteadores cuando se comunican entre ellos con el fin de compartir información de enrutamiento, dicha información se usa para construir y mantener las tablas de ruteo. Otra definición para protocolo de enrutamiento estipula que es la aplicación de un algoritmo de enrutamiento en el software o hardware. El algoritmo de enrutamiento es el encargado de direccionar lógicamente los paquetes a través de varios nodos por los cuales pasarán los paquetes de datos hasta llegar a su destino.

2.1.4. Algoritmo de enrutamiento

En general, un algoritmo de enrutamiento consta de dos componentes. El primero está encargado de recopilar y mantener la información topológica necesaria para realizar el cálculo de las rutas. El segundo componente tiene como objetivo calcular las rutas entre el origen y el destino en base a la información topológica recopilada. Típicamente, ambos componentes se implementan como un módulo del núcleo del sistema operativo o como un demonio del sistema.

Es importante mencionar, que el algoritmo de enrutamiento suele ser independiente del *algoritmo de retransmisión* que es el encargado de enviar un paquete hacia el siguiente salto en su ruta desde el origen al destino. La interacción entre el algoritmo de enrutamiento y el esquema de retransmisión se da por medio de las tablas de enrutamiento. El algoritmo de enrutamiento se encarga de llenar las entradas de esta tabla que para el caso del enrutamiento de camino más corto contiene una entrada por cada destino alcanzable en la red. El algoritmo de retransmisión usa la información contenida en la tabla para enviar el paquete de datos hacía el siguiente nodo hacia el destino. Por razones de eficiencia, el software que implementa el algoritmo de retransmisión se ejecuta como parte del núcleo del sistema operativo.

2.1.5. Protocolos de enrutamiento en redes alambradas

En el caso de las redes alambradas, los protocolos adquieren conocimiento sobre la topología de la red ya sea de forma estática o dinámica. Es necesario recordar que los protocolos de enrutamiento son implementado en los ruteadores que conectan de manera lógica las diferentes redes.

Protocolos de enrutamiento estáticos

Como se ha mencionado, la separación de los grupos de protocolos en estáticos y dinámicos se debe a la manera en cómo se crean y mantienen las tablas de enrutamiento. Un protocolo de enrutamiento estático es aquel que no es capaz de adaptarse a los cambios de topología, al cual hay que introducir de manera manual las rutas por las cuales direccione los paquetes. Las decisiones de qué ruta elegir no se basa en ningún cálculo realizado por el protocolo.

En este tipo de enrutamiento se requiere la participación del administrador de la red ya que es el encargado de crear la tabla de enrutamiento y después de mantenerla, ya que al haber una modificación en la red es necesario actualizar la tabla de ruteo de manera manual para evitar fallos en el envío de paquetes.
¿Cuándo resulta favorable utilizar este tipo de enrutamiento?

- Cuando la red es pequeña.
- Cuando se tiene una sola conexión a un solo ISP o sólo existe un punto de unión en toda la red.
- Cuando no se desea compartir información de enrutamiento.

Principales desventajas del enrutamiento estático:

- Requiere mucha participación del administrador de la red.
- Su complejidad aumenta con el tamaño de la red.
- No responde a los cambios o fallos en la red.

Principales ventajas:

- No utiliza recursos del ruteador (memoria y utilización del CPU).
- Es seguro implementarlo.
- Muy confiable en redes pequeñas.

Protocolos de enrutamiento dinámicos

En este tipo de protocolos la creación de la tabla de enrutamiento se hace de manera dinámica, es decir los ruteadores comparten información acerca de la situación de la red constantemente para así crear y mantener sus tablas de enrutamiento. Lo anterior, para poder responder de manera rápida y dinámica a los cambios sufridos en la red o a posibles fallos en los enlaces dentro de la red y también para obtener información acerca de redes remotas y agregarlas a sus propias tablas. En estos protocolos, las decisiones tomadas hacia dónde mandar los paquetes de datos se hacen con base a los cálculos internos realizados por el algoritmo de enrutamiento.

Como se realizan cálculos internos, es necesario que el ruteador cuente con recursos como memoria dentro de él y de un procesador. Por lo anterior, el administrador de la red no tiene que estar introduciendo las rutas o actualizando la tabla de enrutamiento manualmente ya que el mismo protocolo tiene la capacidad de realizarlo.

Principales ventajas del enrutamiento dinámico:

- Se adapta a los cambios de la topología de la red.
- Responde ante fallos en los enlaces dentro de la red.
- No requiere la participación del administrador de la red.
- Es adecuado para grandes topologías.

Principales desventajas:

- Mayor consumo de los recursos de los dispositivos.
- Menos seguros que los protocolos estáticos.
- Consumo de ancho de banda de la red con los mensajes de control.

2.1.6. Tabla de enrutamiento

La tabla de enrutamiento es un documento electrónico almacenado en cada enrutador o nodo dentro de una red de computadoras. Como hemos mencionado, en el caso del enrutamiento de camino más corto esta tabla tiene una entrada por cada destino de la red en la que se especifica el puerto por el cual deben ser entregados los paquetes dirigidos hacía dicho destino. Las tablas de enrutamiento también cuentan con información extra, dependiendo del protocolo o protocolos que se estén utilizando dentro de la red.

Los protocolos de enrutamiento al ejecutar los algoritmos de enrutamiento, construirán las tablas dentro de cada nodo. Como se menciona en [5] y en [7], este tipo de esquemas de construcción de la tabla de enrutamiento funciona adecuadamente en redes de pocos nodos y se asegura un enrutamiento de paquetes utilizando la ruta más corta. Esto debido a que la cantidad de entradas en la tabla de enrutamiento será de $n - 1$, donde n es el número total de nodos que forman a la red, pero al incrementarse el número de nodos, es importante tomar en cuenta la cantidad de memoria requerida para mantener esta tabla, y conforme la red va aumentando su tamaño, se hace más costoso mantener la tabla y el funcionamiento del protocolo de enrutamiento va degradándose.

2.2. Enrutamiento compacto

A diferencia de los esquemas de enrutamiento tradicional donde se busca que las rutas calculadas por los algoritmos de enrutamiento sean óptimas, pero donde el tamaño de las tablas de enrutamiento es proporcional al tamaño de la red, en el enrutamiento compacto se hace énfasis en la escalabilidad en términos del tamaño de las tablas de enrutamiento y el número de mensajes que son necesarios para poblar dichas tablas aunque se tenga que sacrificar optimalidad en términos de la longitud o costo de las rutas. Este nuevo enfoque surge como respuesta a situaciones donde las redes pueden llegar a contener cientos de miles de nodos en las que es impráctico utilizar enrutamiento tradicional debido a la sobre carga que genera actualizar las tablas de enrutamiento de todos los nodos en la red.

En [19] se define al enrutamiento compacto como *un esquema de enrutamiento que emplea direcciones de tamaño logarítmico, una tabla de enrutamiento de tamaño sublineal y donde el estiramiento (*stretch*) está acotado por una constante.* Como se menciona en la definición, el tamaño de la tabla de enrutamiento es una de las principales características de este tipo de esquemas.

2.2.1. Tabla de enrutamiento compacto

A diferencia de las tablas de enrutamiento de los esquemas tradicionales, las tablas del enrutamiento crecen de manera sublineal con respecto al tamaño de la red, en otras palabras, no habrá una entrada a la tabla por cada nodo dentro de la red, permitiendo así un ahorro de memoria dentro del nodo, haciendo más escalables a los protocolos de enrutamiento compacto. Más aún, los protocolos encargados de mantener actualizada la información de enrutamiento generan menor tráfico de control porque la información de dichas tablas es típicamente local. Lo anterior contrasta con las tablas de enrutamiento tradicional que típicamente incluyen información acerca de toda la red.

Este ahorro, repercute en la longitud de la ruta elegida de un nodo origen a un nodo destino la cual en general no será óptima. A la relación entre la longitud de la ruta calculada por medio del esquema de enrutamiento compacto entre la longitud de la ruta óptima se le conoce como estiramiento (*stretch*). Al igual que el tamaño de la tabla de enrutamiento, el *stretch* es una propiedad característica del enrutamiento compacto, de hecho, es una de las métricas utilizadas para evaluar el desempeño de estos esquemas.

2.2.2. Estiramiento (*Stretch*)

Retomando la definición de *stretch* que se hizo en la introducción de la tesis, se tiene que el *stretch* es el radio máximo entre la longitud de la ruta calculada por el esquema de enrutamiento compacto entre dos puntos y la misma distancia calculada por un esquema de ruta más corta. Entonces, para cada par de nodos en todos los grafos en el conjunto de grafos sobre los que el algoritmo puede operar, encontramos la relación de la longitud de la ruta tomada por el algoritmo a la longitud del camino más corto disponible entre el mismo par de nodos .

El máximo de esta relación entre todos los pares de nodos en todos los grafos, es el *stretch* del algoritmo y está dado por la siguiente ecuación:

$$MAX_{x,y \in N}(\frac{Distancia_{CR}(x,y)}{Distancia_{OPT}(x,y)}) \tag{2.1}$$

Donde x, y son nodos pertenecientes al conjunto de nodos N, y $Distancia_{CR}$ es la distancia calculada por el esquema de enrutamiento compacto y $Distancia_{OPT}(x, y)$ es la distancia mínima entre los nodos x y y.

2.2.3. Esquemas de enrutamiento compacto

Como se ha mencionado a lo largo del trabajo, el principal propósito de los esquemas de enrutamiento compacto es la reducción del tamaño de las tablas de enrutamiento dentro de los nodos y por ende de la cantidad de recursos de cómputo y comunicaciones empleados para ese fin. Para lograr esta reducción, se ha trabajado en diferentes esquemas los cuales han dado como resultado una tabla de enrutamiento más compacta y un enrutamiento dentro de la red.

Para una mejor explicación se emplearán ejemplos basados en redes que pueden ser representadas por grafos conectados. Una red puede ser representada por un grafo conectado $G = (V, E)$ si para cualquier par de nodos $u, v \in V$ existe un camino simple que conecta a u con v compuesto por enlaces $e \in E$ [37].

2.2.4. Enrutamiento jerárquico

Como se describe en [18], en el enrutamiento jerárquico, la idea de reducir el tamaño de la tabla de enrutamiento se realiza manteniendo en cada nodo información de enrutamiento completa acerca de los nodos cercanos a él con relación a la distancia en saltos (o cualquier otra métrica que indique que están cerca) y poca información acerca de los nodos lejanos.

Una manera de implementar lo anterior es tener una entrada a la tabla de enrutamiento por cada nodo cercano y una entrada por cada grupo de nodos lejanos. Para ejemplificar mejor este concepto se mostrarán las siguientes figuras tomadas de [18], donde la figura 2.1 muestra cómo se considera a cualquier nodo como el agrupamiento nivel 0, que a su vez es agrupado con otros nodos para formar el primer nivel de agrupamiento, y posteriormente los grupos de nivel 1 se agrupan para formar el grupo de segundo nivel. Este agrupamiento se realiza de forma sucesiva hasta tener el agrupamiento de nivel m que incluye a todos los nodos.

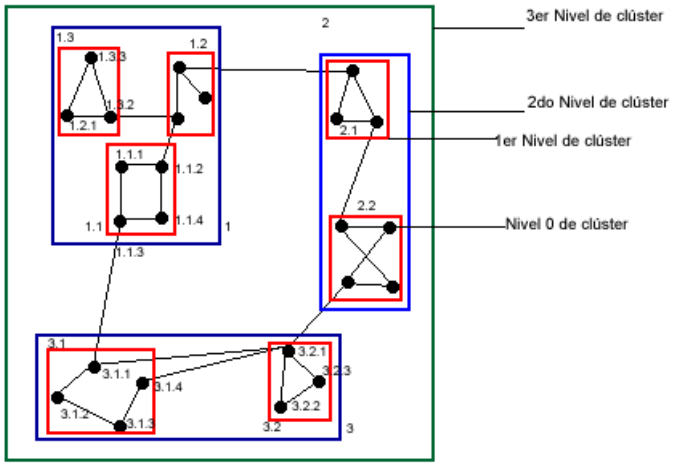


Figura 2.1: Agrupamiento de nivel 3 en una red de 24 nodos.

En la Figura 2.2 se aprecia cómo el agrupamiento de los nodos conlleva a una representación de la red en forma de árbol, donde cada nodo es identificado utilizando la notación decimal de

Dewey. De esta manera podemos asociar una tabla de enrutamiento a cada nodo, por ejemplo, la tabla del nodo 1,1,1 se muestra en la Figura 2.3, y se puede observar que el número de entradas es 10, en comparación de 24 que se tendrían sin realizar las agrupaciones de los nodos.

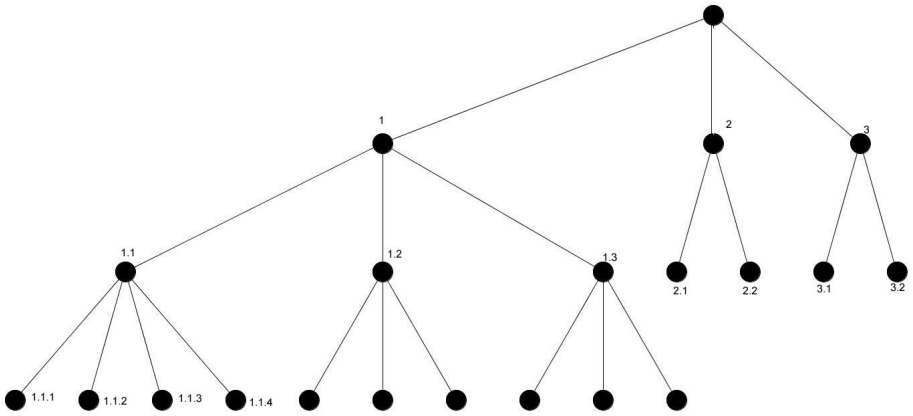


Figura 2.2: Representación por medio de un árbol de una red organizada en un grupo nivel 3.

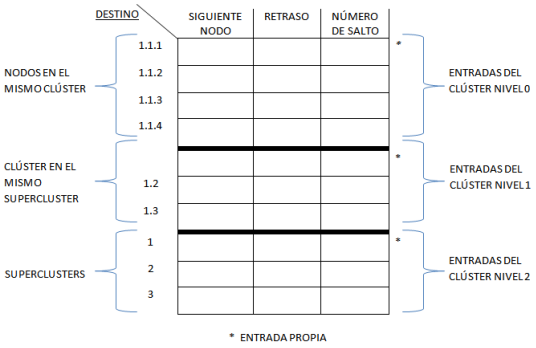


Figura 2.3: Tabla de enrutamiento del nodo 1.1.1.

2.2.5. Etiquetado de árboles

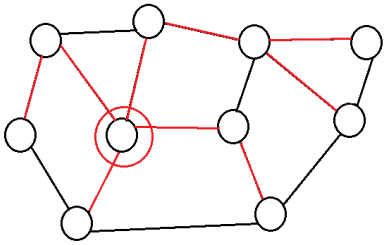
Como se mostró en la sección anterior, el uso de árboles para la representación de las redes es muy común. Muchos esquemas de enrutamiento compacto utilizan árboles lógicos sobrepuestos o montados sobre la infraestructura de las redes. Para poder identificar a cada nodo, se le asigna un identificador único, al cual se le llamará etiqueta como se menciona en [35]. Dentro de estas redes lógicas, también se puede etiquetar los puertos de salida de cada nodo. Ésto brinda información de manera implícita al esquema de enrutamiento.

Para la asignación de las etiquetas, lo que generalmente se realiza es una búsqueda por amplitud o una búsqueda por profundidad comenzando por el nodo que será la raíz del árbol lógico. A continuación, conforme se van agregando nodos al árbol, se les asigna una etiqueta que servirá para su identificación y además ayudará a facilitar el enrutamiento dentro del árbol.

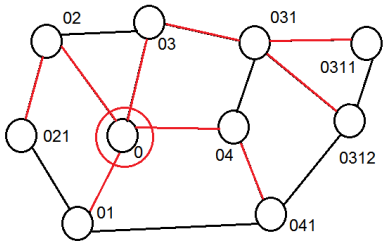
Las etiquetas asignadas a los nodos o puertos generalmente son números enteros asignados en secuencia conforme se van encontrando nuevos nodos. La clasificación decimal de Dewey, utilizada en las clasificación de los libros en una biblioteca también puede ser utilizada para etiquetar nodos en una red, una tercera opción sería tener una función dispersión (*hash*) que

sea la encargada de mapear de un espacio de identificadores a un espacio de etiquetas.

Lo importante del etiquetado de los nodos dentro de un árbol es poder identificarlos. En la primera parte de la Figura 2.4 se muestra la inducción de un árbol lógico dentro una red de 10 nodos. Las aristas pertenecientes al árbol se encuentran resaltadas en color rojo, el nodo raíz es el nodo encerrado en un círculo rojo. En la segunda parte de la figura se muestra la asignación de las etiquetas, el etiquetamiento también comienza desde la raíz.



(a) Inducción del árbol lógico



(b) Etiquetado

Figura 2.4: Inducción y etiquetado de un árbol lógico en una red.

El enrutamiento dentro de los árboles inducidos en la red con un etiquetado para distinguir a los nodos, se realiza salto a salto buscando siempre la mayor coincidencia de la etiqueta destino con la etiqueta del puerto o del vecino al que se le envía el paquete de datos. Cada nodo al recibir el paquete de datos buscará entre sus vecinos o en sus puertos la etiqueta que más coincida con la etiqueta a la cual el paquete va dirigida y es hacia ese lugar a donde re-envía el paquete, esta acción la realiza cada nodo hasta llegar al nodo que cuenta con la etiqueta destino. En caso de que la etiqueta no exista, el paquete se quedará en el nodo que cuente con la etiqueta más parecida a la etiqueta destino.

2.2.6. Enrutamiento por intervalos

El enrutamiento por intervalos es otro esquema que permite la creación de tablas de enrutamiento de tamaño compacto y que a su vez emplea un etiquetamiento de los nodos para facilitar el enrutamiento. Para la definición de este esquema nos basaremos en[4] donde se especifica que :*es una manera de implementar esquemas de enrutamiento en redes arbitrarias. Se basa en representar las tablas de enrutamiento almacenadas en cada nodo de una manera compacta. Lo anterior se logra al agrupar el conjunto de direcciones destino que utilizan el mismo puerto de salida en intervalos de direcciones consecutivas.*

Las etiquetas son asignadas a los nodos y los intervalos de direcciones se asignan a los puertos de salida de los nodos. Cuando un paquete llega a un nodo, éste se encarga de revisar el identificador o etiqueta de destino, si es igual al que posee él, el paquete ha llegado a su destino, si la etiqueta no es la misma que la del nodo actual, entonces, el nodo decidirá hacia

cuál puerto reenviar el paquete de acuerdo con los intervalos establecidos en cada puerto. Ésto pasará con cada nodo intermedio hasta llegar a su destino. Dentro de los intervalos se agrupan cierta cantidad de destinos, para ayudar en las decisiones de enrutamiento los intervalos se definen de a continuación: En una red con N nodos $0, 1, ...N-1$ el intervalo $[p, q)$ entre los nodos p y q define a los identificadores de los nodos que estarán entre ellos de la siguiente manera: si $p < q$, entonces $[p, q) = p, p+1, p+2, ...q-2, q-1$, en cambio si $p \geq q$, significa que $[p, q) = p, p+1, ...N-1, 0, 1, ...q-2, q-1$. Por lo tanto el nodo sólo deberá ver dentro de qué intervalo se encuentra el identificador destino y en qué enlace se encuentra el intervalo par re-enviar el paquete de datos.

En la imagen de la Figura 2.5 se puede observar cómo es el proceso de enrutamiento de un nodo dentro de una red. Se distingue que su tabla de enrutamiento sólo contará con 3 entradas que es igual al grado del nodo. En la Figura 2.6 se aprecia un esquema de enrutamiento por intervalos, tanto los nodos como los puertos han sido etiquetados para poder realizar en el enrutamiento dentro de la red. Los números en los enlaces representan los límites de los intervalos definidos, por ejemplo en en enlace que comparte el nodo 0 con el nodo 7, el nodo 0 establece $[7, 0)$como intervalo para ese enlace . Ambas imágenes fueron tomadas de [12].

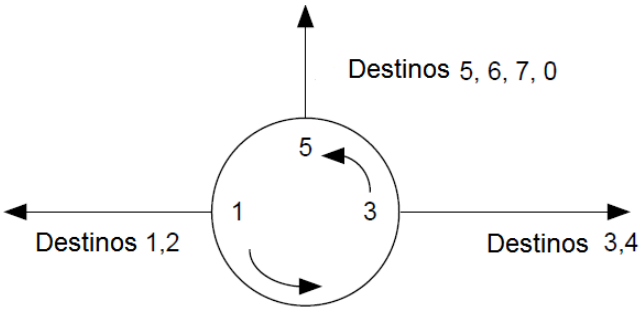


Figura 2.5: Ejemplo del reenvío de mensajes en el enrutamiento por intervalos. En la red hay 8 nodos numerados del 0 al 7.

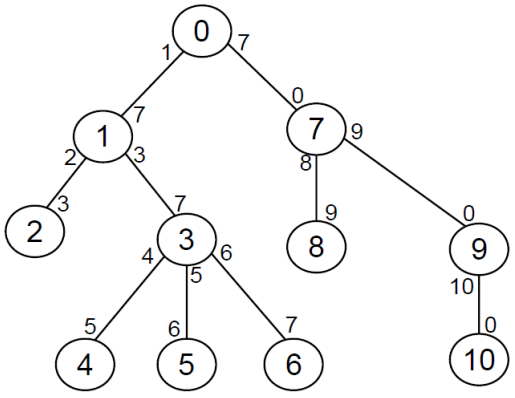


Figura 2.6: Esquema de enrutamiento por intervalos en una red representada por un árbol.

2.2.7. Enrutamiento por prefijos

En el los esquemas de enrutamiento basados en prefijos, tanto los identificadores asignados a los nodos en forma de etiquetas, así como los identificadores asignados a los objetos dentro de la red son tratados como cadenas para el proceso de enrutamiento. Un nodo al recibir un paquete verificará la cadena que representa la dirección destino, después

hará una búsqueda dentro de su tabla de enrutamiento para comparar el destino con las entradas de la misma, la etiqueta puede ser encontrada tal cual en una de las entradas y el nodo solamente tendrá que enviar el paquete por el puerto indicado en esa entrada. En caso de que no encuentre la del destino dentro de su tabla, se buscará la cadena que más se le parezca y por el puerto indicado en esa entrada será por el que se reenviará el paquete.

Este esquema tampoco almacena todos los destinos de la red, al ser un esquema de enrutamiento compacto también "sacrifica" información de la tabla de enrutamiento por ganar espacio de memoria y reducir la sobre carga de control. Al no contar con toda la información de la red, este tipo de enrutamiento también encontrará rutas de tamaño mayor a la óptima. Este tipo de enrutamiento será retomado en los trabajos estudiados en el estado del arte del siguiente capítulo y posteriormente en al capítulo 4 se presentará una propuesta de solución que emplea al enrutamiento basado en prefijos para el problema planteado en el primer capítulo.

2.2.8. Redes orientadas a contenido

Las redes orientadas a contenido[3] (*Content-Oriented Networks*, CON por sus siglas en inglés) son una alternativa propuesta a la arquitectura actual de las redes de computadoras, siendo su principal característica que las comunicaciones en las redes deben permitir a los usuarios enfocarse en el contenido que necesita, en lugar de tener que hacer referencia a la ubicación física de donde se obtienen los datos.

Características
El enrutamiento se realiza por contenido y no por la ubicación del *host* por lo cual:

- La identificación de los nodos se remplaza por una identificación del contenido.
- La localización del contenido es independiente de su nombre.
- El modelo base para este tipo de redes es el paradigma de publicación/suscripción. Ésto significa que la fuente de un contenido anuncia (publica) un objeto de datos o contenido, mientras que un usuario solicita (suscribe) al archivo con dicho contenido. Con este paradigma se puede disociar la generación del contenido con su consumo, tanto en tiempo como en espacio, por lo que el contenido es entregado de manera eficiente y escalable.

Para nombrar al contenido dentro de estas redes se han desarrollado varias técnicas sobre éste, entre las cuales se destacan nombrarlo en orden jerárquico, plano y basado en atributos, siendo el formato plano el más característico, aquí se utilizan nombres planos y en algunos casos de auto-certificación mediante la definición de un identificador de contenido, tal como una llave *hash* o una llave pública, una de sus características es que debido a que utiliza un nombre plano, se tiene que implementar un paso intermedio en el proceso para que se traduzca de un nombre plano a un texto inteligible para que el usuario lo pueda identificar.

La publicación de contenido se realiza de la siguiente manera: el nodo que desea introducir un objeto de datos, debe aplicar la función *hash* al identificador del contenido, una vez realizado esto, el resultado obtenido es el identificador del nodo que será el nodo ancla de ese contenido o su representante. A ese nodo le debe reportar en dónde se encuentra el contenido montado de tal manera que el nodo ancla pueda responder con esa ubicación a las solicitudes de los demás nodos cuando deseen consumir ese contenido en específico.

Para la ubicación del contenido, una cadena de texto, generalmente la dirección URL del contenido es introducida como entrada a una función *hash* la cual será la encargada de mapear del espacio de identificadores a un espacio de tamaño fijo como lo es el espacio de los identificadores de los nodos dentro de la red. Dicha función es conocida por todos los nodos y utilizada para poder obtener la identidad de los nodos ancla, éstos son los nodos que conocen la ubicación del contenido montado sobre la red. Mediante la implementación de esta función *hash* es posible dejar de hacer uso del servicio de los sistemas de nombre de dominios ya que la función que realizan dichos sistemas ahora se encuentra distribuida en la red. A pesar de que en este tipo de redes no es necesario contar con nodos especiales que se encarguen de conocer todas las ubicaciones de todo el contenido dentro de la red, si es necesario contar con el servicio de consultas por contenido (un buscador), éste seguirán proporcionando el mismo servicio, con la diferencia que ahora el resultado que se tendrá serán los identificadores concretos de los objetos de datos para que posteriormente cada nodo pueda aplicar la función *hash* y pueda conocer la ubicación del contenido en cuestión, mediante su respectivo nodo ancla. Para ejemplificar la manera en que se obtiene el contenido en este tipo de redes, utilizaremos como modelo la red mostrada en la Figura 2.7 en donde se aprecia que cuenta con un etiquetado el cual induce un árbol lógico en la red. El etiquetado es el espacio de identificadores de los nodos al cual la función *hash* mapeará desde los identificadores del contenido. En la red, el nodo que cuenta con la etiqueta 021 almacena un objeto de datos, cuyo nombre es "http://nuevastecsomamfyc.files.wordpress.com/2013/01/video.jpg". Por lo tanto, para publicar el contenido debe aplicar la función *hash* para obtener la identidad del nodo ancla. En este ejemplo el nodo ancla es el nodo con la etiqueta 03111, al cual el nodo 012 le hace una consulta para saber en qué nodo está el contenido anteriormente mencionado. Esa solicitud recorre el camino indicado con flechas verdes, debido a que el enrutamiento basado en prefijos busca la mayor coincidencia en cuanto a las etiquetas. Después que el nodo 012 obtiene la ubicación del contenido, simplemente enruta la solicitud al nodo 021, este camino se indica con las flechas azules. Las respuestas a ambas solicitudes siguen el mismo camino pero en sentido contrario debido a que se sigue la misma lógica de enrutamiento de mayor coincidencia de las etiquetas en cada salto del camino.

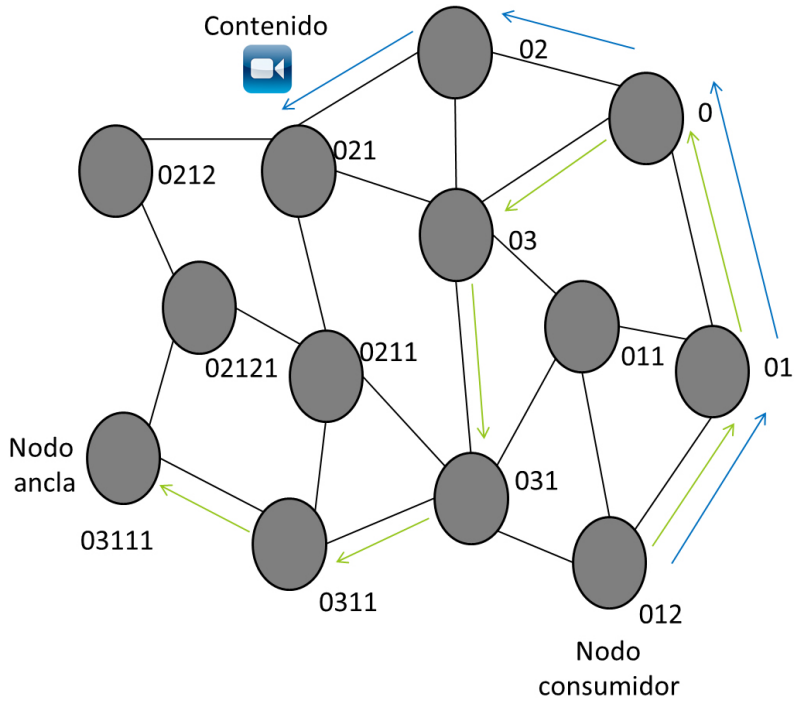


Figura 2.7: Enrutamiento orientado a contenido.

2.3. Redes superpuestas

Una red superpuesta *(overlay network)* es una red que crea una topología virtual sobre una topología física [6]. En otras palabras, es un conjunto de nodos y enlaces lógicos sobre la infraestructura de una red física con el propósito de implementar un servicio sobre la red con el cual no cuenta.

Un ejemplo muy popular de esta clase de redes es el Internet. El Internet es un conjunto de redes de diferentes tipos interconectadas entre sí. Todas estas redes funcionan como una única red lógica y esto se debe a los protocolos que funcionan en ella. La Figura 2.8 se utiliza para mostrar el concepto de red superpuesta.

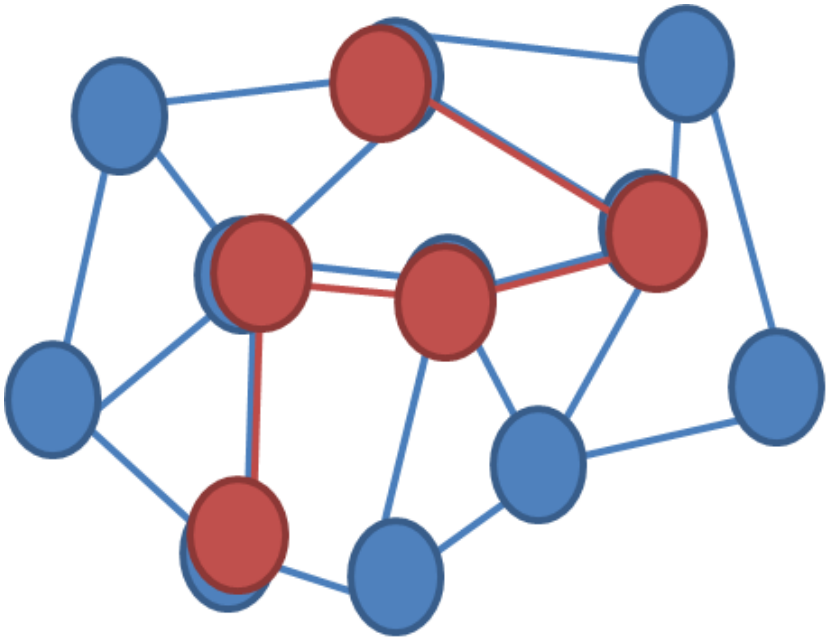


Figura 2.8: Red superpuesta. Tanto los nodos como los enlaces de color rojo forman la red lógica y utilizan la infraestructura de la red azul para funcionar.

2.4. Redes *peer to peer*

Una red entre iguales (*Red peer to peer*) es un tipo de red en la que no existe algún nodo que realice una función extra (servidor) de las que pueden realizar los demás. Es decir, todos los nodos sirven como clientes y servidores al momento del intercambio de información. La mayoría de estas redes suelen sobreponerse sobre redes ya existentes. Este tipo de redes suele utilizarse para compartir archivos de cualquier tipo, debido a que el ancho de banda suele ser administrado, aprovechado y optimizado entre todos los usuarios de la red, logrando así un mejor rendimiento de la red.

Las principales características que comparten este tipo de redes de acuerdo con [6] son las siguientes:

- Garantizan la recuperación de información.
- Su tiempo de búsqueda es demostrable.
- Balanceo de carga automático.
- Auto organización.

Otras características que se buscan dentro de estas redes son las siguientes:

- **Escalabilidad.** Esta característica es deseable debido a que lo que se busca en una red de este tipo es que se agreguen una gran cantidad de usuarios para poder compartir los recursos de cada uno y así aumentar el funcionamiento total de la red.
- **Robustez.** Se busca que las redes entre iguales sean capaces de reaccionar ante la falla de los nodos en ella. Se busca que los objetos de datos estén siempre disponibles para cualquier usuario.
- **Descentralización.** El principal objetivo es que todos los nodos realicen las mismas funciones dentro de la red, ningún nodo deberá contar con más tareas que el resto.
- **Distribución de costes entre usuarios.** Todos los usuarios dentro de la red comparten por igual recursos; estos pueden ser ancho de banda, archivos, procesamiento o almacenamiento.
- **Anonimato.** En estas redes se desea que tanto el creador del contenido, el lugar donde se almacena, el usuario que realiza la búsqueda de ese contenido y otros detalles del propio contenido queden de manera anónima.
- **Seguridad.** La seguridad es una de las características que se buscan en estas redes, ya que no se quiere contenido malicioso, nodos espías, o cualquier otra cosa que pueda afectar el funcionamiento de la red.

2.5. Tabla hash

Para poder entender lo que es una tabla *hash* distribuida empezaremos con el concepto básico de una tabla *hash*. Una tabla *hash* es una estructura de datos que asocia llaves o claves con valores y la búsqueda es la operación principal que soporta de manera eficiente. Ahora bien pasaremos a lo que es una tabla *hash* distribuida, esto debido a que juega un papel importante en los esquemas de enrutamiento compacto al ser la responsable de la asignación de las etiquetas de los nodos al relacionarlas con los identificadores o cualquier otra característica única de los nodos. El llenado de una tabla *hash* se realiza mediante una función *hash*. Una función *hash* tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija.

2.5.1. Tabla *hash* distribuida

Tomando en cuenta las definición de [20], las tablas *hash* distribuidas son servicios de almacenamiento distribuidos que utilizan una red superpuesta estructurada dependiente en los protocolos de enrutamiento basados en claves. En las tablas *hash* distribuidas, a cada nodo y cada bloque de información se le asigna una clave o identificador. El identificador del bloque de información generalmente es el resultado de la función *hash* aplicada al mismo bloque. Cada bloque de información se asocia a un nodo raíz cuyo identificador es el más cercano numéricamente.

Dentro de las tablas *hash* distribuidas resaltan la siguientes propiedades:

- **Descentralización.** No se necesita un nodo que funja como coordinador del sistema.
- **Escalabilidad.** Sin importar que el sistema cuente con un gran número de nodos, éste debe seguir funcionando de manera eficiente.
- **Tolerancia a fallas.** El sistema debe de ser capaz de reaccionar y seguir funcionando a pesar de que haya nodos añadiéndose, partiendo o fallando.

En [28] se mencionan los parámetros utilizados para la medición de las tablas *hash* distribuidas y son las siguientes métricas:

- **Costo de *join/leave*.** El servicio debe adaptarse a los cambios fácilmente. Una pequeña cantidad de nodos deberán de cambiar su estado al unirse o abandonar la red.
- **Congestión.** Ningún nodo debe ser un cuello de botella para en el rendimiento del servicio. La carga incurrida por búsquedas dentro del sistema deberá ser distribuida equitativamente entre todos los nodos.
- **Longitud de la ruta de búsqueda.** La ruta de transmisión de una búsqueda deberá incluir el menor número de nodos.
- **Tolerancia a fallos.** El servicio deberá continuar a pesar de las fallas de nodos o enlaces.
- **Almacenamiento de caché dinámico.** Se busca que el contenido muy solicitado sea replicado en otros nodos para evitar cuellos de botella en los vecinos del nodo que almacena dicho contenido.

2.6. Algoritmos básicos para redes

A continuación se describirán algunos conceptos utilizados en teoría de grafos y algoritmos empleados en grafos. Como se mencionó anteriormente, un grafo puede ser utilizado para la representación de una red y partiendo de un grafo podemos obtener información importante. Esta información será manejada a través de la utilización de matrices.

2.6.1. Estructuras de datos para la representación de grafos

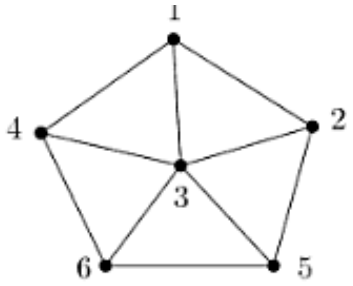
Matriz de adyacencia

Tomando de referencia a [23], la definición de matriz de adyacencia es:
Sea $G = (V, E)$ un grafo con n vértices. Se denotan los vértices por $v_1, v_2, ..., v_n$ (en algún orden arbitrario). La matriz de adyacencia de G, con respecto a la numeración de vértices elegida es una matriz de nXn, $A_G = (a_{ij})_{i,j=1}^{n}$ definida mediante la siguiente regla:

$$a_{ij} = \begin{Bmatrix} 1, & \{v_i, v_j\} & \in E \\ 0, & \{v_i, v_j\} & \notin E \end{Bmatrix}$$

Entonces, una matriz de adyacencia siempre será una matriz cuadrada donde los renglones y las columnas representan a los nodos de la red, con 0´s y 1´s, donde un 1 representa la existencia de un enlace entre el nodo con el número de la columna con el nodo del número de renglón. El 0 representará que no existe conexión entre los nodos y además siempre habrá ceros en la diagonal principal. Otra característica de la matriz es que es simétrica, pero esto sólo aplica si el grafo es no dirigido.

En la Figura 2.9 se muestra un ejemplo de la representación de una red de 5 nodos mediante un grafo no dirigido, en el grafo hay 5 nodos que se encuentran conectados mediante aristas. En la segunda parte de la imagen se tiene la representación del grafo en forma de una matriz de adyacencia.



(a) Grafo G no dirigido con 5 nodos

$$A_G = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

(b) Matriz de adyacencia para el grafo G

Figura 2.9: Grafo G de 5 nodos y su representación en una matriz de adyacencia A_G.

Matriz dispersa

Una matriz es un objeto matemático utilizado en la informática para almacenar datos. La implementación es un arreglo bidimensional que constan de m renglones y n columnas. De acuerdo con [32], una matriz dispersa es una matriz en la que una gran parte de sus elementos son ceros ($0's$). Al tener una gran cantidad de datos en 0 y si sólo aquellos elementos que son distintos a cero son los de importancia para la realización de cálculos o para el funcionamiento de algoritmos, no es necesario (ni recomendable) almacenar la matriz completamente, es por eso que se han buscado maneras de representar a estas matrices de manera más eficiente y así ahorrar memoria y procesamiento. Estas representaciones sólo mantendrán a los valores diferentes de cero en ellas.

La representación que se utilizará en el desarrollo del trabajo es la representación por medio de una tripleta de valores $i, j, valor$. En esta representación, las tripletas representan en su primer campo al renglón i, en el segundo, el valor correspondiente a la columna j y en el tercer campo, el valor almacenado en la celda i, j.
A continuación se mostrará un pequeño ejemplo de la representación por medio de tripletas de una matriz dispersa. En la Figura 2.10 está una matriz de $10x10$, donde sólo hay 10 valores diferentes a 0, por esa razón se considera que la matriz A es una matriz dispersa. En la Figura 2.11 se observa la representación de la matriz A por medio de tripletas de valores del tipo $i, j, valor$. Esta representación facilita el almacenamiento y procesamiento de la misma información en la matriz A.

```
A =

     0     0     0     0     0     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     1     0     0     0     0     0     0
     0     0     1     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
     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     1
     0     0     0     0     1     0     0     0     0     0
```

Figura 2.10: Matriz A considerada una matriz dispersa.

```
A =

   (3,1)        1
   (7,2)        1
   (5,3)        1
   (4,4)        1
  (10,5)        1
   (1,6)        1
   (6,7)        1
   (2,8)        1
   (8,9)        1
  (9,10)        1
```

Figura 2.11: Representación de la matriz A por medio de una representación en tripletas.

Lista de adyacencia

Si bien se dijo que una red puede ser representada por un grafo, existen otras maneras de representarlo también, otro ejemplo es una lista de adyacencia. La lista de adyacencia representa las aristas que unen a los nodos del grafo en una lista. En [27], una lista de adyacencia se define como *listas enlazadas, una para cada nodo, conteniendo los nombres de los nodos a los que el nodo está conectado.*

De esta representación podemos obtener la misma información acerca del grafo que de la matriz de adyacencia. La principal ventaja de la lista de adyacencia sobre la matriz de adyacencia es el ahorro en memoria y en procesamiento al momento de implementarlas. En las Figuras 2.12 y 2.13 se muestran 2 formas distintas de representar al mismo grafo.

```
grafo =
     0     0     0     0     1     0     0     0     1     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     1     0     1     0     0     0     0     0     0     0     0     1     0     0     0     0
     0     0     1     0     1     1     0     0     0     0     0     0     0     0     0     0     0
     1     1     0     1     0     0     1     1     0     1     1     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     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     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
     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     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     0     0     0     0     1     0     0
     0     0     1     0     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     1     0     0     0     0
     0     0     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     0     0     0     1     0     0
     0     0     0     0     0     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     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

Figura 2.12: Representación de un grafo no dirigido de 19 nodos en forma de matriz de adyacencia.

1	[5,9]
2	[3,5]
3	[2,4,13]
4	[3,5,6]
5	[1,2,4,7,8,10,11,12]
6	4
7	5
8	5
9	1
10	5
11	5
12	[5,15]
13	[3,14]
14	13
15	[12,16,17]
16	15
17	[15,18,19]
18	17
19	17

Figura 2.13: Representación de un grafo no dirigido de 19 nodos en forma una lista de adyacencia.

2.6.2. Algoritmo Dijkstra

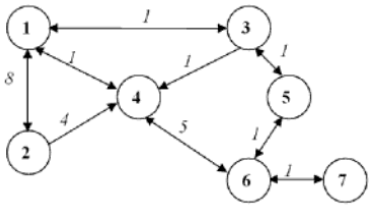
Este algoritmo también llamado algoritmo de caminos cortos, es un algoritmo escrito por Edsger Dijkstra en 1959. El objetivo del algoritmo es encontrar el camino más corto de un nodo origen dado hacia todos los demás nodos de la red y esto lo realiza tomando en cuenta el costo de atravesar los enlaces en los caminos encontrados, es decir, toma el costo de la arista como métrica para la elección de las rutas más cortas. En consecuencia, la ruta óptima será la ruta cuya suma de todos los valores de las aristas que conforman la ruta sea la menor.
Este algoritmo es de tipo *greedy*, ya que en cada iteración dentro de la ejecución, elige la mejor de las posibles soluciones, esto con la esperanza de lograr obtener la mejor solución global. De acuerdo con [30] y [39], para que el algoritmo se comporte de manera correcta, el grafo de entrada podrá o no estar ponderado (que las aristas tengan un costo asociado), pero no podrá haber valores negativos ni ciclos negativos, ya que al existir éstos el algoritmo arrojará resultados incorrectos. A continuación se mostrará el algoritmo presentado en [39] en sus 4 etapas.

1 Paso1: *Inicio*: $T = s$ *y* $L(n) = w(s, n)$, *con* $n \neq s$. ;
2 Paso2: *Nodo siguiente:*;
3 Encontrar el nodo $x \in N$, $L(x) = min\ L(j)$ *con* x, j *no* $\in T$.;
4 Añadir x a $T : T\{s, ..., x\}$;
5 Paso3: Actualizar el camino de coste mínimo.;
6 Calcular $L(n) = min\{L(n), L(n) + w(x, n)\}$ con $'n'$ no $\in\ T$.;
7 Paso4: Finalizar cuando todos los nodos han sido añadidos a T.

Algoritmo 1: Etapas del algoritmo de Dijkstra

Donde
$n \in N$: Conjunto de nodos en la red.
s: nodo origen.
T: Lista de nodos incorporados por el algoritmo.
$w(i, j)$: Coste del enlace directo del nodo i al nodo j.
$L(n)$: Coste del curso desde el nodo s al nodo n.

La implementación básica del algoritmo, es decir sin utilizar una cola de prioridades se ejecuta en un tiempo de $O(|E| + |V|^2) = O(|V|^2)$, donde E es el conjunto de vértices dentro de la red. La Figura 2.14 muestra el ejemplo tomado de [39], donde se aprecia un grafo ponderado no dirigido y en la parte derecha se ve las iteraciones de la ejecución del algoritmo de Dijkstra.



(a) Grafo

i	T	L(2)	Ruta	L(3)	Ruta	L(4)	Ruta	L(5)	Ruta	L(6)	Ruta	L(7)	Ruta
1	1	8	1-2	1	1-3	1	1-4	∞	-	∞	-	∞	-
2	1,3	8	1-2	1	1-3	1	1-4	2	1-3-5	∞	-	∞	-
3	1,3,4	5	1-4-2	1	1-3	1	1-4	2	1-3-5	6	1-4-6	∞	-
4	1,3,4,5	5	1-4-2	1	1-3	1	1-4	2	1-3-5	∞	-	∞	-
5	1,3,4,5,6	5	1-4-2	1	1-3	1	1-4	2	1-3-5	3	1-3-5-6	∞	-
6	1,3,4,5,6,7	5	1-4-2	1	1-3	1	1-4	2	1-3-5	3	1-3-5-6	4	1-3-5-6-7
7	1,2,3,4,5,6,7	5	1-4-2	1	1-3	1	1-4	2	1-3-5	3	1-3-5-6	4	1-3-5-6-7

(b) Ejecución del algoritmo

Figura 2.14: Ejemplo de ejecución del algoritmo de Dijkstra

2.6.3. Algoritmo de búsqueda por amplitud (BFS)

Algoritmo de búsqueda por amplitud (*Breadth-first search o BFS* por sus siglas en inglés)es un algoritmo que se emplea para el recorrido de grafos o en búsquedas sobre los mismos, siendo principalmente árboles en donde se emplea.

El algoritmo funciona de la siguiente manera, se comienza sobre un nodo cualquiera, si se trabaja sobre árboles empieza desde la raíz, después se visita a los nodos vecinos de la raíz. Posteriormente para cada uno de los vecinos de la raíz, se visitarán sus vecinos y así hasta que se hayan visitado todos los nodos. Descrito de la manera en que está presentado en [15], *Breadth-First Search* visita todos los nodos del grafo $G = (V, E)$ que están a k aristas del nodo raíz antes de visitar a los nodos que se encuentran a $k + 1$ arístas de la raíz. Este proceso se realiza hasta que no haya más nodos por visitar. El algoritmo no visitará los nodos que no son alcanzables por la raíz.

Este algoritmo arroja al final un árbol cuya raíz será el nodo con el que se comenzó el recorrido, también estarán todos los nodos visitados y la distancia desde la raíz a cualquier nodo será la distancia más corta ya que el camino de la raíz a cualquier nodo contendrá el mínimo número de aristas. Debido a que el algoritmo toma en cuenta todos los nodos del grafo y sus aristas, la complejidad está dada por $O(|V| + |E|)$ y el algoritmo funciona con grafos dirigidos, grafos no dirigidos y además emplea una cola de datos para su ejecución. A continuación se mostrará el pseudocódigo del algoritmo.

```
Require: Un grafo G = (V, E) y un nodo fuente s
    BFS(G, s) {
    for u ∈ V[G] do
    {
         estado[u]=NO_VISITADO;
         distancia[u]=INFINITO;
         padre[u]=NULL;
    }

    estado[s]=VISITADO;
    distancia[s]=0;
    encolar(Q,s);
    while !vacia(Q) do
    {       u=extraer(Q);
        for v ∈ adyacencia[u] do
        {           if estado[v]==NO_VISITADO then
                        {
                         estado[v]=VISITADO;
                         distancia[v]= distancia[u]*1;
                         padre[v]= u;
                         encolar(Q,v);
                        }
        }
    }
     }
```

Algoritmo 2: Breadth-first search

2.6.4. Algoritmo Floyd-Warshall

El algoritmo Floyd-Warshall [9] fue publicado en 1962 por Robert Floyd, su trabajo se basó en el teorema de Warshall (en ese mismo año). Este algoritmo se emplea en en análisis de grafos y específicamente se utiliza para la obtención de los caminos más cortos entre todos los pares de nodos en un grafo. A pesar de que el algoritmo compara todos los caminos entre un par de nodos y hace esto para todas las combinaciones de pares, el tiempo que toma para realizar estos cálculos es relativamente corto $O(|V|^3)$ y esto es gracias a que utiliza la metodología de la programación dinámica.

El algoritmo trabaja sobre grafos dirigidos ponderados, los costos de las aristas pueden ser positivos y negativos pero para el correcto funcionamiento del algoritmo es necesario que en el grafo no existan algún ciclo de costo negativo. La representación más natural de un grafo para el algoritmo es una matriz de adyacencia. Se asume que se da un grafo dirigido ponderado de la manera $G = (V, E)$ y que los vértices o nodos en V están enumerados $1, 2, 3, ..., n$. Con base en eso, se construye una matriz de la siguiente forma $C[i, j]$, donde (i, j) indica el costo de la arista. Si en $C[i, j]$ no existe una arista, se asume que el valor estará asignado a infinito. A continuación se presenta el algoritmo Floyd-Warshall obtenido de [13], que es en donde se emplea lo anterior.

Require: Un grafo dirigido de la forma $G = (V, E)$ donde $V = \{1, 2, ..., n\}$ y una matriz de costos $C[i, j]$.
Ensure: Una matriz de costos $A[1...n, 1...n]$ en donde $A[i, j]$ es el costo del camino más corto del nodo i al nodo j.

```
Procedimiento Floyd-Warshall (G){
for i:=1 to n do
    for j:=1 to n do
        A[i,j] := C[i,j];
for i:=1 to n do
    A[i,i] := 0;
for k:=1 to n do
    for i:=1 to n do
        for j:=1 to n do
            if A[i,k] + A[k,j] < A[i,j] then
                A[i,j] := A[i,k] + A[k,j];
}
```

Algoritmo 3: Floyd-Warshall

2.7. Resumen del capítulo

Durante este capítulo se presentaron conceptos de gran importancia para el entendimiento de lo que es el enrutamiento compacto, así mismo se describieron las particularidades del enrutamiento compacto al igual que algunos ejemplos También se describieron algunos conceptos empleados en la teoría de grafos y algunos algoritmos utilizados sobre los mismos que ayudaron a la realización de la presente tesis. En el capitulo siguiente se presentarán algunos trabajos de gran importancia y que sirvieron como base para el desarrollo de este trabajo de tesis.

Capítulo 3

Trabajo relacionados

En este capítulo se abordarán dos de los primeros trabajos en utilizar un enrutamiento basado etiquetas que a su vez implementan la orientación a contenido: Tapestry[41] [40] y Chord[36]. También se hablará de dos trabajos más nuevos que son PROSE y MAR[11], en los cuales se emplea un enrutamiento basado en etiquetas que utilizan prefijos y por su parte MAR lo realiza con etiquetas múltiples, ambos trabajos son para redes móviles.Todas las imágenes presentadas en la siguiente sección provienen de los documentos mencionados anteriormente.

3.1. Tapestry

Tapestry es una infraestructura *peer-to-peer* sobrepuesta sobre redes para la localización y ruteo sobre las mismas. Tapestry [41] [40] fue desatollada en la Universidad de Berkeley en el 2001. Su primera aparición fue en forma de reporte técnico y ya después como un artículo en *IEEE Journal on Selected Areas in Communications*. Tapestry ofrece una arquitectura de ruteo escalable, robusta y auto organizada que realiza consultas a contenido eficientemente aun en la presencia de una gran carga dentro de la red y fallas en los nodos.

Tapestry forma parte de la segunda generación de los sistemas sobrepuestos de redes P2P, ésto quiere decir que implementa una interfaz de ruteo basado en *keys* (o claves) que es capaz de soportar enrutamiento determinista de mensajes a un nodo activo que contendrá el identificador del nodo destino.

Antes de explicar el funcionamiento del protocolo de ruteo o búsqueda es necesario mencionar los elementos participantes.

3.1.1. Elementos de Tapestry

Nodos

Los nodos Tapestry serán los nodos participantes en la red sobrepuesta (*overlay*), estos nodos están implantados en nodos reales de la red original. Un nodo físico o real puede contener a más de un nodo Tapestry. Estos nodos reciben un identificador que es asignado uniformemente de manera aleatoria de un gran conjunto de identificadores. Los identificadores son distribuidos uniformemente gracias al algoritmo de seguridad SHA-1. La longitud de los identificadores es de 160 bits en base hexadecimal. El identificador para cada nodo será llamado *nodeIDs*.

Objetos

Los objetos en Tapestry pueden ser información, algún recurso que se desea compartir o hasta una misma replica de otro objeto. Todos los objetos reciben un *GUID* (*Globally Unique Identifier*, por sus siglas en inglés) que también es asignado del mismo espacio de identificadores que el de los nodos. Entonces decimos que el nodo N tiene un *nodeID* N_{id} y un objeto O tiene un *GUID* O_G y para permitir que diversas aplicaciones coexistan en la red Tapestry, cada mensaje contiene un identificador de aplicación A_{id} que es utilizado para seleccionar la aplicación o el proceso que recibirá el mensaje.

DORL

Tapestry, al ser parte de la segunda generación de los sistemas P2P, es capaz de soportar interfaces de alto nivel, en su caso utiliza *DORL*(*Decentralized Object Location and Routing*) que se enfoca en el enrutamiento de mensajes a terminales tales como nodos o réplicas de objetos identificados por sus alias (identificadores asignados del espacio de nombres) en los cuales no se codifica nada acerca de su posición física, proveiendo así una plataforma para implementar aplicaciones distribuidas dentro de la red.

3.1.2. Ubicación, Publicación y Enrutamiento

Ubicación de objetos

Tapestry mapea dinámicamente cada identificador G a un único nodo activo llamado G_R o el identificador de la raíz, por lo tanto la raíz será la encargada de almacenar la ubicación de los objetos. Si existe un nodo N con un $N_{id} = G$, entonces este nodo es el nodo raíz de G. Como consecuencia de lo anterior se puede observar que para la búsqueda y la publicación de objetos dentro de la red, el nodo raíz juega un papel primordial. Este esquema es similar al esquema de localización de Plaxton [29], donde cada nodo que enruta a la raíz almacena la ubicación de las réplicas más cercanas. Ésto permite a Tapestry ser más flexible en comparación con Plaxton. En la Figura 3.1 se muestra cómo los nodos al buscar un objeto, van hacia la copia del objeto más cercana a ellos.

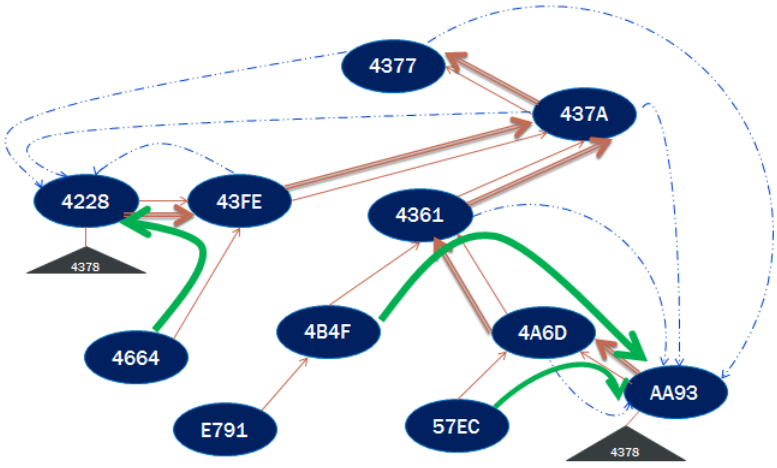


Figura 3.1: Búsqueda de objetos. 4378 es el objeto replicado en la red

Publicación de Objetos

El proceso de publicación de los objetos consiste en enviar un mensaje al nodo raíz, por cada salto en la ruta. El mensaje publicado almacena información de la localización en forma de mapa. Esto es $< Object - ID(O), Server - ID(S) >$, los cuales son simples apuntadores al servidor S donde O está almacenado y no una copia del objeto O. Si existen múltiples objetos, sólo la referencia al objeto más cercano es almacenada.

Enrutamiento

Durante el proceso de búsqueda, los clientes envían mensajes a los objetos. Un mensaje que tiene como destinatario al objeto O, es inicialmente enviado a la raíz del nodo O, inmediatamente es dirigido al servidor que contiene al objeto. Si esto no sucede, el mensaje es reenviado un salto más cerca a la raíz, si el mensaje llega a la raíz se garantiza que encontrará una ruta hacia la ubicación de O.

Para enrutar los mensajes a los destinos, los nodos cuentan con tablas locales llamadas *neighbor maps* o tabla de vecinos (mapa de vecinos), donde se almacenan los *ID´s* junto con las direcciones IP de los nodos con los que tiene un enlace directo, es decir, los vecinos del nodo. Dicha tabla contiene varios niveles, donde cada nivel contiene enlaces a nodos cuyo *ID* coincide hasta cierto dígito en el ID y además la tabla contiene un número de entradas igual a la base del *ID*. La $i - sima$ entrada en el $j - simo$ nivel es el *ID* y la localización del nodo más cercano que empieza con un prefijo $(N, j-1)+i$. Como ejemplo tenemos que la novena entrada del cuarto nivel de un nodo, por ejemplo del nodo 325AE es el nodo más cercano con un *ID* que empieza con 3259.

En la Figura 3.2 se muestra la malla de ruteo del nodo 4227 y se indica el nivel de acuerdo a la coincidencia de los dígitos de los *ID´s* de los nodos vecinos. Después en la Tabla 3.1 se muestra la manera en la que la tabla de vecinos del nodo 4227 quedaría construida de acuerdo con la malla de ruteo mostrada en la figura anterior, en la tabla se puede apreciar la manera en que las entradas están dispuesta de acuerdo con el nivel de coincidencia de los *ID´s*.

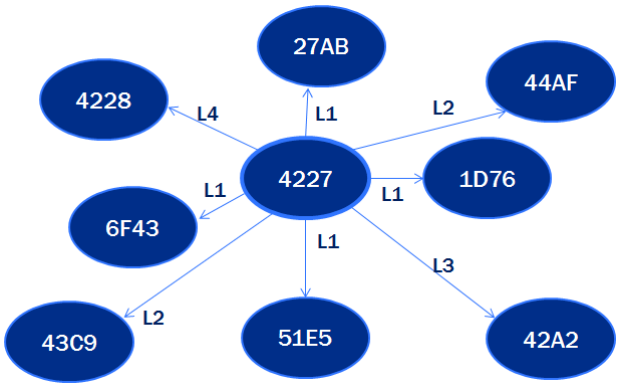


Figura 3.2: Malla de ruteo del nodo 4227 en la cual se muestran los niveles de la tabla de vecinos.

Nivel	1	2	3	4	5	6	7	8	A
1	1D76	27AB			51E5	6F43			
2			43C9	44AF					
3									42A2
4								4228	

Tabla 3.1: Tabla de vecinos para el nodo 4227.

El enrutamiento es basado en sufijos. Este enrutamiento es similar al de los árboles de Plaxton [29]. Las principales características presentes en el enrutamiento por sufijos en Tapestry son:

- Inserción dinámica de nodos.
- Mapeo dinámico del nodo raíz.
- Redundancia en la localización y enrutamiento.
- Protocolos con tolerancia a fallas.
- Adaptativo y auto configurable par la red dinámica.
- Soporta objetos móviles.

Cada nodo en una red Tapestry es capaz de enviar mensajes a los demás nodos dentro de la red con ayuda de la tabla de vecinos. Cada tabla de vecinos está organizada en niveles de ruteo y cada nivel contiene entradas que apuntan a un conjunto de nodos cercanos en la red que coinciden con el sufijo de cada nivel. Cada nodo también mantiene una lista de *backpointers*, que incluye a los nodos que consideran al nodo actual como vecino. En la Figura 3.3 se puede ver la representación de un nodo Tapestry con todos sus componentes, éstos ayudan al correcto funcionamiento de los algoritmos y protocolos dentro de Tapestry.

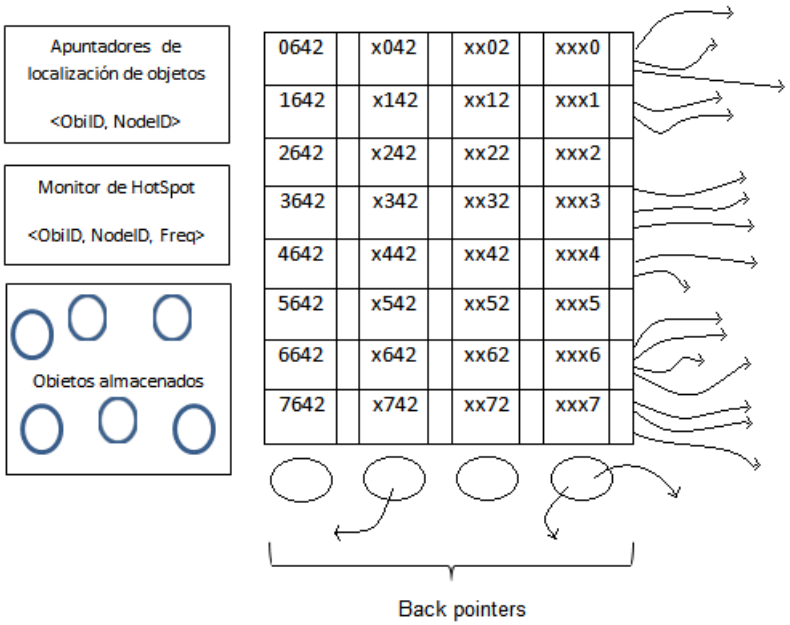


Figura 3.3: Componentes de un nodo Tapestry.

En la Figura 3.4 se presenta en color rojo la ruta que tomó un mensaje a través de la infraestructura Tapestry. El camino para el $n-simo$ salto comparte un prefijo de longitud $\geq n$ con el *ID* de destino. Para enrutar el mensaje, Tapestry revisa la tabla de vecinos en su nivel $(n+1)$ para saber el siguiente dígito en el *ID* de destino. Este método garantiza que cualquier nodo en el sistema pueda ser alcanzado a lo más en $log_{\beta}N$ saltos lógicos en un sistema con un espacio de nombres de tamaño N, *ID´s* de base β y asumiendo que las tablas de vecinos son consistentes.

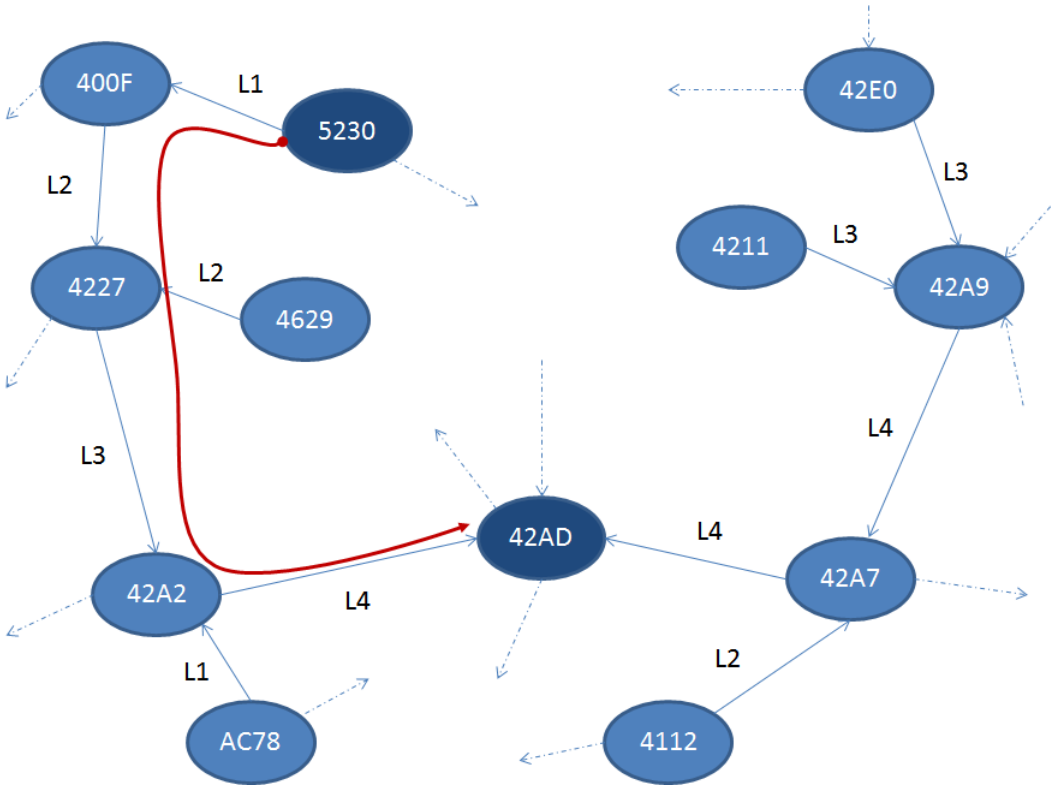


Figura 3.4: Camino de un mensaje enviado desde el nodo 5230 para el nodo 42AD.

3.1.3. Enrutamiento al sustituto (*Surrogate routing*)

En el enrutamiento al sustituto se selecciona tentativamente al nodo raíz de un objeto que tenga el mismo nombre que el *ID* del objeto. Debido a la naturaleza del espacio de nombres, es posible que ese nodo en realidad no exista, sin embargo, Tapestry funciona como si el nodo existiera e intenta enrutar mensajes a él. Una ruta a un identificador no existente encontrará entradas vacías en las tablas de vecinos en varias partes del camino. En estos casos, el objetivo es encontrar un enlace que actúe como una alternativa del enlace que se busca. Esta selección es hecha con una selección determinista entre los apuntadores a vecinos. El enrutamiento termina cuando en una tabla de vecinos la única entrada no vacía apunta hacía el nodo actual. Ese nodo entonces será designado como el sustituto de la raíz del objeto. Esta técnica permite mapear cualquier identificador a un nodo existente dentro de la red.

3.1.4. Tolerancia a fallas

Enrutamiento tolerante a fallas

Para detectar fallas tanto en los enlaces como en los servidores durante las operaciones normales, Tapestry se basa en los tiempos de espera de TCP, además, cada nodo emplea los *backpointers* para enviar señales a los nodos que lo tienen como vecino, esto lo hace por medio de paquetes UDP. Estos son pequeños mensajes "hola"que aseguran que la fuente sigue siendo un vecino elegible para enrutar paquetes. Revisando el *ID* a cada nodo al que llega un mensaje se puede detectar rápidamente un fallo o tablas de vecinos erróneas.

Para operaciones durante fallas, cada entrada en la tabla de vecinos mantiene a dos nodos como respaldo, además de tener al nodo más cercano como vecino. Cuando el primer vecino falla se utiliza a los otros dos vecinos alternativos en orden.

Si un nodo detecta que un vecino es inalcanzable, en lugar de remover su puntero, el nodo marca al vecino como inválido y crea una ruta alternativa. Dado que la mayoría de las fallas en los nodos y en los enlaces son detectadas y reparadas en un tiempo relativamente corto, se mantiene un periodo de segunda oportunidad en el cual un conjunto de mensajes son enrutados al servidor que ha fallado. Esos mensajes que han fallado son reenviados a la ruta alternativa después que ha pasado el tiempo de espera. Un mensaje exitoso indica que la falla ha sido reparada y el apuntador original a la ruta es marcado como válido nuevamente.

Ubicación tolerante a fallas

Para prevenir problemas en la ubicación de los objetos se implementan múltiples nodos raíces, esto es que se concatena una pequeña y constante serie de valores *salt* a cada *ObjectID*, después al resultado se le aplica una función *hash* para identificar a las raíces apropiadas. Estas raíces son utilizadas durante el proceso de publicación (por el enrutamiento al sustituto) para insertar la información de la ubicación en Tapestry. Cuando se localiza a un objeto, Tapestry realiza el mismo proceso de *hash* con el destino de *ObjectID*, generando un conjunto de raíces para realizar la búsqueda.

3.1.5. Algoritmos de nodo dinámico

Tapestry incluye un número de mecanismos para mantener las tablas de ruteo consistentes y asegurar la disponibilidad de los objetos. Estos mecanismos se explicarán brevemente a continuación.

Inserción de nodos

Hay cuatro componentes presentes en la inserción de un nodo N en Tapestry:

- a) Necesidad de saber quién se inserta. Los nodos son notificados de la inserción de N porque éste llena una entrada vacía en sus tablas de enrutamiento.
- b) N puede convertirse en la nueva raíz de objetos existentes. Las referencias a esos objetos deben ser cambiadas a N para mantener la disponibilidad de los objetos.
- c) Los algoritmos deben construir una tabla de enrutamiento óptima para N.
- d) Los nodos cercanos a N son notificados y pueden considerar a N en sus tablas de enrutamiento como una optimización.

La inserción del nodo comienza en el sustituto de N, el nodo S. S encuentra p, la longitud máxima del prefijo que su ID comparte con N_id. S envía un mensaje de reconocimiento multicast que llega a todos los nodos existentes que comparten el mismo prefijo recorriendo un árbol basado en sus *NodeIDs*. Cuando los nodos reciben el mensaje, añaden a N a sus tablas de enrutamiento y transfieren referencias de apuntadores locales si es necesario, de esta manera se completan los incisos a) y b).

Los nodos alcanzados por el mensaje multicast, se convierten en el primer conjunto de vecinos utilizados en la construcción de la tabla de enrutamiento de N. Después N realiza una búsqueda iterativa de vecinos más cercanos empezando con el nivel de enrutamiento p. N utiliza el conjunto de vecinos para llenar el nivel p de enrutamiento, recorta la lista a únicamente los k nodos más cercanos y les solicita que le envíen sus *backpointers* en ese nivel. El grupo resultante contiene a todos los nodos que apuntan a cualquiera de los k nodos en el nivel de enrutamiento previo y se convierten en el siguiente conjunto de vecinos. N decrementa p y repite el proceso hasta que que todos los niveles son llenados. Este proceso completa al inciso c) y todos los nodos contactados durante el proceso iterativo utilizarán a N para optimizar sus tablas de enrutamiento cuando sea necesario y así completar el inciso d). En la Figura 3.5 se ejemplifica la inserción de un nodo. Se pueden observar los cuatro componentes de la inserción mencionados anteriormente.

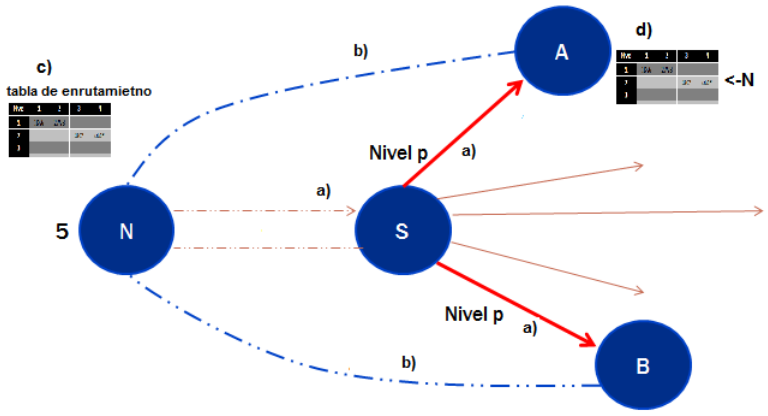


Figura 3.5: Proceso de inserción de un nodo.

Partida voluntaria de nodos Cuando el nodo N nodo tiene intención de abandonar Tapestry se encarga de anunciar a los nodos que hacen o tienen referencia a él, de su intención mediante un mensaje y en ese mensaje también envía su remplazo para cada nivel. En caso de ser raíz de algún objeto también redirige las referencias a sus remplazos. Cada uno de los nodos notificados vuelven a publicar el tráfico para informar de la partida de N y del remplazo del mismo.

En la Figura 3.6 el nodo B está a punto de abandonar la red Tapestry, por lo tanto informa al servidor S. El servidor se encarga de volver a publicar el mensaje para redireccionar los apuntadores.

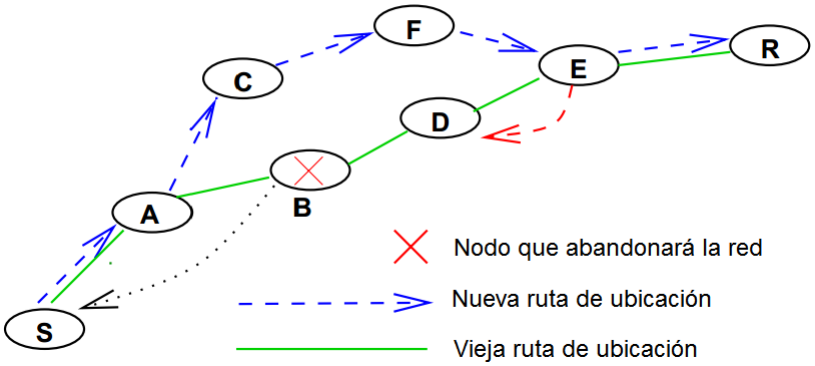


Figura 3.6: Actualización de los apuntadores para nodos salientes.

Partida involuntaria de nodos En una red dinámica, los problemas ocurren debido a fallas en los enlaces o en las particiones de la red, además de la movilidad de los nodos ya que pueden entrar y abandonar la red en periodos de tiempo muy cortos. Tapestry mejora la disponibilidad de los objetos y enrutamiento en tales ambientes construyendo tablas de enrutamiento y referencias a la ubicación de objetos redundantes. Para mantener la redundancia y la disponibilidad, los nodos Tapestry utilizan periódicamente un faro para detectar enlaces nuevos o fallas en los existentes, estos eventos disparan la reparación de la malla y también una redistribución de las referencias a los objetos.

3.2. Chord

Chord [36], fue desarrollado en el 2002 como un protocolo de búsqueda distribuida el cual propone un mecanismo para localizar de forma eficiente un nodo que almacena a un conjunto de datos particulares. Chord proporciona soporte para una sola operación: dada una clave (o llave), Chord la mapea a un nodo. La ubicación de la información puede ser implementada fácilmente en Chord asociando una clave con cada objeto de información y almacenando el conjunto de la clave con el objeto de información en el nodo al que se le asigna la clave. Chord adapta el mecanismo en el que los nodos pueden entrar y salir del sistema y es capaz de responder a las consultas de información aun en ese estado de constante cambio. Sus características principales son su simplicidad y que se puede caracterizar formalmente su desempeño, así como demostrar que el algoritmo es correcto.

3.2.1. Modelo del sistema

Chord simplifica el diseño de sistemas *peer to peer* (sistemas entre pares) y de las aplicaciones basadas en los sistemas P2P tomando en cuenta los siguientes problemas:

- **Balanceo de carga.** Chord actúa como una función hash distribuida repartiendo las claves a los nodos de manera uniforme. Esto brinda un nivel de balanceo de carga natural dentro del sistema.
- **Descentralización.** Chord es completamente distribuido, ningún nodo es más importante que otro. Esto mejora la robustez de Chord y lo hace apropiado para las aplicaciones p2p débilmente organizadas.
- **Escalabilidad.** El costo de una búsqueda Chord crece como el logaritmo del número de nodos en la red, incluso en sistemas muy grandes las búsquedas siguen siendo factibles. No se requieren ajustes en los parámetros para lograr la escalabilidad.
- **Disponibilidad.** Chord ajusta automáticamente sus tablas internas para reflejar la unión de nuevos nodos, así como los fallos en los mismos. Esto garantiza que salvo grandes fallas en la red subyacente, el nodo responsable de una clave siempre podrá ser encontrado. Esto es cierto aun si el sistema está en un estado de cambio continuo.
- **Nombramiento flexible.** Chord no restringe la estructura de las claves. El espacio de las claves es plano, ésto proporciona a las aplicaciones una gran flexibilidad en la forma en que éstas mapean sus propios nombres a las claves de Chord.

El software de Chord toma la forma de una biblioteca para enlazarse con las aplicaciones de cliente y de servidor que lo utilizan. La aplicación interactúa con Chord de dos maneras. En la primera, Chord provee un algoritmo de búsqueda el cual ofrece la dirección IP del nodo responsable de la clave. En la segunda, el software de Chord en cada nodo notifica a la aplicación de cambios en el conjunto de claves de las que el nodo es responsable. Lo anterior permite que el software de aplicación pueda mover los valores correspondientes a sus nuevos destinos cuando un nuevo nodo se une a la red Chord. La aplicación que utiliza a Chord es la responsable de proporcionar cualquier tipo de autenticación deseada, almacenamiento en caché, replicación y la facilidad de nombrar a los objetos.

3.2.2. Protocolo Chord

Chord ofrece el cómputo distribuido de una función hash que mapea claves a nodos responsables de almacenar valores. Chord utiliza una función hash consistente que posee varias propiedades deseadas. Con una alta probabilidad, la función hash balancea la carga (todos los nodos recibirán casi la misma cantidad de claves). Cuando el N-ésimo nodo llega a la red o la deja, sólo $O(1/N)$ de las claves serán cambiadas a una diferente ubicación, ésto para mantener el balanceo de la carga en los nodos.

Un nodo Chord necesita sólo una pequeña cantidad de información de ruteo acerca de los demás nodos, ya que la información está distribuida. Un nodo resuelve la función hash comunicándose con pocos nodos. En una red con N-nodos, cada nodo mantiene información acerca de $O(\log N)$ nodos y una búsqueda requiere $O(\log N)$ mensajes para realizarse.

Función hash consistente
La función hash consistente [21][16] asigna a cada nodo y a cada clave un identificador de m-bits utilizando una función hash base como $SHA-1$. El identificador del nodo se obtiene mediante la función hash a su dirección IP y el identificador de la clave se obtiene con el resultado de la hash a la misma clave. El identificador de longitud m debe de ser lo suficientemente grande para evitar que dos nodos o claves tengan el mismo identificador.

La función hash consistente asigna las claves a los nodos de la siguiente manera. Los identificadores son ordenados en un anillo de identificadores módulo 2^m. La clave k se le asigna al primer nodo cuyo identificador es igual o sucede al identificador k en el espacio de identificadores. Este nodo es llamado el nodo sucesor de la clave k, denotado por *sucessor(k)*. En la Figura 3.7 se muestra un anillo de identificadores con una $m = 6$. El anillo cuenta con 10 nodos y almacena 5 claves. El sucesor del identificador 10 es el nodo 14, por lo tanto la clave 10 estará ubicada en el nodo 14 y de manera similar las claves 24 y 30 estarán ubicadas en el nodo 32, la clave 38 en el nodo 38 y la clave 54 en el nodo 56.

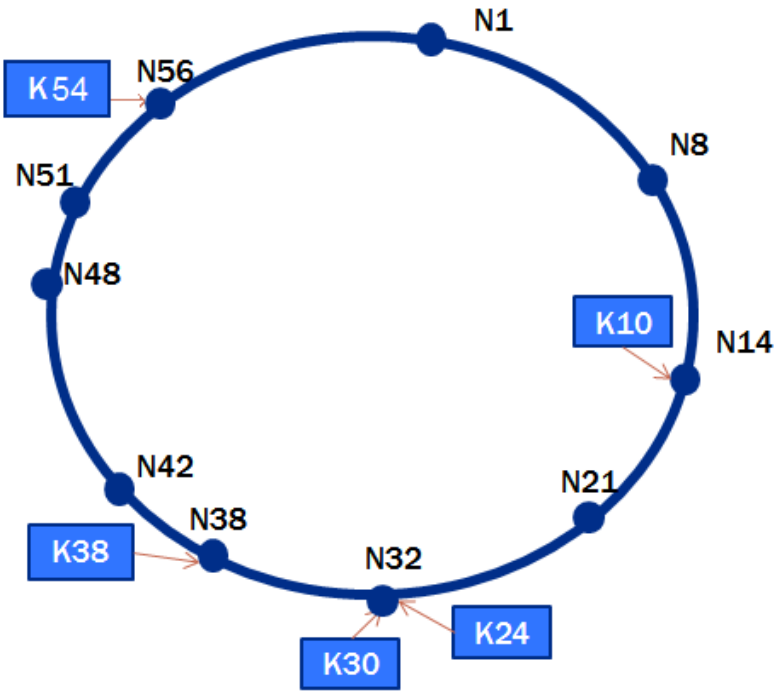


Figura 3.7: Anillo de identificadores con 10 nodos almacenando 5 claves.

La función *hash* consistente está diseñada para permitir a los nodos entrar y salir de la red con un mínimo de ruptura en las tablas de enrutamiento. Para mantener el mapeo de la función consistente cuando entra un nodo n se le asignan ciertas claves de los que tenía asignado su sucesor y al salir el nodo se le asignan de nuevo a su sucesor.

El siguiente teorema está demostrado en los artículos [21][16] en los cuales se presentó a la función *hash* consistente.

Teorema 3.1 *Para cada conjunto N de nodos y K claves, con una alta probabilidad:*

- *1. Cada nodo es responsable de a lo mas $(1+\epsilon)K/N$ claves.*
- *2. Cuando un $(N+1)$-esimo nodo se une o abandona la red, la responsabilidad de $O(K/N)$ claves cambia. Únicamente hacia el nodo entrante o desde el nodo saliente.*

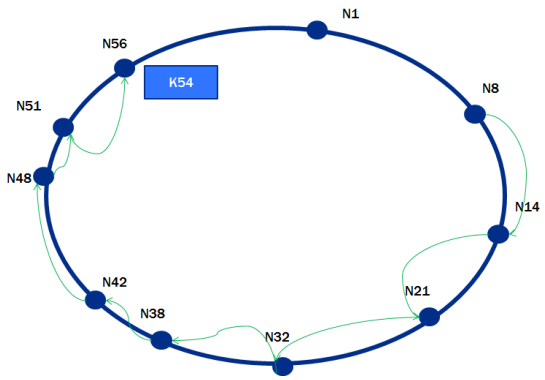
Cuando una función *hash* consistente es implementada, el teorema se demuestra con una cota de $\epsilon = O(\log N)$

Ubicación simple de la clave

Las búsquedas de una clave pueden ser implementadas en un anillo lógico de Chord con poca información del estado de los nodos. Cada nodo sólo necesita saber cómo contactar a su nodo sucesor actual en el anillo de identificadores. Las búsquedas para un identificador dado pueden ser pasadas en el anillo Chord a través de los apuntadores a los sucesores de los nodos hasta que encuentren un par de nodos que coincidan con el identificador deseado; el segundo elemento del par es el nodo al que mapea la búsqueda.

```
// pregunta al nodo n para encontrar al sucesor del id
n.find_sucessor(id)
        if (id∈ (n, n.sucessor)
                return n.sucessor;
        else
                //enviar la consulta al rededor del anillo
                return sucesor.find_sucessor(id);
```

(a) Psuedocódigo para encontrar al nodo sucesor de un identificador *id*



(b) Camino que sigue la búsqueda de la clave 54 hecha por el nodo 8 siguiendo el algoritmo en (a).

Figura 3.8: Pseudocódigo y camino que realiza una búsqueda dentro del anillo Chord.

En la Figura 3.8 (a) se muestra el pseudocódigo implementado en el proceso de búsqueda del nodo sucesor del identificador *id*, las llamadas a procedimientos remotos y las búsquedas de las variables son precedidas por el nodo remoto. En la parte (b) se muestra un ejemplo en el que el nodo 8 realiza una búsqueda de la clave 54. El nodo 8 invoca el método *find_ sucessor* para la clave 54 el cual regresará eventualmente el nodo sucesor de esa clave, el nodo 56. Esta consulta visita a todos los nodos en el círculo entre el nodo 8 y el 56 y el resultado viajará sobre el camino de la consulta pero en dirección contraria.

Ubicación escalable de la clave

El esquema de búsqueda simple anteriormente presentado utiliza un número lineal de mensajes de acuerdo con el número de nodos. Para acelerar las búsquedas, Chord mantiene información de enrutamiento adicional.

De acuerdo con lo anterior tenemos. Sea m el número de bits en la clave y en los identificadores de los nodos. Cada nodo n mantiene una tabla de enrutamiento con (a lo más) m entradas. Estas tablas son llamadas *finger tables*. La i-ésima entrada en la tabla del nodo n contiene la identidad del primer nodo s, que sucede por lo menos en 2^{i-1} en el círculo de identificadores a n ($s = sucesor(n + 2^{i-1})$, donde $1 \leq i \leq m$. El nodo s es llamado el i-ésimo *finger* del nodo n y es denotado como *n.fingeri*. Una entrada en la *finger table* incluye tanto el identificador Chord como la dirección IP (el número de puerto) del nodo relevante. El primer *finger* de n es el sucesor inmediato de n en el anillo y por conveniencia siempre se referirán a éste como el sucesor. En la Tabla 3.2 se muestran las definiciones de las notaciones utilizadas en los algoritmos de búsqueda dentro de los nodos Chord.

Notación	Definición
finger[k]	Primer nodo en el anillo que sucede (n+2k-1) mod 2m, 1≤k≤m
sucessor	El siguiente nodo en el anillo de identificadores; finger[1].node
predecessor	El nodo previo en el anillo de identificadores.

Tabla 3.2: Definición de las variables para el nodo n utilizando identificadores de $m-$bits.

```
//Pregunta al nodo n para encontrar al sucesor del id
n.find_sucessor(id)
        if(key ∈ (n.nsucesor)
                return n.sucessor;
        else
                n' = closest_preceding_node(id);
                return n'.find_sucessor(id);
```

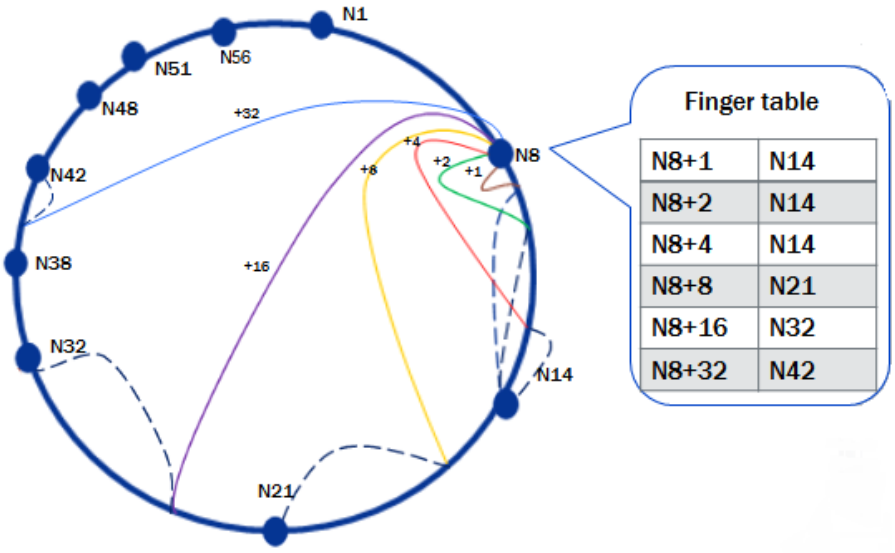
```
// Búsqueda local en la tabla del predecesor del id
n.closest_preceding_node(id)
        for i= m down to 1
                if(finger[i] ∈ (n,id))
                        return finger[i];
                return n;
```

Figura 3.9: Consulta escalable de claves utilizando *finger tables*.

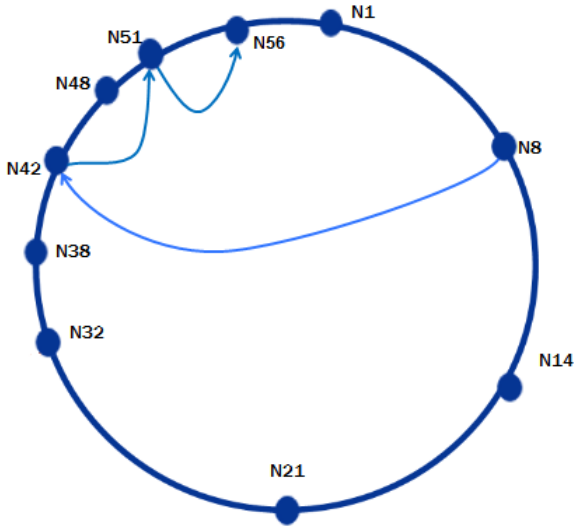
En la Figura 3.9 se muestra el pseudocódigo de la operación *find_ sucessor* modificada para utilizar las *finger tables*. Si el *id* se encuentra entre n y el sucesor de n' , *find_ sucessor* termina y n regresa su sucesor. De otra manera n busca en su *finger table* al nodo n' cuyo *ID* preceda inmediatamente a *id* y después invoca a *find_ sucessor* en n'. Como ejemplo podemos tomar el anillo Chord en la Figura 3.10 (b) y suponer que el nodo 8 quiere encontrar al sucesor de la clave 54. Ya que el *finger* más grande que tiene el nodo 8 en su tabla y que precede a 54 es el nodo 42, el nodo 8 le solicitará al nodo 42 que resuelva la consulta y éste a su vez determinará cuál es el *finger* más grande que preceda a 54 en su tabla. Este es el nodo 51, finalmente el nodo 51 determinará que su propio sucesor, el nodo 56 sucede a la clave 54, por lo tanto el nodo 51 enviará como respuesta el nodo 56 al nodo que inició la consulta, en este caso al nodo 8.

Este esquema cuenta con dos características importantes.

- Cada nodo almacena información únicamente acerca de una pequeña cantidad de nodos y sabe más acerca de los nodos cercanos a él en el anillo de identificadores que de los lejanos a él en el anillo.
- La *finger table* de un nodo, generalmente no contiene la información necesaria para determinar el sucesor de una clave k determinada. Por ejemplo, en la Figura 3.10 (a), el nodo 8 no puede determinar el sucesor de la clave 34 por sí solo, debido a que el nodo sucesor de la clave 34 no aparece en la *finger table* del nodo 8.



(a) Entradas de la *finger table* del nodo 8



(b) Camino que sigue la búsqueda de la clave 54 hecha por el nodo 8 utilizando el algoritmo de la Figura 3.9

Figura 3.10: Ubicación escalable de una clave.

En la Figura 3.10 se puede observar la *finger table* del nodo 8. El primer *finger* del nodo 8 apunta al nodo 14, el nodo 14 es el primer nodo que sucede debido a que $(8 + 2^5) mod 2^6 = 9$. De manera similar, el último *finger* del nodo 8 apunta al nodo 42, ya que el nodo 42 es el primer nodo que sucede a $(8 + 2^0) mod 2^6 = 40$.

Dado que cada nodo tiene entradas en la *finger table* en intervalos de potencias de dos en el anillo de identificadores, cada nodo puede reenviar las consultas al menos a la mitad del camino entre el nodo y el identificador destino. Partiendo de esto se tiene el siguiente teorema.

Teorema 3.2 *Con una alta probabilidad (o bajo las aseveraciones de dureza estándar) el número de nodos que deberán ser contactados para encontrar un sucesor en una red de* $N - nodos$ *es* $O(logN)$.

3.2.3. Operaciones dinámicas y fallas

Chord necesita lidiar con nodos añadiéndose al sistema y con nodos que fallan o que abandonan la red de manera voluntaria. A continuación se describirá cómo es que Chord maneja estas situaciones.

Unión de nodos y estabilización
Para asegurar que las búsquedas se ejecuten de manera correcta, cuando el conjunto de nodos participantes cambia, Chord debe asegurarse que cada apuntador a sucesor está actualizado. Esto lo realiza con la utilización de un protocolo de estabilización básico. Chord verifica y actualiza las entradas de las *finger tables* utilizando una combinación de *fingers* existentes (y posiblemente desactualizados) así como apuntadores a sucesores correctos.

Sí al unirse nodos se afectó una región del anillo Chord, una búsqueda que ocurre antes de que el protocolo de estabilización haya terminado puede presentar uno de los siguientes tres comportamientos:

- Primer caso. Todas las entradas en las *finger tables* involucradas en la búsqueda estén razonablemente actualizadas y la búsqueda encuentre al sucesor correcto en $O(\log N)$ pasos.
- Segundo caso. Los apuntadores a sucesores son correctos pero las entradas en las *finger tables* son incorrectas. Esto realizará búsquedas correctas pero puede que sean más lentas.
- Tercer caso. Los nodos en la región afectada del anillo Chord tienen apuntadores a sucesores incorrectos, o las claves puede que aún no hayan migrado a los nodos que recién se añadieron a la red y la búsqueda posiblemente fallará. La capa más alta utilizando Chord notará que la información deseada no fue encontrada y tendrá la opción de realizar nuevamente la búsqueda después de una pausa. Esta pausa puede ser corta debido a que el protocolo de estabilización repara los apuntadores a sucesores de manera rápida.

Este esquema de estabilización garantiza que al unirse nuevos nodos al anillo Chord, se mantendrá la accesibilidad a los nodos existentes aun si se enfrenta a uniones y fallas concurrentes. En la Figura 3.11 se muestra el pseudocódigo para las uniones y la estabilización. Cuando un nodo n se une, llama a $n.join(n')$, donde n' es cualquier nodo Chord conocido. La función $join()$ le solicita a n' que encuentre el sucesor inmediato de n. La función $join()$ por sí sola no hace que la red tenga conocimiento de n.

```
// Construcción de la finger table de n'
n.build_fingers(n')
        i0 := ⌊ log (sucessor-n)⌋+1; // primer finger no trivial
        for each  i≥ i0 index into finger[];
                finger[i]=n'.find_sucessor (n+2^(i-1));

        n.join(n')
                predecessor=nil;
                s=n'.find_sucessor(n);
                build_fingers(s):
                successor=s;
```

```
// Verificación periódica de los sucesores inmediatos de
// n'  y se le informa al sucesor acerca de n
n.stabilize()
        x=sucessor.predecessor;
        if(x ∈ (n, sucessor))
                sucessor=x;
        sucessor.notify(n);

//n' cree que puede ser un predecesor
n.notify(n')
        if(predecessor is nil or n' ∈ (predecessor))
                Predecessor=n';
```

Figura 3.11: Pseudocódigo del protocolo de estabilización.

Cada nodo ejecuta periódicamente la función *stabilize()* y así es como los nodos del sistema saben de los nodos que recién se han unido. Cuando el nodo n ejecuta *stabilize()*, le pregunta a su sucesor por su predecesor p, y n decide si p debería ser sucesor de n.*stabilize()* también se encarga de informar al sucesor de n de la existencia de n permitiendo que el sucesor cambie sus apuntadores a predecesor hacia n. Como ejemplo, suponemos que el nodo n se une al sistema y su *ID* queda entre los nodos n_p y n_s. Al ejecutarse *join()* en el nodo n éste adquiere n_s como su sucesor. Adicionalmente n copia todas las claves con *ID* mayor o igual al su *ID* desde n_s. Cuando el nodo n_s ejecuta nuevamente la función *stabilize()*, preguntará a n_s por su predecesor (que ahora es n); entonces éste declarará a n como su sucesor. Finalmente n_p notificará a n y n adquirirá a n_p como su predecesor. En este punto todos apuntadores a predecesores y sucesores son correctos.

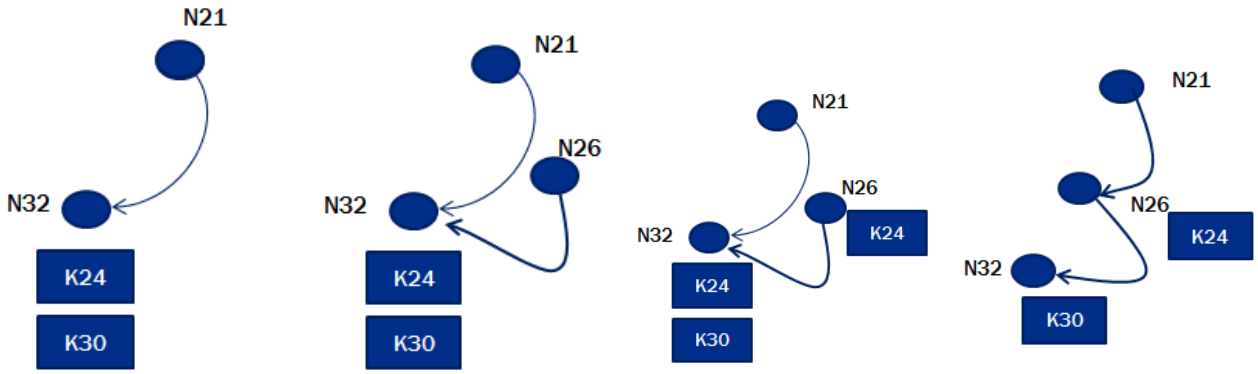


Figura 3.12: Ejemplo de la unión de un nodo. El nodo 26 se une al sistema en medio del nodo 21 y 32.

En la Figura 3.12 se puede apreciar el ejemplo en el que el nodo 26 se une al sistema en medio del nodo 21 y el nodo 32. Los arcos representan las relaciones de sucesores. En la Figura 3.12(a) se muestra el estado inicial del sistema: el nodo 21 apunta al nodo 32 como su sucesor. En la Figura 3.12(b) el nodo 26 encuentra a su sucesor, el nodo 32 y dirige sus apuntadores de sucesor a él. En la Figura 3.12(c) el nodo 26 copia en él todas las claves entre entre 26 y 31 desde el nodo 32. Finalmente, en la Figura 3.12(d) el procedimiento de estabilización actualiza el apuntador a sucesor del nodo 21 al nodo 26.

Tan pronto como los apuntadores a los sucesores son correctos, las invocaciones a la función *find_ sucessor* funcionarán de manera correcta. Los nuevos nodos añadidos a la red que no han sido agregados a las *finger tables* pueden causar una respuesta lenta en la búsqueda pero eventualmente las tablas se ajustarán y se encontrará el predecesor correcto. Dado lo anterior se considera el siguiente teorema.

Teorema 3.3 *Si cualquier secuencia de operaciones de unión es ejecutada en intervalos conjuntamente con el proceso de estabilización, en cierto tiempo después de la última unión, los apuntadores a sucesor formarán un ciclo con todos los nodos en la red.*

En otras palabras, después de cierto tiempo, cualquier nodo será capaz de alcanzar a cualquier otro nodo en la red siguiendo los apuntadores a sucesor.

3.2.4. Impacto de la unión de nodos en el desempeño de las búsquedas

Una vez que el proceso de estabilización se ha completado, los nodos nuevos no causarán otro efecto más que aumentar el valor de N en el tiempo de búsqueda $O(\log N)$. Si la estabilización no se ha completado, las entradas en la *finger tables* de los nodos existentes no reflejarán la adición de los nuevos nodos. Partiendo de esto se llega a siguiente teorema.

Teorema 3.4 *Si tomamos una red estable de N nodos, y otro conjunto de hasta N nodos se une a la red y todos los apuntadores a sucesores son correctos (pero quizá no todos los apuntadores en la finger table), entonces las búsquedas con una alta probabilidad seguirán tomando el tiempo de $O(\log N)$ en realizarse.*

Esto quiere decir que mientras el tiempo que tome en ajustar las entradas en las *finger tables* sea menor que el tiempo que tarde la red en duplicar su tamaño, las búsquedas seguirán siendo en $O(\log N)$ saltos.

3.2.5. Fallas y replicación

La exactitud del protocolo Chord se basa en el hecho de que cada nodo conoce a su sucesor, sin embargo, esta invariante puede ser comprometida si los nodos fallan debido a que un sucesor incorrecto conducirá a consultas incorrectas. Para incrementar la robustez, cada nodo Chord mantiene una lista de sucesores de tamaño r. Si el sucesor inmediato de un nodo no responde, el nodo podrá substituirlo con su sucesor en la segunda entrada de la tabla. Para quebrantar el anillo de Chord, los r sucesores en la tabla tendrían que fallar al mismo tiempo y este suceso es poco probable con tamaños de r pequeños. Una implementación debería utilizar un tamaño fijo de r el cual debe ser $2 \log_2 N$ para un número previsible N de nodos.

Para poder implementar la lista de sucesores en los pseudocódigos de las Figuras 3.9 y 3.11 se requieren de pequeños cambios. En el protocolo de estabilización, la lista de sucesores se estabiliza de la siguiente manera. El nodo u construye su lista con sus s sucesores copiando la lista l de s, agregando a s al frente de l y eliminando al último elemento. Si el nodo n nota que su sucesor ha fallado, éste lo remplaza con la primera entrada en la lista de sucesores y compone su lista de sucesores con el nuevo sucesor. En este punto el nodo n puede hacer búsquedas ordinarias para las claves en donde el nodo que ha fallado era sucesor.

Con respecto a la función *closest_preceding_node*, las búsquedas no son únicamente en las *finger tables*, sino también en la lista de sucesores para el predecesor inmediato del *id*. Asimismo, el pseudocódigo necesita ser mejorado para poder manejar las fallas de los nodos. Si un nodo falla durante el procedimiento *find_sucesor*, la búsqueda se realizará después de un tiempo de espera utilizando la siguiente mejor opción de sucesor entre los nodos en la *finger table* y la lista de sucesores.

Los siguientes teoremas ayudan a demostrar la robustez de Chord mostrando que ni las consultas exitosas o el desempeño de las búsquedas en Chord se ven afectadas aun con la falla masiva de nodos. Ambos teoremas asumen que la lista de sucesores es de tamaño $r = O(\log N)$ y que un anillo Chord es estable si cada apuntador a sucesor en los nodos es correcto.

Teorema 3.5 *Si se utiliza una lista de sucesores de tamaño* $r = O(\log N)$ *en una red que es inicialmente estable y cada nodo falla con una probabilidad de 1/2, entonces find_sucessor regresará el nodo sucesor activo más cercano a la clave.*

Teorema 3.6 *En un red que inicialmente es estable, si cada nodo falla con una probabilidad de* $1/2$*, el tiempo esperado para la ejecución de de find_sucesosor es de* $O(\log N)$.

3.2.6. Salida voluntaria de nodos

Debido a que Chord es robusto en cuanto a fallas, un nodo que abandona voluntariamente el sistema puede ser tratado como un nodo que ha fallado. Sin embargo, existen dos mejoras que pueden ayudar en el desempeño de Chord cuando un nodo abandona voluntariamente el la red y se mencionan a continuación.

- Un nodo n que está a punto de abandonar la red podría transferir sus claves a su sucesor antes de que abandone la red.
- n podría notificar a su predecesor p y a su sucesor s que abandonará la red, esto para que p pueda remover a n de su lista de sucesores y después agregar al último nodo en la lista de sucesores de n a su propia lista. De igual manera, el nodo s remplazará a su predecesor con el predecesor de n. Aquí se asume que n envía su predecesor a s y el último nodo en su lista de sucesores a p.

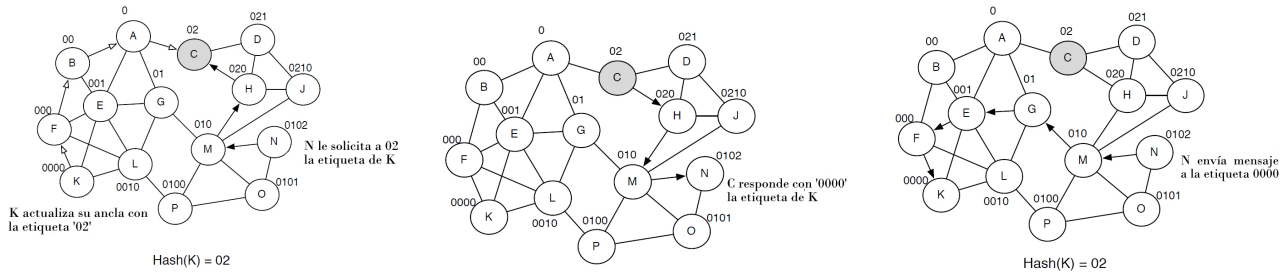
3.3. PROSE

En esta sección se presenta a PROSE[34], por sus siglas en inglés (*Prefix Routing Over Set Elements*). PROSE es un protocolo para el enrutamiento escalable en redes MANETs basado en una combinación del uso de etiquetas que utilizan prefijos y tablas *hash* distribuidas.

3.3.1. Funcionamiento

El protocolo PROSE enruta paquetes de fuentes hacia destinos al asignar etiquetas basadas en prefijos a nodos y realizando el mapeo dinámico de identificadores únicos de los nodo hacia etiquetas basadas en prefijos.

Además, construye y mantiene un grafo etiquetado acíclico dirigido (LDAG, *Labeled Directed Acyclic Graph*, por sis siglas en inglés) con raíz en un nodo elegido de manera distribuida, el cual es llamado nodo raíz. La elección del nodo raíz es similar a la elección de la raíz en el algoritmo de un árbol de expansión distribuido. Se propagan paquetes de señalización de vecino a vecino desde la raíz hacia toda la red de la misma manera como en la búsqueda por amplitud. Esto asegura que cada nodo reciba una etiqueta basada en prefijos del mismo nodo raíz. Un nodo anuncia a sus vecinos su propia etiqueta y las etiquetas basadas en prefijos que asigna a sus hijos en el LDAG y almacena las etiquetas que escucha de sus vecinos. En su publicación los autores de PROSE asumen que un nodo informa a sus vecinos sólo la etiqueta basada en prefijos más pequeña que es asignada por sus parientes. La Figura 3.13 muestra un ejemplo de una MANET en la que el nodo A se elige como nodo raíz.



(a) (b) (c)

Figura 3.13: Funcionamiento de PROSE: El nodo A es la raíz; el nodo C es el nodo ancla del nodo K; el nodo N solicita la etiqueta basada en prefijo del nodo K y después envía los mensajes directamente a él.

Claramente, una fuente debe saber la etiqueta prefijo de su destino para que el enrutamiento basado en prefijos sea efectivo. Sin embargo, almacenar todos lo mapeos de NIDs (*Node Identifiers*, por sus siglas en inglés) hacia etiquetas basadas en prefijos en uno o sólo unos cuantos nodos podría provocar cuellos de botella y puntos de únicos de fallas. En consecuencia, los nodos construyen y mantienen una tabla *hash* distribuida (DHT, por sus siglas en inglés) para almacenar los mapeos a través de toda la red y llevar a cabo las operaciones de suscripción-publicación usando señalización de estado suave, la cual usa etiquetas para enrutar paquetes hacia nodos específicos.

Cada destino usa una función *hash* consistente que toma como entrada su NID, y regresa su etiqueta basada en prefijos pública, la cual establece la localización dónde su mapeo debería ser almacenado. El nodo con la etiqueta basada en prefijos más parecida a la etiqueta se

convierte en el ancla para ese destino. Posteriormente, un destino publica su presencia en la red al enviar su propio mapeo a su ancla (Figura 3.13 (a)). Para encontrar a un destino y suscribirse a él, una fuente usa la misma función *hash* consistente con el NID del destino como entrada. Entonces, la fuente envía una solicitud hacia la etiqueta del ancla resultante. El ancla del destino responde a la fuente con la etiqueta basada en prefijos del destino, o reenvía la solicitud hacia el destino. Entonces, el destino responde directamente a la fuente. Una vez que el destino y la fuente conocen las etiquetas uno del otro ellos se pueden comunicar directamente (ver Figura 3.13 (c)). Como se puede ver en la figura, es claro que PROSE soporta enrutamiento múlticamino.

EL LDAG sirve como una estructura fuerte tolerante a la salida de nodos de la red, la cual por omisión asigna a un nodo la etiqueta más parecida cuando una solicitud para una concordancia falla (cuando el nodo buscado no se encuentra). El algoritmo de enrutamiento que trabaja sobre el LDAG usa una estrategia *greedy*. Cuando encuentra un mínimo local, escoge el siguiente salto con el NID más bajo. Dado que los nodos tienen etiquetas basadas en prefijos, la opción del NID no afecta el camino hacia el destino. Un nodo usa la lógica de similitud máxima de las etiquetas del algoritmo de enrutamiento para escoger su siguiente salto.

3.3.2. Estructura de las etiquetas basadas en prefijos

Sea Σ el alfabeto que contiene un número finito de símbolos y Σ^* el conjunto de todas las cadenas sobre Σ tal que $|\Sigma| \geq 2$. Cada nodo etiqueta a sus enlaces para cada uno de sus vecinos con una letra ω de Σ. Si ω_i representa la letra asignada al i-ésimo enlace de cualquier nodo entonces, $\omega_i \mapsto \omega_i \in \Sigma | \omega_i \neq \omega_{i+1} \forall_i \leq d-1$, donde d es el grado del nodo. Por lo tanto, se asigna una única letra a cada enlace que conecta a algún nodo con sus vecinos. La lógica etiquetadora etiqueta a cada nodo en el LDAG de manera parecida a la búsqueda por amplitud. Dado que el LDAG se puede organizar como un árbol $k-ario$, siendo k el grado del LDAG, a cada hijo (hasta k) se le asigna una etiqueta basad en prefijos Λ como se define a continuación:

Etiqueta basad en prefijos: Una etiqueta basada en prefijos Λ para el nodo y es una palabra dentro de Σ^* tal que $\Lambda = \Lambda parent \odot l$, donde $\Lambda parent$ es la etiqueta prefijo obtenida del padre y $\odot$ es el operador de concatenación donde $\Lambda parent$ es concatenada con un único sufijo sobre k diferentes opciones de Σ para formar Λ.

Se deduce de la definición anterior que la etiqueta Λ de un nodo identifica únicamente al nodo en un LDAG dado. Las etiquetas de los nodos definen una relación predecesor en el LDAG, denotada por $\hookleftarrow$, tal que para dos nodos cualesquiera s y d en el LDAG:

1) $s \hookleftarrow d$: s *precede* a d. En este caso se puede alcanzar a d al recorrer los subárboles de s.

2) $d \hookleftarrow s$: d *precede* a s . En este caso se puede alcanzar a d yendo hacia arriba del flujo de s al recorrer los antecesores de s.

3) $d \approx s$: d y s son pares donde $\Lambda_d = \Lambda_s$ y ambos comparten un antecesor común. En este caso, d puede ser alcanzado al recorrer hacia arriba al antecesor común hasta que se mantenga $r \hookleftarrow s$, donde r es el antecesor común, y entonces recorrer hacia abajo del subárbol más parecido, mientras $r \hookleftarrow d$ se mantenga y d sea alcanzado. Por lo tanto, $d \approx s \Rightarrow r \hookleftarrow s \Rightarrow r \hookleftarrow d$.

4) $dr \approx s$: Este es un caso especial del anterior, donde $|\Lambda_d| \neq |\Lambda_s|$ y el único antecesor

común es el nodo raíz.

Los nodos construyen y mantienen un LDAG con raíz en un nodo elegido usando mensajes hola intercambiados entre los vecinos del *1-vecindario*. Cada mensaje hola específica el NID de la raíz, un número secuencial que se incrementa monotónicamente el cual es asignado por el nodo raíz, la etiqueta basada en prefijos, el NID del nodo emisor, y una lista de tuplas con la asignación de etiquetas hacia NIDs. El algoritmo para la elección de la raíz escoge el nodo con el cubrimiento más grande hecho a un salto y rompe cualquier empate con el NID más chico. En cuanto la red inicia a autoorganizarse con etiquetas basadas en prefijos, el proceso de etiquetamiento puede producir múltiples LDAGs, la etiqueta de cada uno de ellos se compara y la etiqueta más grande lexicográficamente se elige como la dominante.

Consideremos el ejemplo en la Figura 3.13 Las etiquetas para los nodos son asignadas sobre el alfabeto $\Sigma = 1, 2,$. A la raíz del árbol se le asigna una letra de Σ y los nodos unidos a la raíz son etiquetados 00 y 01 sucesivamente. En el siguiente nivel, a cada nodo se le asigna una única etiqueta sobre el enlace combinado con el prefijo de su padre. Los nodos E y M son etiquetados con 001 y 010, respectivamente.

3.3.3. Enrutamiento

Para enrutar hacia un destino d un nodo s escoge al enlace de cualquiera de sus vecinos ubicados a dos saltos de distancia que ofrezca la longitud máxima de similitud al comparar la etiqueta de cada uno con la etiqueta basad en prefijos del destino. Eso es simplemente la lógica de máxima similitud la cual selecciona el siguiente salto usando una estrategia *greedy* que considera a los vecinos ubicados a dos saltos de un nodo y puede encontrar caminos más cortos que el enrutamiento tradicional basado en un sólo árbol prefijo, al sacar ventaja de la diversidad de caminos de un LDAG comparado con un árbol hecho mediante prefijos. Por ejemplo, si existe una etiqueta en el *2-vecindario* que es lexicográficamente más parecida al destino, se escoge al siguiente salto tal que el paquete se envía al nodo en lugar de enrutarlo a través del padre en el árbol prefijo. Se asegura que esta estrategia greedy no encuentra un mínimo local al seleccionar aleatoriamente el siguiente salto cuando todos los nodos ofrecen la misma similitud en los prefijos.

A partir de la relación de predecesor inducida por las etiquetas en el LDAG de una MANET, un nodo dado selecciona su siguiente salto mediante su etiqueta de acuerdo a los cuatro posibles casos permitidos por la relación de predecesor. El siguiente lema muestra que PROSE converge al proporcionar rutas correctas.

Lema 3.1 *Las rutas basadas en prefijos construidas usando PROSE son correctas en una red conectada sin cambios en la topología.*
Demostración: Por la definición de las etiquetas, cada nodo tiene una única etiqueta basada en prefijos, y el LDAG es libre de ciclos al construirse. Sin pérdida de generalidad, considere una fuente s y un destino d. Debido a que ellos pertenecen al LDAG, deben satisfacer uno de los cuatro casos de la relación de predecesor, y cada nodo que es elegido como siguiente salto hacia d iniciando en s debe satisfacer también la misma relación. Debido a que el LDAG es finito y no cambian, debe existir al menos un camino libre de ciclos desde s hasta d en el LDAG.

Sin embargo, la topología de una MANET cambia constantemente y los nodos se unen o dejan la red arbitrariamente. Esto resulta en etiquetas inconsistentes en el punto del

etiquetamiento. Para preservar la consistencia, se fuerza a un etiquetado estricto al ordenar las etiquetas usando números secuenciales.

Sea L que representa una tupla (S_n^a, Λ_n^a), donde S_n^a denota el número secuencial que se origina en a y es reenviado por el nodo n, y Λ_n^a denota la etiqueta basada en prefijos del nodo actual con respecto a a. S es un entero que se incrementa monotónicamente, mientras Λ es una palabra en Σ. Se define un operador $\prec$ sobre el conjunto del par ordenado de indentificadores en L. Si L_x^a, L_y^a son dos tuplas entonces:

$$\{L_x^a \prec L_y^a | L_{x,y}^a \in \Sigma\}$$
$$\text{if } \{(S_x^a \prec S_y^a) \vee [(S_x^a = S_y^a) \wedge (\Lambda_x^k) \succ \Lambda_y^i]\}$$

Para satisfacer la afirmación de que esta tupla establece un ordenamiento entre los nodos, se demuestra que el operador $\prec$ es antireflexivo y transitivo sobre el conjunto Σ^*.

Lema 3.2 *La relación $\prec$ es un ordenamiento parcial de tuplas sobre Σ^*.*
Demostración: La primera parte del lema es intuitiva, ya que dos tuplas son iguales si sus números secuenciales y las etiquetas prefijo son iguales. Por lo tanto, $L_x^a \prec L_y^a$ o $L_y^a \prec L_x^a$, sólo se puede mantener a uno y por lo tanto es antireflexiva. Para probar que la transitividad se mantiene, considere tres etiquetas y para conveniencia se les llamaran L_1, L_2, L_3, tal que $L_1 \prec L_2$ y $L_2 \prec L_3$. De la ecuación anterior se puede ver que, $S_1 < S_2 < S_3$ donde las S_s crecen de manera monótona. Esto implica que $S_1 < S_3$. Por lo tanto, si $S_1 \not< S_3$ entonces $L_1 \prec L_3$ sólo si $S_1 = S_4$ y $\Lambda_1 < \Lambda_3$ (la comparación para Λ está dentro del alfabeto). Debido a que se sabe que Λ se mantiene para la transitividad, la relación $\prec$ está en orden sobre Λ^* y establece una relación sucesor-predecesor.

Una vez que se ha demostrado que las etiquetas basadas en prefijos secuenciadas preservan un ordenamiento, se muestra en un teorema más general que el algoritmo de enrutamiento en PROSE converge para caminos libres de ciclos, incluso cuando se tiene una topología dinámica.

Lema 3.3 *Un modelo de enrutamiento basado en prefijos con números secuenciales enruta correctamente.*
Demostración: Del lema 3.1 se sabe que todos los nodos tienen una etiqueta y que cada nodo comparte una relación con todos sus nodos vecinos a distancia uno. Del lema 3.2, se tiene que cada nodo etiquetado está ordenado con respecto a la longitud lexicográfica de la etiqueta y un número secuencial. Por lo tanto, si el algoritmo de enrutamiento enruta entre dos nodos, la estrategia *greedy* garantiza que mejora alguna métrica (conteo de salto, distancia, carga, etc.) hacia algún destino en cada salto. Por lo tanto, un modelo de enrutamiento utilizando etiquetas basadas en prefijos con números secuenciales enruta correctamente, incluso cuando se presentan cambios topológicos.

3.3.4. Construcción de las tablas hash distribuidas

Como se estableció anteriormente, para evitar cuellos de botella y puntos únicos de fallas, se deben diseminar los mapeos de NIDs a etiquetas en toda la red. El protocolo PROSE usa una función hash consistente para lograr tal objetivo. Cada destino publica su existencia al enviar una solicitud de publicación al nodo de quien su etiqueta tenga la mejor coincidencia con el resultado obtenido de aplicar la función *hash* al NID del destino, a este nodo se le llama el ancla del destino. Una solicitud de publicación es una tupla que consiste del NID y la etiqueta de algún nodo. Una fuente que está esperando para

comunicarse con un destino, envía una solicitud de suscripción al nodo de quien su etiqueta tenga la mejor coincidencia con el resultado obtenido de aplicar la función *hash* al NID del destino intencionado. Una solicitud de suscripción consiste del NID, la etiqueta basad en prefijo de la fuente y el NID del destino. El ancla puede responder la solicitud de la fuente o reenviar la solicitud hacia el destino, dependiendo de la naturaleza del intercambio entre la fuente-destino. En este apartado se asume que el ancla responde a la fuente de la solicitud.

El protocolo PROSE usa señalización de estado suave en la diseminación de solicitudes de publicación y suscripción, en el que las fuentes y los destinos son responsables del éxito de sus solicitudes, y las anclas trabajan sobre una base de mejor esfuerzo. Para ahorrar ancho de banda, se agregan las solicitudes hacia diferentes anclas en los mensajes hola intercambiados entre nodos. Para manejar los cambios en la topología de la red, los destinos actualizan sus anclas periódicamente, además, los eventos que cambian la etiqueta prefijo de un nodo también la actualizan. Una actualización refresca el mapeo almacenado en las anclas. Adicionalmente se mantienen temporizadores en los nodos fuentes y anclas para refrescar tales mapeos y asegurar que cualquier información vieja sea eliminada.

3.3.5. Adaptación de PROSE a cambios en la red

Reetiquetamiento de los nodos

Los nodos adquieren una nueva etiqueta cuando se recuperan de algún cambio en la red. Dependiendo del tipo de nodo, la red se organiza de manera diferente. El evento de re-etiquetamiento se divide en uno de unión o uno de abandono. Un evento de unión se maneja de manera similar al proceso de etiquetamiento inicial. Cada nodo adquiere una etiqueta del nodo que es su predecesor inmediato en el punto de unión. Por otro lado, un evento de abandono se maneja de manera diferente para diferentes tipos de nodos. Si el nodo que abandona es un nodo hoja, entonces no se afectan más nodos por este evento. Sin embargo, si el nodo saliente es un nodo interno, entonces la adaptación es más complicada. Esto es principalmente debido a la construcción del LDAG, en el que las etiquetas del subárbol son derivadas de los prefijos o algún ancestro. Sin embargo, mientras los prefijos deben cambiar debido a la reposición, los sufijos permanecen únicos en el subárbol entero.

Para entender más a fondo cómo se maneja el evento en el que un nodo interno abandona la red, consideremos tres nodos x, y y z como se muestra en la Figura 3.14 Estos tres nodos tienen al mismo padre p y el LDAG entero tiene como raíz al nodo r. Sea $k\alpha p$ la etiqueta más reciente de p y sean $k\alpha px$, $k\alpha py$ y $k\alpha pz$ las etiquetas más recientes de sus hijos. Además se asume que existe algún camino entre los tres subárboles tal que un nodo en cada subárbol puede alcanzar a un nodo arbitrario en su subárbol hermano a través de un camino distinto hacia su padre. Si el nodo p fallara o abandonara, se llevan a cabo los siguientes pasos para asegurar que el resto de la red permanezca no afectada en gran parte:

- Cada hijo $k\alpha px$, $k\alpha py$ y $k\alpha pz$ inicia una rutina para descubrir sus nodos hermanos (o aquellos de los que sabía de su existencia) usando rutas distintas a la fallida.
- Conforme cada hijo descubra caminos, el hermano con el identificador global más bajo, digamos $k\alpha px$ se elige como una raíz-secundaria y adquiere la etiqueta de su padre perdido como su propia etiqueta.
- Una vez que se establece una raíz-secundaria, los hermanos restantes continúan usando

su etiqueta basada en prefijos sin modificaciones pero enrutan hacia los demás subárboles usando rutas alternativas.

- Si un nodo estuviera por unirse a cualquiera de los nodos hermanos, digamos $k\alpha px$ el cual tiene caminos más cortos hacia todos sus hermanos entonces la raíz-secundaria renuncia a su posición y se restablece una nueva raíz. El nuevo nodo raíz envía una actualización con un número secuencial más grande para asegurar que sus caminos más cortos no son resultado de un ciclo en la red.

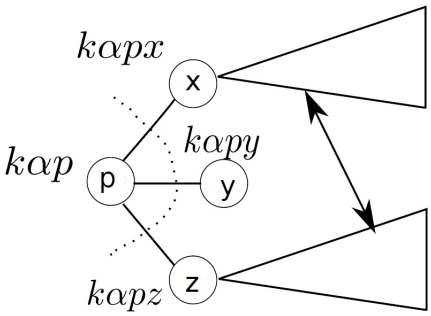


Figura 3.14: Capacidad de recuperación de las etiquetas basadas en prefijos.

Se asume que algún nodo con etiqueta $k\alpha x$ es el ancla designada para el nodo y en la red. El nodo $k\alpha x$ actualiza proactivamente a su ancla con su etiqueta actual. Si el nodo $k\alpha x$ falla o se mueve hacia otro lado en la red, o si no existe algún nodo con tal etiqueta, entonces se designa a un nodo en algún subárbol que tenga la máxima coincidencia prefija como el ancla-secundaria. Se debe notar que el ancla-secundaria está lógicamente en el camino hacia el ancla. Por ejemplo, si un nodo con etiqueta $k\alpha x$ se apagara y el prefijo de esta etiqueta fuera $k\alpha$, entonces el último nodo activo comprometido en proveer el servicio de directorio está en el subárbol α. En el escenario del peor caso, el modelo realiza respaldos en el nodo raíz, debido a que ningún otro nodo tiene un prefijo parecido al destino que se está buscando. Si otro nodo se une como un hijo al mismo padre y adquiere una etiqueta que se parece más al ancla, las actualizaciones proactivas del nodo expiran automáticamente al nodo padre el cual era el ancla y se enrutan hacia la nueva ancla.

En el caso de topologías cambiantes, el costo de reiniciar las etiquetas de un subárbol entero puede ser alto, ya que se depende de la cantidad de nodos que existan en ese subárbol. En el momento en el que algún nodo hijo no sea capaz de alcanzar a cualquiera de sus hermanos, o volverse completamente desconectado, inicia un proceso de re-etiquetamiento. El re-etiquetamiento inicia con el prefijo de un subárbol conocido o con un nuevo LDAG con raíz en el hermano desconectado. Este proceso se determina al establecer un temporizador de espera. El temporizador de espera funciona por un cierto tiempo para permitir que el enlace se reconecte. Cuando el temporizador de espera expira, asume la desconexión e inicia el proceso de reinicio de etiqueta.

3.4. MAR

MAR[11] por sus siglas en inglés (*Multi-label Automatic Routing*), es el pimer protocolo de enrutamiento compacto que alcanza un *stretch* bajos mientras mantiene un estado de enrutamiento bajo en redes móviles. En MAR los nodos se asignan etiquetas así mismos, estas etiquetas están basadas en la posición de los nodos dentro de la red mediante un algoritmo distribuido. Las tablas hash distribuidas están establecidas en ciertos nodos ancla para el etiquetado, una vez que las etiquetas están establecidas, el enrutamiento es automático basado en las etiquetas que reflejan la posición de los nodos dentro de la red y en las búsquedas con las tablas hash distribuidas. Esta manera de enrutar elimina la inundación de paquetes, a diferencia de los protocolos de enrutamiento tradicionales, MAR no necesita tablas de enrutamiento basadas en los destinos. Debido a que MAR mantiene poca información de enrutamiento, con el uso de múltiples etiquetas por nodo, el promedio de la longitud de las rutas encontradas por MAR, es muy cercano al enrutamiento de camino óptimo y además MAR encuentra varios caminos entre el nodo origen y el nodo destino.

3.4.1. Descripción del protocolo

MAR es un esquema de enrutamiento compacto distribuido. Cada nodo en MAR enruta paquetes basándose en las etiquetas de enrutamiento asignadas y en información acerca de su vecindario inmediato. Debido a que el enrutamiento en MAR es automático e incremental y no necesita de la elaboración de tablas de enrutamiento en cada nodo, la complejidad del almacenamiento y de la comunicación crecen de manera sublineal con respecto al número de nodos o al número de enlaces en la red.

3.4.2. Almacenamiento de la información e intercambio de mensajes

Cada nodo en MAR mantiene información acerca de su *2-vecindario*, cada entrada acerca de cada uno de los nodos de su *1-vecindario* contiene: el identificador del nodo, las etiquetas y el número secuencial más reciente obtenido de ese vecino. Para el *2-vecindario*, la información almacenada es la misma que para el primer vecindario, excepto que ahora sólo se intercambia y se almacena la etiqueta básica del árbol para evitar la saturación de mensajes de control. El nodo ancla de otro nodo mantiene el mapeo entre el identificador del nodo y la etiqueta del mismo.

MAR utiliza mensajes *Hello*,*Anchor Update* y *Anchor Reply*. El paquete *Hello* es un mensaje de tipo *broadcast* de vecino a vecino que es originado periódicamente en la raíz con un nuevo número de secuencia y se propaga a través del árbol. El mensaje contiene el identificador de la raíz, el número secuencial de la raíz, el identificador del nodo, el número secuencial del nodo, la etiqueta del nodo, la lista de los identificadores y etiquetas de los nodos en el *1-vecindario* y del *2-vecindario*.

El mensaje *Anchor Request* es un mensaje que va implícito en el primer mensaje de datos enviados desde un nodo que origina el flujo de datos a un nodo ancla . Este mensaje es enviado al nodo ancla de un nodo destino para conocer las etiquetas del nodo al que se desea llegar.

Anchor Reply es el mensaje con el que un nodo ancla contesta a la fuente del flujo, esta respuesta es de tipo *unicast* y contiene el mapeo del identificador del nodo destino a sus etiquetas; después el ancla se encarga de redirigir el tráfico al nodo destino. Todos los nodos envían periódicamente mensajes *Anchor Update* a sus nodos ancla con sus etiquetas asignadas

para así actualizar la información almacenada.

3.4.3. Elección distribuida del nodo raíz

La elección de nodo raíz se realiza de manera distribuida dentro de la red utilizando los mensajes *Hello*. Cuando un nodo se une a la red trata de obtener la etiqueta más pequeña y una raíz del mensaje *Hello* enviado por su vecino, si el nodo no obtiene una etiqueta dentro de un periodo de espera, él se designa a sí mismo como el nodo raíz y después enviará mensajes de tipo *Hello* a su *1-vecindario* para asignarles etiquetas. Cuando un nodo ya etiquetado recibe un mensaje con un identificador del nodo raíz más pequeño que con el que cuenta, éste aceptará a esa nueva raíz y recibirá una nueva etiqueta. Eventualmente el nodo con el identificador más pequeño será elegido como el nodo raíz de la red.

3.4.4. Etiquetas basadas en prefijos en el árbol básico

Un nodo en MAR tiene dos identificadores únicos, el primer identificador es independiente a la ubicación y el segundo es la etiqueta basada en la posición del nodo dentro de la red. Conforme el nodo se mueve dentro de la red, el primer identificador permanece igual pero el segundo será reasignado.

Desde el punto de vista del etiquetamiento, la red es visualizada como un grafo *k-ario*, dirigido, acíclico y etiquetado (*LDAG* por sus siglas en inglés), donde k es el grado del *LDAG*. Cada nodo dentro del *LDAG* es etiquetado de manera similar a como se recorre un grafo por profundidad empezando por el nodo raíz.

Si $\sum$ es el conjunto finito de símbolos, entonces la etiqueta basada en prefijo l de un nodo, es una cadena de símbolos de $\sum$ tal que $|l| \geq 1$. El nodo raíz tiene la etiqueta más pequeña. Una vez que un nodo obtiene una etiqueta con un prefijo, éste le asigna un sufijo único s_i a cada uno de sus hijos i. Después el hijo se asigna a sí mismo la etiqueta $l \bigodot s_i$, donde $\bigodot$ es el operación de concatenación. El mensaje *Hello* que es enviado de vecino a vecino es utilizado para asignar las etiquetas a los nodos de acuerdo con su posición con respecto al nodo raíz y el *LDAG* resultante por la asignación de estas etiquetas es llamado el árbol básico de etiquetas basadas en prefijos (*BPT, Basic Prefix Tree* por sus siglas en inglés).

Un nodo fuente S puede llegar a cualquier nodo destino D recorriendo el *BPT* y este recorrido está basado en la máxima coincidencia del prefijo con las etiquetas de los nodos vecinos. Esto resulta en un recorrido hacia arriba del árbol desde S a un ancestro común y después el recorrido se hace hacia abajo hasta llegar a D, algunas veces el ancestro en común es el nodo raíz.

3.4.5. Asignación de múltiples etiquetas

Cualquier nodo a excepción de la raíz puede recibir más etiquetas además de la recibida por su padre dentro del *BPT*. Estas etiquetas pueden ser recibidas de cualquiera de los nodos vecinos en el *1-vecindario* siempre y cuando ese vecino no se encuentre en un subárbol con respecto a esa etiqueta. Cada nodo publica sus etiquetas a través del mensaje *Hello* y el nodo que recibe el mensaje sólo puede tomar una de todas las etiquetas publicadas por cada vecino. Cada nodo elegirá la etiqueta que esté más disjunta a las etiquetas con las que cuenta como

la nueva etiqueta del nodo vecino, esto asegura que el nodo cuenta con el máximo número de caminos disjuntos para llegar a un destino.

3.4.6. Publicación y operación de suscripción

Una tabla *hash* distribuida es mantenida en la red para almacenar el mapeo entre el identificador del nodo y la etiqueta basada en prefijos y como su nombre lo indica, esta tabla para la búsqueda de nodos está distribuida entre los nodos ancla dentro de la red. El nodo a que almacena la asignación del identificador del nodo a la etiqueta del nodo n es llamado el nodo ancla de n. Una función hash conocida globalmente toma como argumento el identificador del nodo n y regresa la etiqueta basada en prefijos del nodo que será su ancla. El nodo n le informa a su nodo ancla a mediante un mensaje *Anchor Update* incluido en un mensaje *Hello*; el mensaje va viajando a través de la red y sólo el nodo cuya etiqueta coincida con la etiqueta del ancla se convertirá en el ancla del nodo n. Esta operación de suscripción se realiza periódicamente cada vez que un tiempo de espera expira.

Un nodo se suscribe a un destino si éste tiene información que mandarle, la fuente aplica la función hash al identificador del nodo destino y obtiene la etiqueta del nodo ancla, después envía el primer paquete de datos al nodo ancla, éste al recibirlo lo reenvía al nodo destino y envía un mensaje *Anchor Relply* a la fuente con la etiqueta del nodo destino y finalmente la fuente termina de enviar el resto de la información a la etiqueta del nodo destino.

3.4.7. Enrutamiento

En MAR, los paquetes son enrutados utilizando la información acerca del *2-vecindario* y la etiqueta del nodo destino. El nodo encuentra la mayor coincidencia con la etiqueta del nodo destino de entre su *1-vecindario*, en caso de no encontrar una coincidencia, el nodo tomará en cuenta a los nodos en su *2-vecindario*. El uso de múltiples etiquetas permite a los nodos encontrar varios caminos de la misma longitud y otros de diferente hacia el mismo destino y aunque el camino con la mayor coincidencia en las etiquetas es elegido para tener rutas de menor longitud, los caminos encontrados con una mayor longitud son utilizados para evitar la congestión alrededor del nodo raíz.

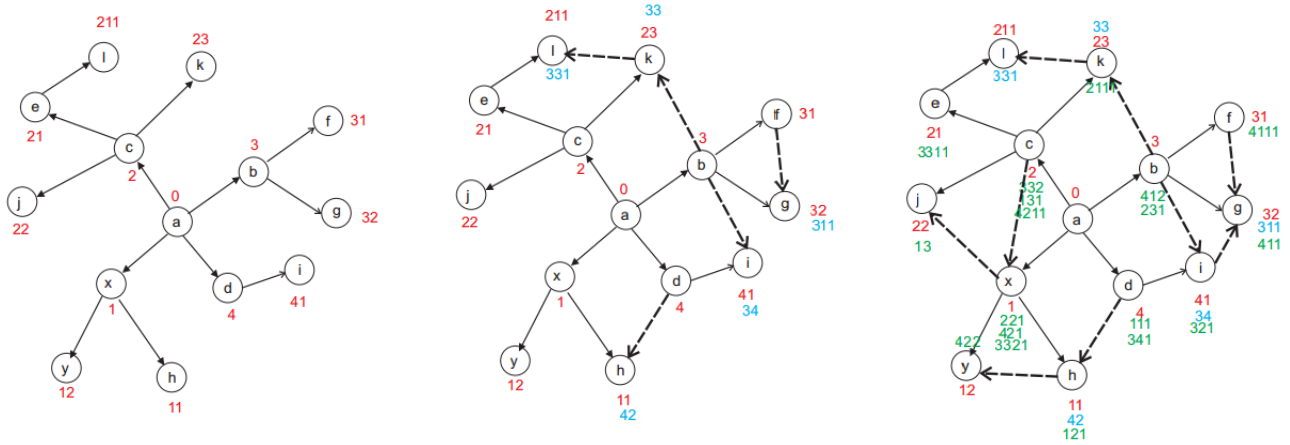
3.4.8. Dinámica de la red

Existen muchas características las cuales permiten a los nodos encontrar sus nuevas etiquetas de manera rápida si alguno de los nodos se mueve o si los enlaces entre ellos fallan. Cuando un nodo se une a la red MAR, éste asume que es la raíz. Si éste encuentra una mejor opción de raíz por medio de los mensajes de saludo con los nodos vecinos, asignará a esa opción como raíz y se re-etiquetará, después elegirá la etiqueta más pequeña de las ofrecidas por sus vecinos como su etiqueta dentro del *BPT* y las demás las almacenará para poder encontrar caminos alternos.

Si un nodo n se cambia de lugar dentro de la red, seguirá el proceso de unión a la red. Si el nodo n era una hoja, ninguno de sus viejos vecinos necesita ser re-etiquetado, sin embargo, si era un nodo interno, los nodos en el subárbol previo tienen que remover la etiqueta vieja y seleccionar la etiqueta menor con la que cuenten como la etiqueta dentro del *BPT*. Si un nodo

con hijos se cambia de lugar o falla, los nodos hijos detectan esto mediante una secuencia de mensajes fallidos. La etiqueta asignada por ese padre es removida y si el nodo era padre dentro del *BPT*, los nodos seleccionarán a la siguiente etiqueta de menor tamaño con la que cuenten como su etiqueta para formar parte del *BPT* otra vez.

Existe una zona de nodos al rededor de cada nodo ancla que también mantienen los mapeos de la tabla hash distribuida del ancla en caso de que el nodo ancla se moviera, manteniendo así la información necesaria para alcanzar al nodo destino. En caso de que un enlace fallara, los nodos realizan una reparación local mediante la utilización de los caminos alternos con los que cuenta y por ahí es que continua con la retransmisión de la información.



(a) Etiquetas basadas en prefijos básicas

(b) Etiquetas recibidas por otros padres

(c) Etiquetas recibidas de otros hijos

Figura 3.15: Ejemplo que muestra el establecimiento de etiquetas en MAR.

En la Figura 3.15(a), se muestra una red ad hoc de 14 nodos. El nodo a ha sido seleccionado como el nodo raíz mediante el algoritmo de elección de nodo raíz distribuido y se le ha asignado la etiqueta "0". También se puede apreciar el *BPT* y las etiquetas de los nodos dentro de él con respecto al nodo raíz. Las etiquetas están en color rojo. En La Figura 3.15(b) se puede observar como después de haberse establecido el *LDAG*, un nodo puede recibir etiquetas adicionales de otros nodos padres. Aquí se muestran estas etiquetas en color azul y los enlaces con líneas punteadas. Como ejemplo tenemos que el nodo l forma la etiqueta "331" añadiendo el número 1 a la etiqueta recibida del nodo k. La etiqueta adicional es elegida debido a que es la más disjunta a la etiqueta con la que cuenta y es lo más pequeña posible. En el caso de del nodo l, la etiqueta "33" de k es más disjunta lexicográficamente a la etiqueta "211" con la que cuenta la otra etiqueta de k la cual es "23". La Figura 3.15(c) muestra cómo etiquetas adicionales enviadas por los hijos de los nodos pueden ser aceptadas, éstas están en color verde. El nodo x es vecino del nodo raíz y tiene 3 etiquetas más recibidas por parte de 3 vecinos de la raíz y estas comienzan con "2", "3" y "4". Estas 4 etiquetas han sido generadas por sus hijos.

3.5. Resumen del capítulo

En este capítulo se expusieron 4 de los trabajos más relacionados con la tesis que se está desarrollando. Se describió a detalle el funcionamiento de cada uno de ellos y los elementos que los conforman. En el siguiente capítulo se expondrá la propuesta de solución y se describirá en qué consiste.

Capítulo 4

Propuesta de solución

En este capitulo se presenta el análisis y diseño de los algoritmos que componen la propuesta de solución a los problemas que fueron detectados en el planteamiento del problema de la presente tesis. Coemzaremos por retomar las ideas expuestas en el planteamiento del problema para detallar los principales retos, así como sus características más importantes.

Como se mencionó en el Capítulo 1, el objetivo de los protocolos de enrutamiento es encontrar una ruta de un nodo origen a un nodo destino dentro de una red de comunicaciones, buscando minimizar siempre las alguna de las métricas de interés como puede ser la longitud de la ruta o la congestión experimentada por los paquetes de datos al transitar dicha ruta. Aunado a los problemas comunes del enrutamiento tradicional, en este trabajo se presentará un esquema de enrutamiento compacto, por lo tanto también es necesario tomar en cuenta los problemas que este tipo de enrutamiento presenta. El primer problema, como ya se mencionó, consiste en encontrar un camino C desde un nodo origen n_s a un nodo destino n_d de longitud mínima en términos del número de nodos que lo componen. Al decir que se busca que el camino cuente con el menor número posible de nodos, es porque lo que se buscará obtener mediante el algoritmo de enrutamiento compacto usando etiquetas basadas en prefijos, son rutas lo más cercano posible a la ruta óptima, con esto queda claro que una de las métricas consideradas en el desarrollo del trabajo fue la longitud de las rutas. Con la longitud de las rutas viene un término para describir la diferencia entre las longitudes de las rutas obtenidas por el protocolo de enrutamiento propuesto y la longitud de la ruta óptima, el *stretch*. En el presente trabajo se busca reducir el *stretch* para poder obtener rutas más cortas a las encontradas por el esquema considerado base.

Para que un protocolo de enrutamiento sea capaz de encontrar caminos de diferentes orígenes a destinos, necesita contar con información para la toma de decisiones. Esta información está almacenada localmente en los nodos en forma de tabla de enrutamiento, la cual juega un papel muy importante en cuanto a la escalabilidad de los protocolos. Para que un protocolo sea considerado escalable, se requiere que la información transmitida de nodo a nodo para poder llenar, actualizar y mantener sus tablas de enrutamiento no generen una sobrecarga en la red. Además de que debe mantener un comportamiento estable sin importar el número de nodos dentro de la red.

Una propiedad del entrutamiento compacto que le permite ser escalable es el tamaño de las tablas de enrutamiento que genera y la cantidad de mensajes necesarios para mantener actualizadas esas tablas. Lo anterior en respuesta a la incapacidad de los protocolos de enrutamiento tradicionales de ser escalables donde el tamaño de las tablas de enrutamiento es proporcional al tamaño de la red.

Un problema detectado en el enrutamiento compacto, específicamente en los protocolos de enrutamiento que inducen un árbol en la red para posteriormente etiquetar a los nodos, fue el de la tormenta de re-etiquetado. El problema que genera la tormenta de re-etiquetado es una gran cantidad de flujo de datos administrativos para poder reparar las rutas rotas por la falla de algún nodo en un período corto de tiempo. Esta acción se realiza con el fin de que todos los nodos debajo del subárbol del nodo que falló, obtengan una etiqueta proveniente de otro nodo padre para reparar al ábrol lógico. Esto genera gran congestión en la red impidiendo así el flujo de paquetes de información.

Por otro lado, este tipo de esquemas de enrutamiento en donde se induce un árbol lógico en la red para enrutar de acuerdo a etiquetas basadas ya sea en prefijos o sufijos, tiende a concentrar el tráfico en las regios de la red cercanas a la raíz del árbol, ésto, debido a que la raíz es el nodo en común para todos los nodos en un etiquetamiento dentro de la red y por lo tanto la mayoría de los caminos pasarán sobre ese nodo. La falta de caminos alternos desde un origen a un destino es una de la causas de la generación de la congestión en esos nodos próximos al origen del etiquetado y se observó que ese es uno de los puntos débiles a mejorar para lograr un balanceo de la carga y así presentar una mayor tolerancia a fallas y todo esto se refleja en un mejor desempeño de la red y una mejor experiencia para los usuarios.

Otro aspecto importante a considerar es que a los usuarios lo que les importa son los contenidos que consumen y no el lugar dónde dichos contenidos se localicen físicamente. A este tipo de enrutamiento en donde no se toma en cuenta la ubicación del contenido sino el contenido en sí, se le conoce como enrutamiento orientado a contenido. Partiendo de ese hecho, se observa que los protocolos utilizados hoy en día siguen separando la identidad de la máquina del tipo de contenido que almacenan, por lo tanto se sigue empleando nodos que guardan la relación entre la dirección IP de las máquinas con la de los identificadores del contenido. Estos nodos serán consultados por los demás nodos siempre que se desee conocer la ubicación de un contenido en específico, generando así cuellos de botella y congestión en esas zonas de la red.

4.1. Descripción general

Considerando todos los aspectos de la problemática anteriormente expuestos, en la presente tesis se propone un algoritmo de enrutamiento compacto usando etiquetas basadas en prefijos como una propuesta de solución. Este algoritmo plantea obtener dos ordenamientos al mismo tiempo en una misma red. Lo anterior se logrará mediante la selección de dos nodos que actuarán como nodos iniciadores de dos etiquetados.

Dichos nodos serán seleccionados de acuerdo con diferentes criterios que tienen como objetivo mejorar la calidad de los etiquetados en términos del *stretch* y de la congestión. Una vez que se han determinado los nodos raíz, por cada uno de ellos se realizará un búsqueda por amplitud de la red y mientras va descubriendo nodos, les asignará una etiqueta única que servirá para identificar a cada nodo dentro de la red. Gracias a que habrá dos nodos que iniciarán dos etiquetados, cada nodo dentro de la red recibirá 2 identificadores (etiquetas) únicos dentro de cada uno de los etiquetados.

Una vez que se ha terminado de recorrer la red y de haber etiquetado todos los nodos, el enrutamiento se realiza en base a las etiquetas, esto es, buscando la mayor coincidencia posible con la etiqueta del nodo destino. Cada nodo es capaz de localizar al nodo destino mediante una función *hash* determinista conocida por todos los nodos de la red. Al contar con esta

función *hash*, el enrutamiento dejará de ser para nodos y pasará a ser orientado a contenido. Lo anterior gracias a que la función *hash* mapea el contenido que se desea buscar con el espacio de etiquetas (nombres de los nodos) que se utilizan en la red.

Al haber dos etiquetados dentro de la misma red, cada nodo podrá elegir en cuál de los dos etiquetados le conviene enrutar los paquetes para así minimizar la distancia o evitar congestiones. Al haber dos rutas al mismo destino, si un nodo falla, el proceso de re-etiquetado no es una tarea urgente, por lo tanto al momento de reparar el etiquetado que se ha perdido no generará un tráfico intenso de paquetes en un periodo de tiempo pequeño. Además de los beneficios mencionados anteriormente, el tamaño de las tablas de enrutamiento será de tamaño reducido y a lo más tendrá d entradas, donde d es el grado máximo de los nodos que componen la red, conservando así las propiedades del enrutamiento compacto y permitiéndole adaptarse aun si el número de nodos es muy grande.

4.2. Etapas del algoritmo

El algoritmo propuesto se compone en tres etapas principales:

- Selección de los nodos raíz.
- Etiquetado de la red.
- Enrutamiento dentro de la red.

A continuación se describirá a detalle cada una de las tres etapas que forman parte del algoritmo de enrutamiento compacto usando etiquetas basadas en prefijos.

4.2.1. Selección de los nodos raíz

Antes de comenzar a definir los criterios de selección, es necesario definir cuál es el esquema considerado como base. El esquema que se tomó como referencia es un esquema de enrutamiento compacto que utiliza etiquetas basadas en prefijos, donde el ordenamiento de la red depende únicamente de un nodo iniciador del etiquetado y donde este nodo es elegido de manera arbitraria. Una vez que se ha elegido al nodo raíz, se procede a etiquetar a los nodos que componen la red y con base en esas etiquetas se realiza el enrutamiento buscando siempre la mayor coincidencia de los prefijos en las etiquetas.

Como mejora a este esquema base se plantean tres diferentes maneras de seleccionar a un sólo nodo raíz que tienen como objeto producir etiquetados de mayor calidad en términos del *stretch* y de la congestión. Dichos experimentos y resultados serán expuestos en el siguiente capítulo de este trabajo. A continuación se listan los criterios para la selección de un sólo nodo raíz en el esquema base.

- Nodo con mayor grado.
- Centro de la red. Nodo con cuya distancia máxima hacia otros nodos de la red es mínima.
- Centro promedio de la red. Nodo con cuya distancia promedio hacia otros nodos de la red es mínima.

Nodo con mayor grado

Haciendo alusión al titulo, aquí se buscará al nodo que cuente con el grado mayor en toda la red, con la finalidad que sea el nodo raíz del árbol lógico que se inducirá por medio del algoritmo BFS (*Breadth First Search*). Partiendo de la matriz de adyacencia que se genera al empezar a tratar al grafo con el que se está trabajando, se ejecutará el Algoritmo 1 el cual arrojará como resultado el identificador del nodo que fungirá como el iniciador del etiquetamiento gracias a que posee el mayor grado de la red. El algoritmo logra ese resultado con una complejidad de $O(n \times n)$. Para ejemplificar esta selección se tomará un ejemplo de una red de que posee 30 nodos, dicho ejemplo se muestra en la Figura 4.1 donde se pueden apreciar a los nodos y sus enlaces con otros nodos. Siguiendo con el ejemplo, al ejecutar el algoritmo de selección de nodo raíz se asigna al nodo 4 como la raíz ya que aunque no es fácil de apreciarlo su grado es de 13.

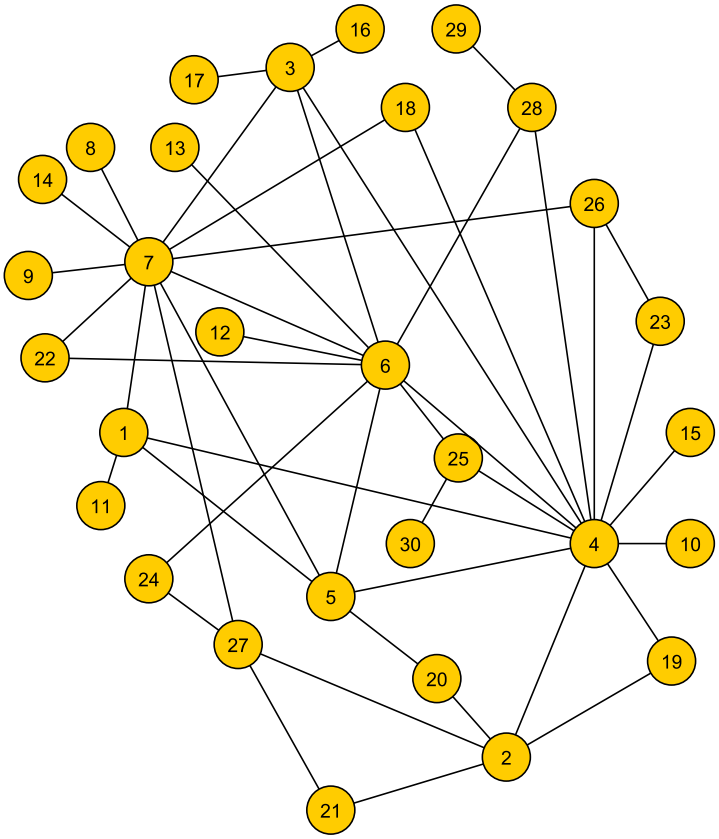


Figura 4.1: Grafo original.

Require: Una matriz de Adyacencia $M[i, j]$ y un vector V para almacenar el grado del nodo.
Ensure: Un nodo raíz p.
Procedimiento *Grado (M){*
for *i:=1* ***to*** n **do**
for *j:=1* ***to*** n **do**
$V[i]$:= suma de $M[i, j]$;
for *i:=1* ***to*** n **do**
Buscar en V el elemento mayor y su posición p;
Asignar al nodo p como nodo raíz. }

Algoritmo 4: Selección de centro:Nodo con grado mayor.

Para la demostración del Teorema 1 que estipula que el Algoritmo funciona correctamente, se parte de tres suposiciones.

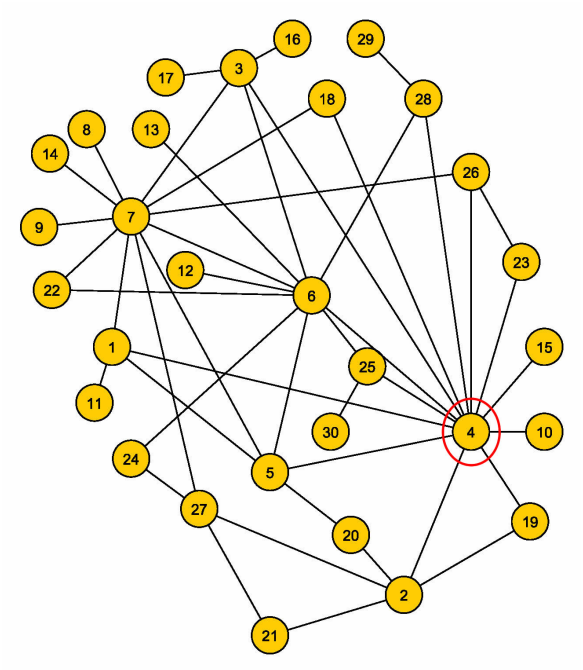
- La matriz de adyacencia M se construye de manera correcta.
- Las operaciones de suma en la matriz M son correctas.
- El llenado del vector V se realiza correctamente con las sumas de elementos de los renglones de M.

Teorema 1. *Dado un grafo codificado por medio de su matriz de adyacencia M, el Algoritmo selecciona al nodo de grado máximo.*

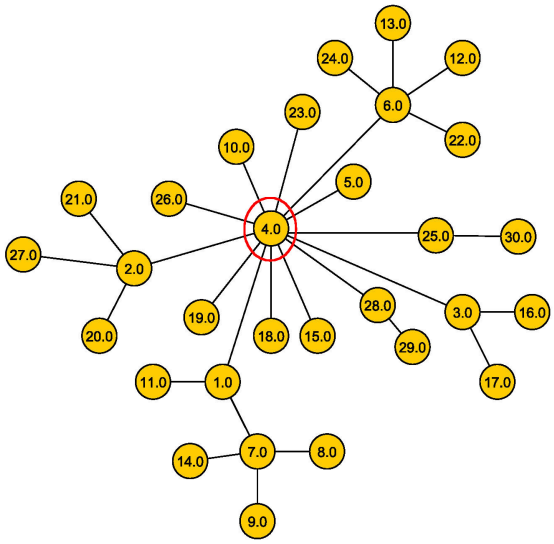
Demostración. Procedamos por contradicción y supongamos que existe un nodo f que cuenta con un grado mayor que el nodo p encontrado por el algoritmo. Por lo tanto p no es el nodo con mayor grado en la red. Al realizar la suma de los renglones de M para llenar V, el renglón de f suma un número mayor que el renglón de p. Al realizar la búsqueda en V del elemento mayor, el grado de f sería el elegido al ser considerado el grado mayor. Debido a que el algoritmo al buscar el elemento mayor, encontró como nodo con grado mayor a p, quiere decir que el nodo f no sumó mayor grado que p por lo tanto p si es el nodo con grado mayor.

□

En la Figura 4.2 se puede observar cómo después de la selección del nodo 4, el cual es el nodo con el grado mayor, se induce un árbol lógico en la red con raíz en ese nodo y dicho árbol será etiquetado para poder realizar el enrutamiento dentro de la red con base en las etiquetas basadas en prefijos.



(a) Selección de centro.



(b) Árbol inducido en la red al seleccionar el nodo raíz y ejecutar el algoritmo BFS.

Figura 4.2: Inducción del árbol lógico de acuerdo con el nodo raíz con grado mayor.

Centro de la red

La selección del centro de la red bajo el criterio de centro de la red se realiza con base en la matriz de distancias más cortas entregada al finalizar la ejecución del algoritmo Floyd-Warshall sobre la matriz de adyacencia del grafo original. Aquí lo que se busca es el nodo cuya distancia máxima a cualquier nodo sea la distancia más pequeña de entre todos los nodos. La tarea anteriormente descrita es ejecutada en el Algoritmo 5 con la misma complejidad que el algoritmo anterior, es decir $O(n \times n)$.

Require: Una matriz de distancias más cortas D[i,j] y un vector V para almacenar la distancia máxima de un nodo a otro nodo.
Ensure: Un nodo raíz p.
Procedimiento *Centro (D){*
for *i:=1* ***to*** n **do**
for *j:=1* ***to*** n **do**
$V[i]$:= valor máximo en $M[i, j]$;
for *i:=1* ***to*** n **do**
Buscar en V el elemento menor y su posición p;
Asignar al nodo p como nodo raíz. }

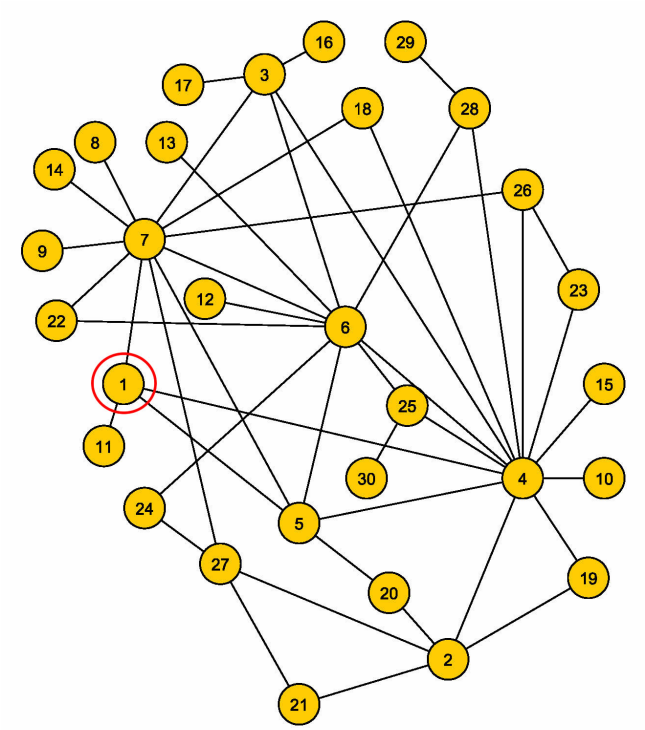
Algoritmo 5: Selección de centro: Centro de la red.

Para la demostración del Teorema 2 que estipula que el Algoritmo 5 funciona correctamente se parte de tres suposiciones.

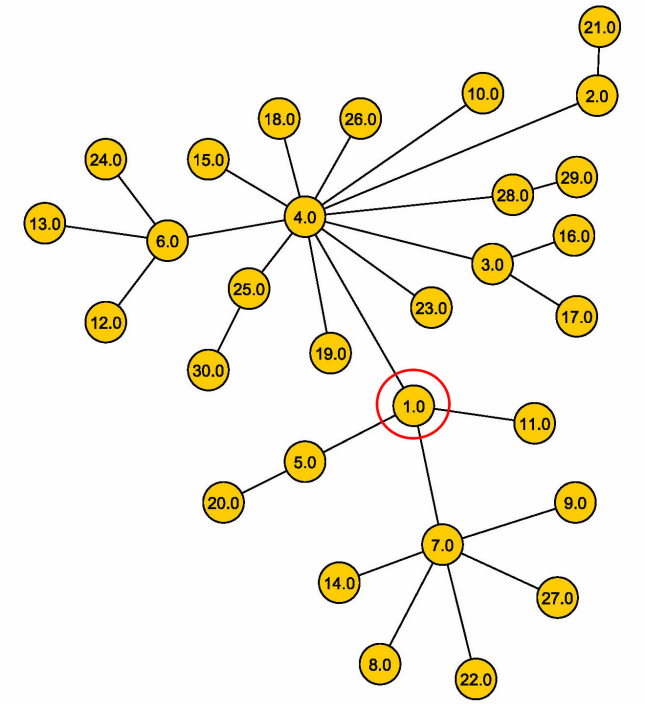
- La matriz de distancias más cortas D se construye de manera correcta.
- Las operaciones de búsqueda de el valor más grande en los renglones de la matriz D son correctas.
- El llenado del vector V se realiza correctamente con los elementos mayores de los renglones de M.

Teorema 2. *Dada una matriz con las distancias más cortas D entre todos los nodos, el Algoritmo 5 selecciona al nodo que considerado el centro de la red.*

Demostración. Se procede por contradicción y se supone que existe un nodo f que cuenta con una distancia mayor más pequeña que el nodo p encontrado por el algoritmo, por lo tanto p no es el centro en la red. Al realizar las búsquedas de los elementos mayores en D por renglón para llenar V, el valor del renglón del nodo f es menor que el del nodo p. Al realizar la búsqueda en V del elemento menor, el valor de f sería el elegido al ser considerado la distancia mayor más chica. Debido a que el algoritmo al buscar el elemento menor, encontró como nodo centro a p, quiere decir que el nodo f no tenía una distancia mayor más chica que p por lo tanto p si es el centro de la red. □



(a) Selección de centro.



(b) Árbol inducido en la red al seleccionar el nodo raíz y ejecutar el algoritmo BFS.

Figura 4.3: Inducción del árbol lógico de acuerdo con el nodo raíz centro de la red.

Para ejemplificar de la misma manera que en el criterio anterior, en la Figura 4.3 se muestra cómo después de la ejecución del algoritmo de selección de centro se decide que el nodo raíz sea el nodo 1, debido a que su distancia mayor hacia otro nodo fue la más pequeña a comparación de los demás; después se observa cómo queda el árbol lógico inducido sobre la red que tiene como raíz al nodo 1.

Centro promedio

En este tercer criterio para la elección de un nodo raíz, se trabaja sobre la misma matriz de distancias más cortas entregadas por el algoritmo Floyd-Warshall. Aquí lo que se busca es al nodo cuya suma de todas las distancias mínimas hacia todos los demás nodos sea la suma más pequeña. A continuación se presenta al Algoritmo 6 que es el responsable de la elección del nodo raíz con base a la suma de sus distancias mínimas. Al igual que los algoritmos anteriores, la complejidad del algoritmo es de $O(nxn)$.

Require: Una matriz de distancias más cortas D[i,j] y un vector V para almacenar la suma de las distancias mínimas a los demás nodos.
Ensure: Un nodo raíz p.
Procedimiento *Centro_ Promedio (D){*
for *i:=1* ***to*** n **do**
for *j:=1* ***to*** n **do**
$V[i]$:= suma de $M[i, j]$;
for *i:=1* ***to*** n **do**
Buscar en V el elemento menor y su posición p;
Asignar al nodo p como nodo raíz. }

Algoritmo 6: Selección de centro:Centro promedio.

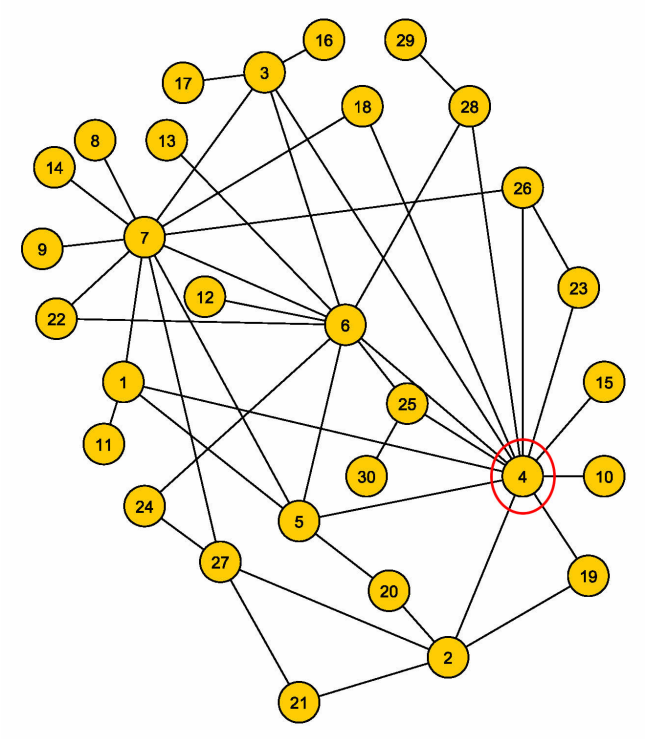
Para la demostración del Teorema 3 que estipula que el Algoritmo 6 funciona correctamente se parte de las siguientes suposiciones.

- La matriz de distancias más cortas D se construye de manera correcta.
- Las operaciones de suma en los renglones de la matriz D son correctas.
- El llenado del vector V se realiza correctamente con la suma de los renglones de M.

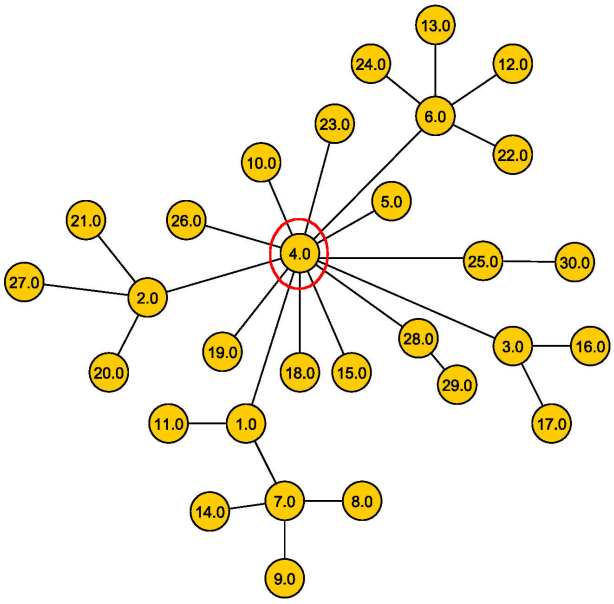
Teorema 3. *Dada una matriz con las distancias más cortas D entre todos los nodos, el Algoritmo 6 selecciona al nodo considerado el centro promedio de la red.*

Demostración. Para la demostración se procederá por contradicción por lo tanto, supongamos que existe un nodo f que cuenta con una suma de distancias menor que el nodo p encontrado por el algoritmo, por lo tanto p no es el centro promedio de la red. Al realizar las sumas en D por renglón para llenar V, el valor del renglón del nodo f es menor que el del nodo p. Al realizar la búsqueda en V del elemento menor, el valor de f sería el elegido al ser considerado suma menor. Debido a que el algoritmo al buscar el elemento menor, encontró como nodo centro promedio a p, quiere decir que el nodo f no tenía una suma menor que p por lo tanto p si es el centro promedio de la red. □

La ejecución del algoritmo anterior, dentro del ejemplo que se ha venido manejando arroja como resultado al nodo 4 nuevamente ya que la suma obtenida de sus distancias mínimas fue la más pequeña de entre todos los nodos. En la Figura 4.4 se muestra el mismo resultado que en la Figura 4.2, con la diferencia que el mismo nodo fue elegido por un criterio diferente al anterior.



(a) Selección de centro.



(b) Árbol inducido en la red al seleccionar el nodo raíz y ejecutar el algoritmo BFS.

Figura 4.4: Inducción del árbol lógico de acuerdo con el nodo raíz centro promedio.

Dos nodos raíz

Retomando al algoritmo propuesto, en éste se busca que haya dos nodos los cuales se encarguen de asignar dos etiquetas a todos los nodos dentro de la red. Debido a que se busca obtener los mejores resultados posibles al momento de realizar el enrutamiento, aquí se experimentará con criterios para la selección de esos dos nodos ya que como se planteó en el primer capítulo, la elección de los nodos raíz influye en la longitud de las rutas dentro de la red. A continuación se describirán los criterios con los cuales se experimentó para la selección de dos nodos raíz.

- Centro promedio de la red y el nodo más lejano a él.
- Los dos nodos más alejados entre sí.
- Dos nodos arbitrarios.

Centro promedio de la red y el nodo más alejado a él

Basado en la selección de un sólo nodo raíz con el criterio de centro promedio de la red, aquí lo que se busca es el segundo etiquetado comience lo más alejado al primero para que a los nodos se les asignen etiquetas que no se parezcan tanto y así evitar la congestión y poder generar rutas alternas. A continuación se muestra el Algoritmo 7 que se encarga de la selección de ambos nodos raíz y que también trabaja sobre la matriz de distancias más cortas dada por la ejecución del algoritmo Floyd-Warshall sobre la matriz de adyacencia del grafo original y mantiene la complejidad de $O(n \times n)$.

Require: Una matriz de distancias más cortas D[i,j], un vector V para almacenar la suma de las distancias mínimas a los demás nodos y un vector U para almacenar la distancia del nodo más alejado.
Ensure: Dos nodos raíz p y q.
Procedimiento *Centro_ y_ lejano (D){*
for *i:=1* ***to*** n **do**
for *j:=1* ***to*** n **do**
$V[i]$:= suma de $D[i, j]$;
$U[i]$:= valor máximo en $D[i, j]$;
for *i:=1* ***to*** n **do**
Buscar en V el elemento menor y su posición p;
Buscar $U[p]$ y guardar el valor x;
Buscar en $D[p, j]$ a x y guardar su posición q
Asignar al nodo p y al nodo q como nodos raíz. }

Algoritmo 7: Selección de dos centros. Centro promedio y nodo más lejano a él.

Para la demostración del Teorema 3 que estipula que el Algoritmo 7 funciona correctamente se parte de cinco suposiciones.

- La matriz de distancias más cortas D se construye de manera correcta.
- Las operaciones de suma en los renglones de la matriz D son correctas.
- El llenado del vector V se realiza correctamente con la suma de los renglones de D.
- El llenado del vector U se realiza correctamente con el valor máximo por cada renglón de D

- La selección del centro promedio se realiza correctamente.

Teorema 4. *Dada una matriz de distancias más cortas D entre todos los nodos, el Algoritmo 7 seleccionará al nodo considerado el centro promedio y al nodo más alejado a él.*

Demostración. Procederemos por contradicción. Supongamos que existe un nodo f que está más alejado del nodo p con una distancia d_f que el nodo q con su distancia x. Al ser f el nodo más alejado del nodo p, la distancia d_f será el valor que se guardará en U para el nodo p. Al realizar la búsqueda del elemento mayor en $D[p, j]$ para saber la identidad del segundo nodo raíz, no se buscó el valor d_f, sino el valor x. Esto quiere decir que el nodo f no era el nodo más alejado al nodo p ya que de haberlo sido se hubiera buscado el valor d_f y no x. Por lo tanto el nodo q si es el nodo más alejado a p. □

Al terminar de ejecutarse el algoritmo 7 se obtiene como resultado dos nodos que serán los encargados de etiquetar la red. Cada nodo inducirá un árbol lógico dentro de la red, por lo tanto todos los nodos pertenecientes a la red serán parte de ambos árboles. Se continuará ejemplificando la selección de los nodos con la red que se ha manejado, en este caso en la Imagen 4.5, se quiere explicar cómo a partir de una red y de la ejecución del algoritmo anterior se obtienen dos ordenamientos al mismo tiempo, uno comenzado por el nodo 4 y el otro por el nodo 8. El nodo 4 fue elegido por se el nodo con la suma de distancias mínimas más pequeña, el nodo 8 por ser el nodo más alejado al nodo 4.

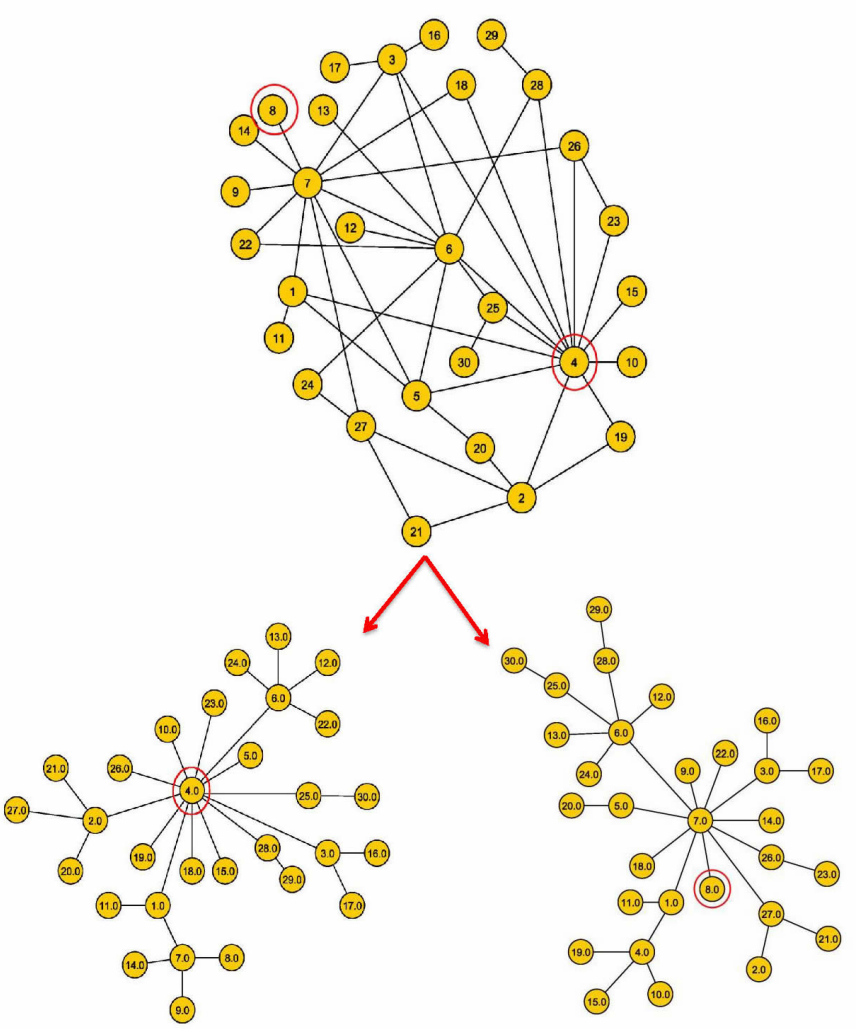


Figura 4.5: Dos árboles de búsqueda por amplitud generados a partir de seleccionar diferentes nodos raíz.

Los dos nodos más alejados entre sí

Este es el segundo criterio propuesto para elegir a dos nodos raíz. Aquí lo que se busca es elegir a los dos nodos más alejados entre sí. Al igual que los criterios anteriores, el algoritmo que buscará a esos dos nodos trabaja sobre la matriz de distancias más corta proporcionada por el algoritmo Floyd-Warshall. Simplemente se buscará el valor mayor de las distancia más corta y se observará a qué nodos pertenece dicha distancia para designarlos como nodos raíz. En el Algoritmo 8 se muestra el pseudocódigo para la selección de los dos nodos más alejados entre sí dentro de la red. La complejidad para el siguiente algoritmo sigue siendo la misma que en los anteriores $O(n \times n)$

Require: Una matriz de distancias más cortas D[i,j], un vector V para almacenar los valores de las distancias mínimas más grandes por nodo.
Ensure: Dos nodos raíz p y q.
Procedimiento *Separados (D){*
for *i:=1* ***to*** n **do**
for *j:=1* ***to*** n **do**
$V[i]$:= valor máximo en $D[i, j]$;

for *i:=1* ***to*** n **do**
Buscar en V el elemento mayor x y obtener su posición p;
Buscar en $D[p, i]$ a x y guardar la posición q en la que fue encontrado;
Asignar al nodo p y al nodo q como nodos raíz. }

Algoritmo 8: Selección de dos centros. Nodos más alejados entre sí.

Para la demostración del Teorema 5 que estipula que el Algoritmo 8 funciona correctamente se parte de tres suposiciones.

- La matriz de distancias más cortas D se construye de manera correcta.
- Las operaciones de búsqueda del valor máximo en los renglones de la matriz D son correctas.
- El llenado del vector V se realiza correctamente con los valores máximos por cada renglón de D

Teorema 5. *Dada una matriz de distancias más cortas entre todos los nodos D, el Algoritmo 8 seleccionará a los dos nodos más alejados entre sí.*

Demostración. Se procederá por contradicción, por lo tanto se supone que existen dos nodos f_1 y f_2 que son los nodos más separados en la red con una distancia d_f, y d_f es mayor que la distancia x entre el nodo p y el nodo q. Al realizar la búsqueda del valor máximo por renglón en la matriz D, se guarda el valor d_f en el vector V. Al realizar la búsqueda del valor máximo en el vector V, el valor d_f sería el elegido al ser considerado el valor mayor. Debido a que el algoritmo al buscar el elemento mayor, encontró a x como el valor máximo, quiere decir que el d_f no era el valor de la distancia mayor entre dos nodos, teniendo como consecuencia que el nodo f_1 y el nodo f_2 no eran los nodos más separados y por lo tanto el nodo p y el nodo q si son los nodos más alejados entre sí. □

En la Figura 4.6 se muestra que nodos serían los elegidos como raíz al ejecutar el algoritmo anteriormente descrito. En este ejemplo de la red de 30 nodos, los nodos raíz serían en nodo 8 y el nodo 10 ya que son los más alejados en toda la red, después de eso inducirían un árbol lógico en la red.

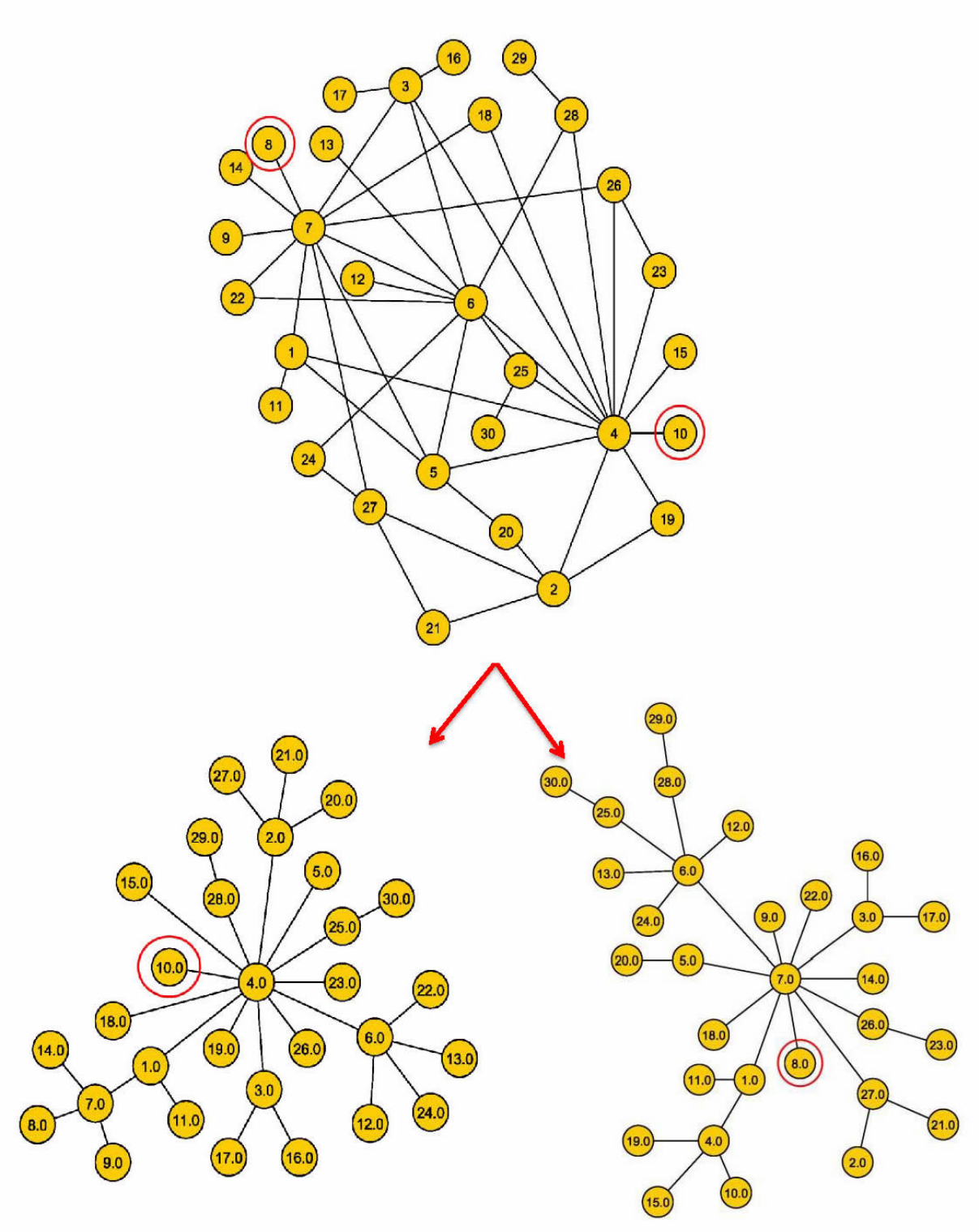


Figura 4.6: Dos ordenamientos obtenidos a partir de seleccionar como raíz a los nodos más alejados entre sí.

Dos nodos arbitrarios

Al igual que en los criterios anteriores, al elegir dos nodos arbitrariamente, se inducirá un árbol lógico por cada nodo raíz en la red que estarán enraizados en dichos nodos. Esta manera de elegir a los nodos es para comparar los resultados en los experimentos y poder verificar si en realidad, al invertir recursos de los nodos en la elección de los nodos ayuda en la generación de rutas más cortas dentro de la red. En la Figura 4.7 se continua con la red de ejemplo que se ha manejado. Después de haber ejecutado el algoritmo donde simplemente se eligen a dos nodos de manera aleatoria se obtuvo como resultado que el nodo 4 y el nodo 16 fueron los elegidos como raíz.

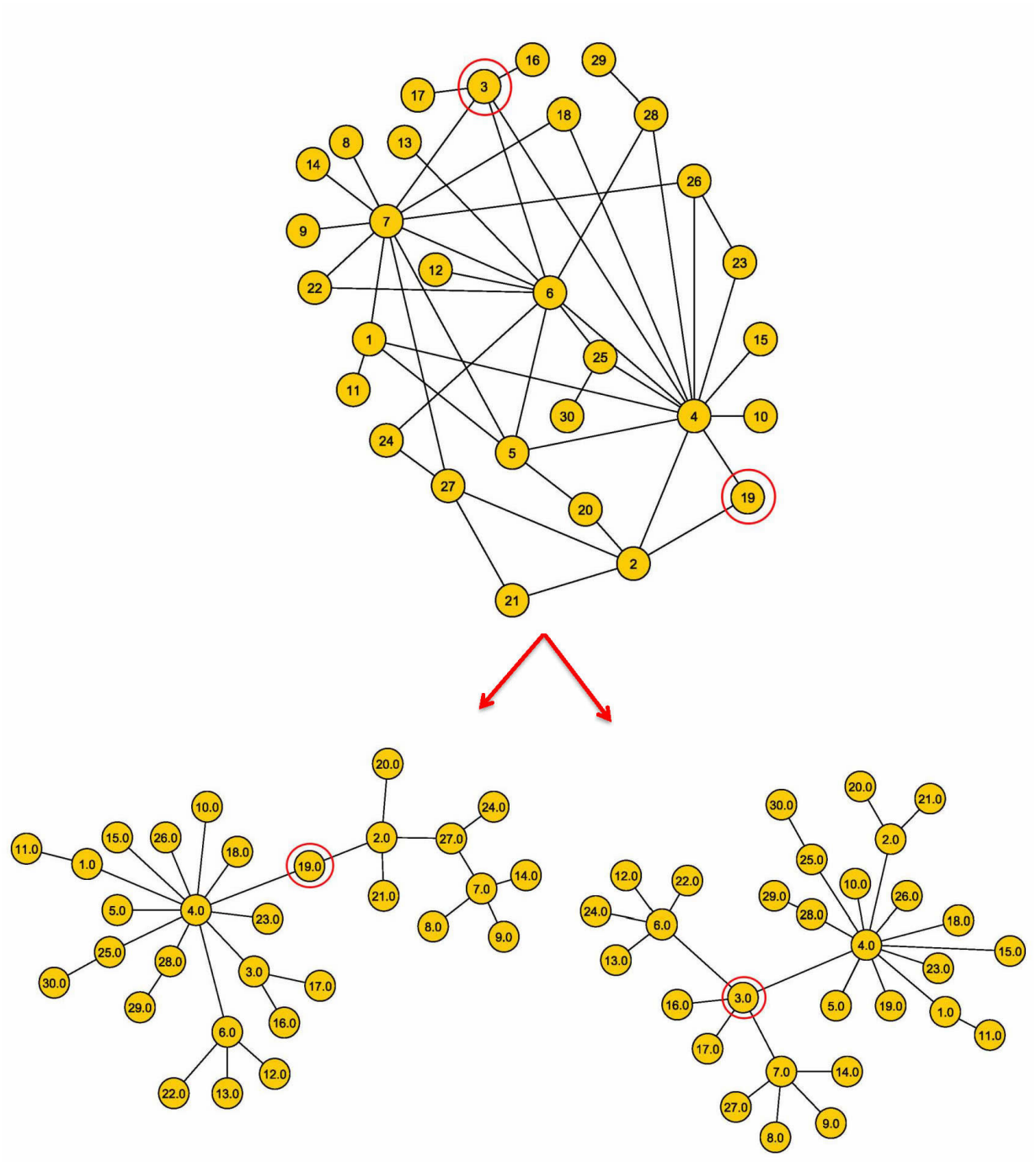


Figura 4.7: Dos ordenamientos obtenidos a partir de seleccionar como raíz a dos nodos arbitrarios.

4.2.2. Etiquetado

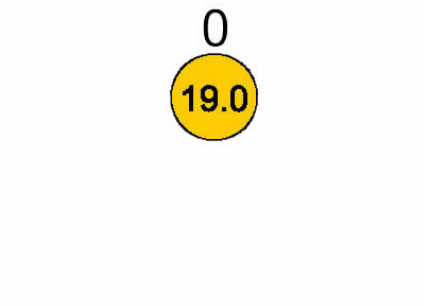
Una vez que se ha seleccionado a los nodos que serán los nodos raíz dentro de la red, el siguiente paso es inducir el árbol lógico dentro de la red mediante el algoritmo para recorrer grafos BFS (Búsqueda por amplitud), conforme se van descubriendo los nodos de la red y se van generando las relaciones padre e hijo entre los nodos, se irá asignando una etiqueta única a cada nodo.

La etiqueta servirá para poder identificar y ubicar al nodo dentro de la red. Cada nodo recibirá dos etiquetas en la red, por lo tanto contará con tres identificadores, uno independiente de la posición en la que se encuentre en la red y que es con que inicia y los otros dos, las etiquetas que sí son dependientes de la posición en la que estén en relación con los nodos designados como raíz. Las etiquetas también reflejarán la relación que existe entre los nodos dentro del árbol lógico ya que al estar basadas en prefijos se podrá observar qué nodo es padre de qué otros nodos. Al estar basadas en prefijos y al asignarse al estar descubriendo los nodos en el algoritmo BFS se puede ver que el padre será el que le asigne las etiquetas a sus hijos. Todos los nodos a excepción del nodo raíz recibirán una etiqueta de su padre.

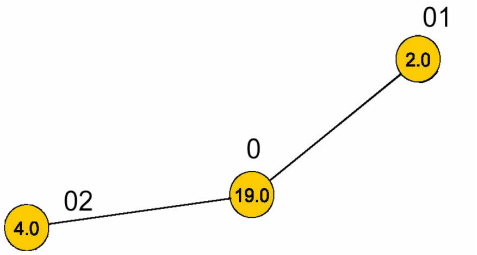
La raíz será el único nodo que se asigne la etiqueta a sí misma, esto debido a que es la iniciadora del etiquetamiento. El nodo raíz siempre se asignará la etiqueta '0'y después al ir descubriendo a sus hijos les asignará una etiqueta única formada de su propia etiqueta más una letra de un alfabeto finito de símbolos. La forma en que un nodo etiqueta a sus hijos se define de la siguiente manera.

Si $\sum$ es el alfabeto que contiene una cantidad finita de símbolos y $\sum^*$ es el conjunto de todas las cadenas de símbolos sobre $\sum$ tal que $|\sum| \geq 2$, entonces una etiqueta basada en prefijos l dentro de $\sum^*$ es la concatenación de la etiqueta del padre l con un sufijo único s_i, tal que $l \odot s_i$, donde $\odot$ es el operador de la concatenación. Una vez que el padre ha recibido una etiqueta basada en prefijos, él empezará con la asignación de etiquetas a sus i hijos que posteriormente etiquetarán a los nodos que encuentren como hijos, así hasta tener toda la red etiquetada.

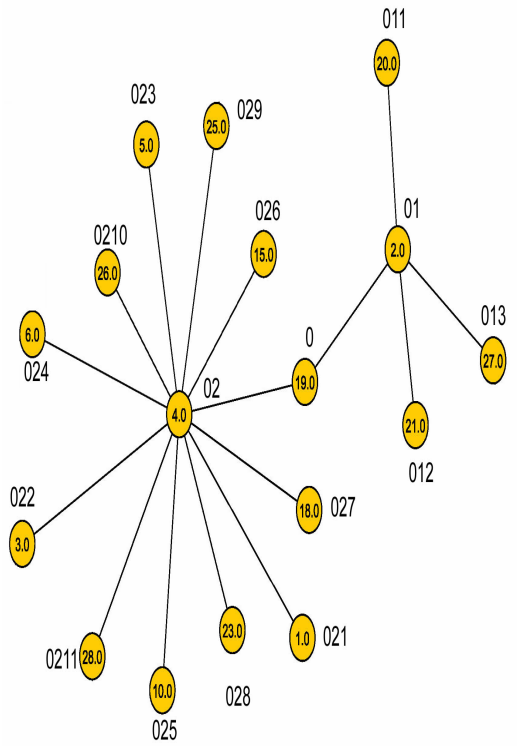
En la Figura 4.8 se muestra el proceso de etiquetado a la par que se van descubriendo los nodos por medio del algoritmo de búsqueda por amplitud. En la Figura 4.8(a) se puede ver que el nodo raíz siempre se asigna a él mismo la etiqueta 0. En la Figura 4.8(b) se muestra cómo la raíz le asigna una etiqueta única a sus hijos concatenando su propia etiqueta con un sufijo que les asigna al ir descubriendo a cada uno con la búsqueda por anchura. Para el caso de la Figura 4.8(c) se repite el proceso anterior, es decir, cada nodo etiqueta a sus hijos y ese proceso se realiza hasta que se han descubierto todos los nodos dentro de la red y se han etiquetado al mismo tiempo para obtener como resultado la imagen mostrada en la Figura 4.8(d). Esta red de ejemplo es la que se ha manejado desde que comenzó el presente capítulo y para ejemplificar el proceso de etiquetado se eligió la selección de un nodo raíz con el criterio de nodo arbitrario, el nodo elegido fue el nodo 19.



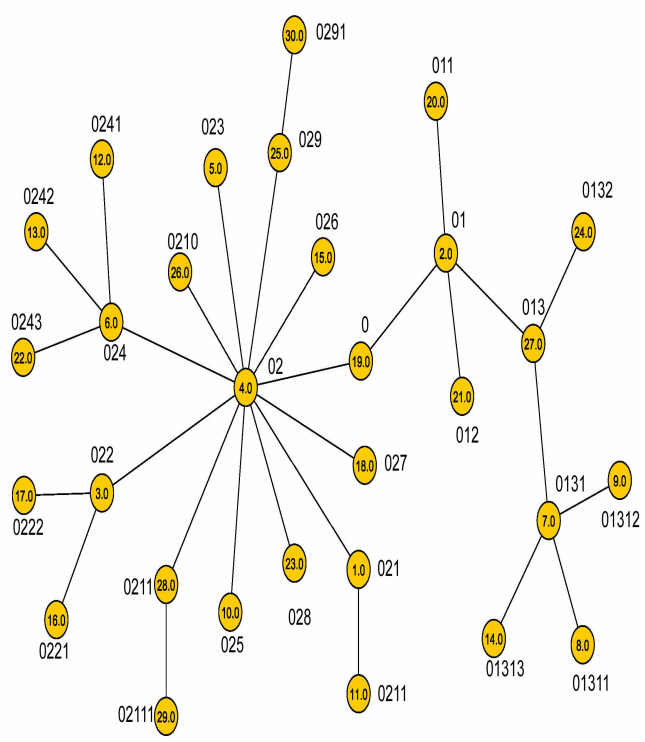
(a) La raíz se asigna la etiqueta 0.



(b) La raíz etiqueta a sus hijos al descubrirlos durante el proceso de búsqueda por amplitud.



(c) Los nuevos nodos padres etiquetan a sus hijos descubiertos.



(d) Cada nodo etiqueta a sus hijos hasta que la red está completamente etiquetada

Figura 4.8: Ejemplo del etiquetado de la red.

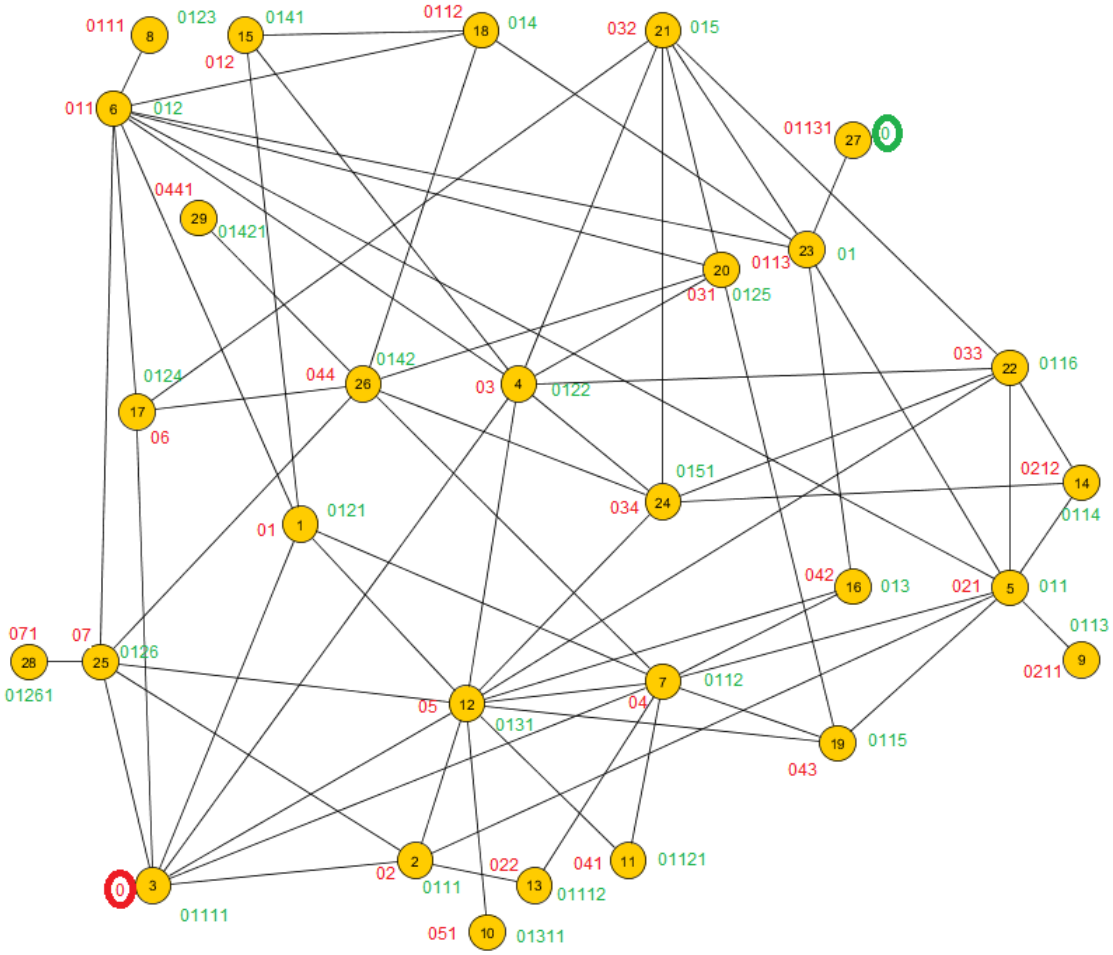


Figura 4.9: Doble etiquetado en la red.

En el algoritmo que se propone en este documento se busca obtener un doble ordenamiento por lo tanto el proceso de etiquetado que se describió anteriormente se realiza dos veces, comenzando en cada nodo que se eligió como raíz. Para ejemplificar este doble ordenamiento de la red, se tomará el ejemplo de los dos nodos más alejados entre si en una red de 29 nodos. Los ordenamientos de la red corren a cargo del nodo 27 y 3. En la Figura 4.9 el color verde indica el etiquetado comenzado desde el nodo 27 y el rojo el etiquetado asignado desde el nodo 3.

Teorema 6. *Cada nodo recibirá una etiqueta basada en prefijos única de cada etiquetamiento.*

Demostración. Gracias a la definición de la forma en que se construyen las etiquetas basadas en prefijos y debido a que cada nodo al irse descubriendo mediante el algoritmo BFS establece y mantiene una relación de padre-hijo con los nodos en su *1-vecindario* para formar un árbol lógico acíclico, no habrá manera que se repitan una etiqueta dentro de un mismo etiquetado. □

4.2.3. Enrutamiento

Enrutamiento en un único etiquetado

El enrutamiento en el algoritmo propuesto se realiza con base en las etiquetas basadas en prefijos asignadas durante la etapa anterior. Cada nodo enrutará los paquetes utilizando la información de su *1-vecindario* y la etiqueta basada en prefijos del nodo destino. Cada nodo revisará la etiqueta del nodo al cual se dirige el paquete y la comparará con cada una de las etiquetas de los nodos que tiene como vecinos, buscando así la mayor coincidencia en las etiquetas, una vez que ha encontrado la etiqueta de su vecino que tiene un mayor número de dígitos que coinciden con los dígitos de la etiqueta destino, simplemente reenviará el paquete por el enlace que tiene con ese nodo.

Un nodo, al comparar su etiqueta con la etiqueta destino, sabrá hacia que parte del árbol lógico se enviará el paquete, es decir, si la etiqueta destino contiene a su propia etiqueta más otro sufijo, sabrá que la mayor coincidencia es hacia los nodos que él tiene como hijos en el árbol inducido. En cualquier otro caso, el nodo reenviará el paquete hacia el nodo que el considera como su padre para seguir buscando un nodo con el cual tenga una mayor coincidencia.

En caso de no existir un nodo con la etiqueta que se tiene como destino, el paquete de datos terminará su recorrido en el nodo que tenga la etiqueta más cercana a la etiqueta destino. Debido a que cada nodo sólo requiere la información de su *1-vecindario* para realizar el enrutamiento se puede apreciar que el tamaño de las tablas de enrutamiento es compacto y que será de tamaño d, donde d es el grado del nodo.

Enrutamiento en etiquetado doble

Básicamente la manera de enrutar los paquetes en este modelo es igual que en el anterior, con la diferencia que cada nodo contará con dos etiquetas, una correspondiente a cada raíz que inició el etiquetado. El uso de dos etiquetados para la red les permite a los nodos el poder elegir entre rutas alternas, pudiendo ser éstas de menor distancia y realizar la entrega del paquete de manera más rápida o elegir rutas más largas para poder así evitar rutas con una probabilidad de congestión alta.

Cada nodo al recibir un paquete podrá calcular sobre qué etiquetado está más cerca del nodo destino, esto se realiza conociendo ambas etiquetas de su *1-vecindario* y las dos etiquetas con las que él mismo cuenta. Al tener esta información se realizará un cálculo de distancias para poder elegir sobre qué árbol inducido se moverá el paquete de datos para alcanzar su destino. Al haber elegido el etiquetado en el que está más cerca del nodo al cual se dirige el paquete, comparará las etiquetas de sus nodos vecinos para buscar la mayor coincidencia con la etiqueta destino siguiendo la lógica que dice que si su etiqueta está contenida en la etiqueta destino, el paquete será reenviado hacia el hijo con el cual la etiqueta coincida más y en casos contrarios se enviará hacia el nodo considerado como padre en el etiquetado que haya decidido utilizar para el reenvío. Una vez que se ha comparado la etiqueta destino con los nodos vecinos reenviará el paquete por el enlace con el vecino con mayor coincidencia en las etiquetas.

Una de las ventajas que se obtiene al tener dos etiquetados, como bien se ha dicho, es la generación de rutas alternas a todos los nodos de la red. En caso de que un nodo falle y dentro de un etiquetado tenga muchos hijos, el subárbol debajo de ese nodo sigue siendo alcanzable por los demás nodos por medio del otro etiquetado con el que se cuenta. Gracias a ésto, el problema causado por la *tormenta de re-etiquetado* se mitiga, ya que no es una tarea que se

necesita realizar lo más rápido posible y en un período de tiempo muy corto. Aunque si es necesario reconstruir la sección del árbol que se ha perdido, no presenta la misma urgencia que cuando sólo hay un etiquetado.

Teorema 7. *El enrutamiento basado en prefijos es correcto dentro de la red.*

Demostración. Gracias a la definición de las etiquetas basadas en prefijos, a que cada nodo tiene una etiqueta única en cada etiquetado, y el ábol lógico es libre de ciclos al construirse. Considere un nodo fuente s y un nodo destino d. Debido a que ellos pertenecen al árbol lógico inducido en la red, deben satisfacer la relación padre-hijo que se forma dentro del árbol y cada nodo que es elegido como siguiente salto hacia d iniciando en s debe satisfacer también la misma relación. Debido a que el árbol lógico es finito y no cambian, debe existir al menos un camino libre de ciclos desde s hasta d en el árbol lógico. □

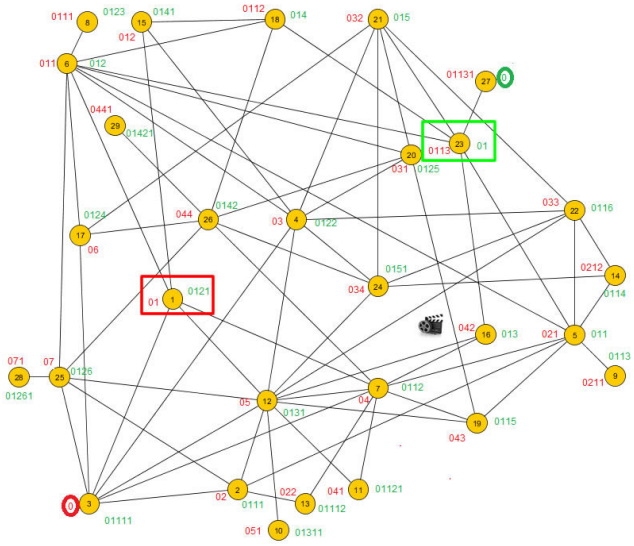
Teorema 8. *La tabla de enrutamiento son de tamaño $O(d)$, donde d es el grado máximo de cualquier nodo.*

Demostración. Ya que el enrutamiento se realiza pasando paquetes de datos de un nodo a otro nodo en su *1-vecindario* comparando la etiqueta destino con la propia y la de sus vecinos, cada nodo sólo necesita conocer las etiquetas de los nodos en su *1-vecindario*. □

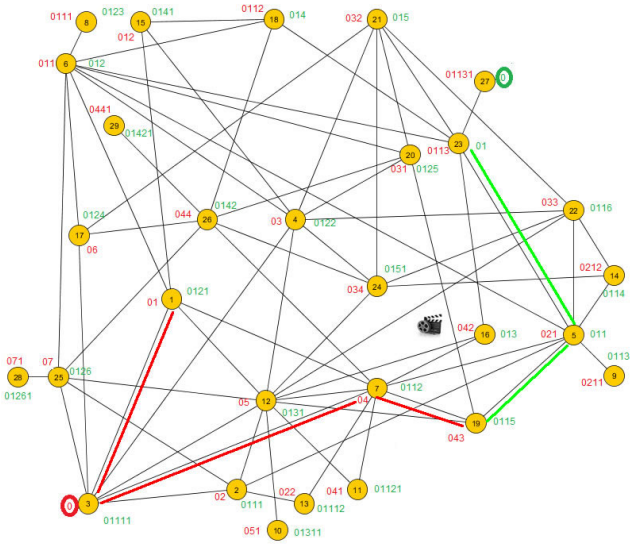
Publicación de contenido

Debido a que este esquema de enrutamiento permite el enrutamiento orientado a contenido, a continuación se describe el proceso que se lleva para incrustar los bloques de información en la red. El primer paso consiste en aplicar una función *hash* de dispersión que sea determinista al identificador del contenido que se desea montar en la red. Dicha función *hash* es conocida por todos los nodos dentro de la red y la función se encarga de mapear los identificadores del contenido al espacio de las etiquetas. La etiqueta obtenida de aplicar la función *hash* al identificador del contenido, será la etiqueta del nodo que conocerá la verdadera ubicación del contenido. Cuando un nodo desee consumir dicho contenido, aplicará él mismo la función *hash* al identificador del contenido y obtendrá la etiqueta del nodo ancla el cual almacenará las dos etiquetas correspondientes a cada uno de los etiquetados del nodo que almacena el contenido. Cuando el nodo consumidor obtiene como respuesta del nodo ancla las dos etiquetas del nodo destino, simplemente elegirá moverse por el etiquetado por el cual le genere una ruta más corta.

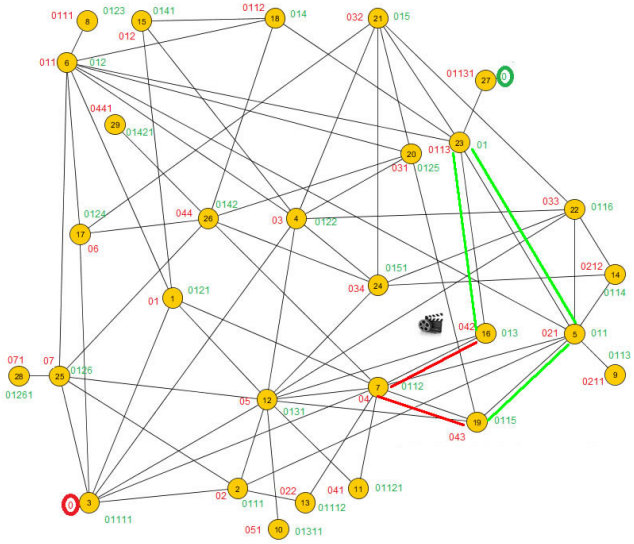
En la Figura 4.10 se muestra un ejemplo de publicación y obtención de contenido dentro de una red con doble etiquetado. En 4.10 (a) se muestra como el nodo 16 desea publicar contenido, el contenido en este ejemplo es un vídeo llamado "https://www.youtube.com/watch?v=BPidLpADlaM". A continuación el nodo 16 debe aplicar la función *hash* que todos los nodos en la red conocen al identificador del contenido, en este caso es al nombre del vídeo, la función *hash* da como resultado la etiqueta 01, la cual será la etiqueta de los nodos dentro de cada etiquetado que conocerán la verdadera ubicación del contenido. En 4.10 (b) El nodo 19 es el que ahora desea consumir dicho contenido por lo que aplica la misma función *hash* que el nodo anterior para poder obtener las etiquetas de los nodos que conocen la ubicación del contenido deseado, por lo tanto el también obtiene la etiqueta 01. El nodo 19 les hace la pregunta a ambos nodos cuyas etiquetas corresponden a 01 acerca de la etiqueta correspondiente dentro de cada etiquetado del nodo que contiene el vídeo que se desea ver y recibe como respuesta en el etiquetado verde la etiqueta 013 y en el rojo la etiqueta 042. En 4.10(c) se aprecia como se calcula da distancia en cada etiquetado desde el nodo consumidor para poder decidir la ruta más corta, que en este ejemplo fue dentro del etiquetado rojo. En 4.10 (d) simplemente se ejemplifica que el nodo consumidor obtiene el contenido deseado.



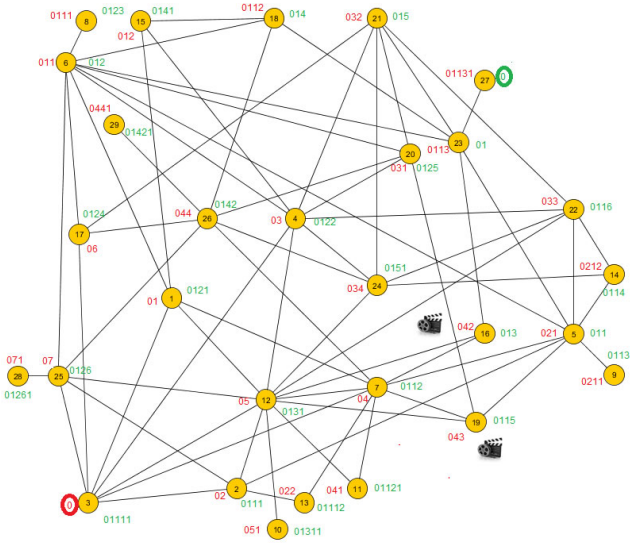
(a) Se incrusta el contenido en la red y se les avisa a los nodos ancla.



(b) El nodo 19 desea consumir el contenido y pregunta por la ubicación del contenido.



(c) Calcula las distancias en ambos etiquetados para decidir la ruta más corta



(d) El nodo por fin consume el contenido deseado

Figura 4.10: Publicación y obtención de contenido.

4.3. Resumen del capítulo

En este capítulo se presentaron las 3 etapas de la propuesta de solución: la primera es la selección de dos nodos raíz, ésta se realiza con base en 3 diferentes criterios y ésta además es la principal aportación del presente trabajo. Después se describe al etiquetado de la red el cual se basa en la manera en que PROSE[34] aetiqueta a la red y finalmente se describe la tercera etapa que consta del enrtuamiento, que igual que la etapa anterior se basa en PROSE. En el siguiente capítulo se exhibirán los resultados obtenidos durante la experimentación.

Capítulo 5

Pruebas y resultados

En el presente capítulo se presentan los resultados obtenidos al realizar la experimentación para la evaluación del comportamiento del algoritmo de enrutamiento compacto usando etiquetas basadas en prefijos tanto para el esquema base como para la propuesta de solución que se planteó.

Debido a que se trabajó con temas de teoría de grafos y algoritmos sobre ellos fue necesario la utilización de una herramienta para generar dichos grafos. El generador elegido fue *aSHIIP: A random topology generator of interdomain*[38]. aSHIIP es la más reciente versión del generador de topologías aleatorias que ha sido desarrollado en Supélec, una de las principales escuelas de ingeniería Francesa. Este generador permite la creación de redes de inter-dominio utilizando diferentes modelos de generación. El tamaño de las redes creadas mediante aSHIIP puede alcanzar tamaños mucho mayores a los que las redes en la vida real pueden tener. Las ventajas de este generador sobre BRITE[25], otro generador de topologías aleatorias, es que aSHIIP provee una jerarquía en los grafos generados, el grado de los nodos en los grafos generados respeta la leyes de las potencias que se ha observado a lo largo de los años como reporta [8] y que las topologías generadas por aSHIIP sí son conectadas.

Las pruebas se realizaron sobre 40 grafos generados por aSHIIP. De los 40 grafos, 10 fueron generados con 50 nodos, 10 grafos fueron de 75 nodos, 10 fueron con 100 nodos y los últimos 10 cuentan con 200 nodos. Los grafos generados se utilizaron para representar redes fijas sin cambios topológicos dinámicos durante la ejecución del algoritmo. Estas topologías son lo más parecidas a la arquitectura del Internet actual gracias a las características mencionadas anteriormente con las que cuenta el generador. De acuerdo a la información en [38], el modelo para generar dichos grafos fue GLP (por sus siglas en inglés *General Linear Preference*) cuyo funcionamiento se describe en [2]. El parámetro p estuvo establecido en 0.50 y éste indica la probabilidad de que un nodo aumente su grado y el parámetro responsable de la elección de un nodo para agregar un enlace nuevo debió estar en 0.75. Se consideró que el tamaño de la muestra seleccionada fue estadísticamente representativa para los experimentos que se realizaron.

Se realizaron las pruebas sobre 40 grafos diferentes debido a que se busca caracterizar el comportamiento del algoritmo propuesto en diferentes topologías fijas sin cambios durante la ejecución del algoritmo. Para lograr observar la manera en que se desempeña el algoritmo se consideró como mejor opción el uso de grafos diferentes, ya que si se utilizaba un grafo de menor cantidad de nodos como base para después crecer la cantidad de nodos se presentaría cierto comportamiento repetitivo el cual no nos ayudaría a conocer la manera en que el algoritmo se comporta.

Para la realización de los experimentos el primer paso fue establecer los parámetros en los valores indicados por el artículo del generador de topologías para obtener las topologías lo más parecidas a la arquitectura del Internet actual. En la Figura 5.1 se aprecia la interfaz en la cual se establecen los parámetros del generador y además se muestra que el modelo de generación será GLP, que se desea una topología con 200 nodos y los parámetros p y beta en 0,55 y 0,75 respectivamente.

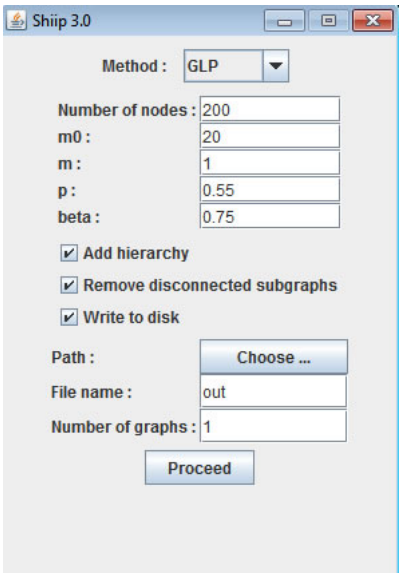


Figura 5.1: Generador con parámetros establecidos.

Al haber introducido los valores deseados, el generador da como salida un documento de texto plano con la información de la topología que se ha generado. El nombre del archivo por defecto es *out1.txt*. Dicho documento posteriormente es tratado para obtener la información del grafo y pasarla una matriz de adyacencia, la cual sirve como entrada al algoritmo Floyd-Warshall y como punto de partida de diversos algoritmos que se ejecutan sobre el grafo para su análisis.

Las métricas a considerar en los experimentos realizados fueron la longitud de las rutas encontradas por el algoritmo, la cantidad de caminos que pasan por los nodos y finalmente el *stretch* obtenido con cada criterio de selección de nodos. Al tomar en cuenta el tamaño de las rutas encontradas por el algoritmo es posible hacer una comparación contra las longitudes de las rutas obtenidas con el algoritmo de Dijkstra, el cual entrega las rutas más cortas entre cualquier par de nodos. Con esta información fue posible hacer el análisis del *stretch*. El número de caminos que atraviesan a cada nodo se consideró para poder observar si al tener dos alternativas de rutas, hacia todos los nodos, el tráfico generado dejaba de concentrarse en los nodos raíz

5.1. Tamaño de las rutas encontradas

En esta sección del capítulo, se hablará acerca de la longitud de las rutas obtenidas mediante el algoritmo de Dijkstra, que fue el algoritmo seleccionado para calcular las rutas más cortas entre todos los nodos y así poder comparar el desempeño del algoritmo propuesto. La métrica está dada en número de saltos.

5.1.1. Un sólo nodo raíz

A continuación se exhibirán los resultados que se obtuvieron con relación a la primera métrica considerada.

50 Nodos

	1 CENTRO									
50 Nodos	Dijkstra	σ	Grado mayor	σ	Centro	σ	Centro prom	σ	Arbitrario	σ
Grafo 1	3.0757	1.0166	3.9133	1.3575	3.682	1.2089	3.682	1.2089	4.3146	1.621
Grafo 2	2.7823	0.90778	3.517	1.2041	3.7704	1.2041	3.517	1.2041	4.8163	2.0145
Grafo 3	3.0876	1.0331	3.6395	1.1151	3.6395	1.1151	3.6395	1.1151	3.8146	1.2788
Grafo 4	2.5434	0.84238	3.0867	0.97856	3.0867	0.97856	3.0867	0.97856	3.9575	1.3677
Grafo 5	2.7109	0.87796	3.1752	0.97715	3.716	1.218	3.1752	0.97715	3.4235	1.1644
Grafo 6	2.6709	0.85733	3.2789	1.1715	3.3776	1.1844	3.2789	1.1715	3.5816	1.4334
Grafo 7	3.1335	1.0535	3.6565	1.2257	4.2976	1.6084	3.6565	1.2257	4.4133	1.7058
Grafo 8	2.7993	0.91728	3.7126	1.2865	3.6105	1.169	3.381	1.0572	3.9388	1.3425
Grafo 9	2.898	0.97359	3.6378	1.1985	3.8673	1.232	3.6378	1.1985	3.9269	1.4129
Grafo 10	2.5901	0.78953	2.9575	0.95422	3.5952	1.1855	2.9575	0.95422	3.1497	1.1842
Proms	**2.82917**	**0.926905**	**3.4575**	**1.146883**	**3.66428**	**1.210396**	**3.40121**	**1.109093**	**3.93368**	**1.45252**

Tabla 5.1: Comparación de tamaños de rutas para redes de 50 nodos.

En la Tabla 5.1 se muestran los promedio de las distancias mínimas obtenidas en los grafos de 50 nodos. Estos promedios son calculados para todos los posibles pares de nodos origen y destino en la red. La primer columna corresponde al promedio por grafo de las distancias obtenidas por el algoritmo de Dijkstra, el cuál representa el valor óptimo debido a que Dijkstra garantiza el descubrimiento de las rutas más cortas. Lo anterior nos servirá para determinar qué tan alejadas o cercanas se encuentran las longitudes de las rutas encontradas por el algoritmo propuesto. La tercera columna corresponde al promedio de las distancias obtenidas por el algoritmo que se está proponiendo, cuando el nodo con mayor grado es utilizado como nodo raíz. En la quinta columna se muestran los promedios utilizando al centro de la red como nodo raíz, mientras que en la séptima aparecen los promedios de las distancias al considerar como nodo raíz al nodo catalogado como centro promedio de la red. Finalmente en la novena columna se exponen los resultados al tener como nodo raíz a un nodo arbitrario. Después de cada columna con los promedios asociados a cada criterio de selección de un nodo raíz se tiene una columna con los valores de las desviaciones estándar correspondientes a cada criterio de selección de raíz dentro de cada grafo. El último renglón de la tabla corresponde al promedio final de los promedios por cada columna, tanto para los valores de los criterios como para los valores de las desviaciones estándar. Con base en los resultados obtenidos y mostrados, la mejor estrategia de selección de un nodo raíz para 50 nodos fue la del centro promedio ya que su promedio fue el menor de todos los criterios.

75 Nodos

	1 CENTRO									
75 Nodos	Dijkstra	σ	Grado mayor	σ	Centro	σ	Centro prom	σ	Arbitrario	σ
Grafo 1	2.9448	0.91546	3.4598	1.0578	3.4598	1.0578	3.4598	1.0578	4.5372	1.6135
Grafo 2	2.7986	0.84862	3.3525	0.99712	3.8449	1.1598	3.3525	0.99712	4.197	1.4349
Grafo 3	2.896	0.87173	3.7512	1.1354	3.9974	1.227	3.7512	1.1354	3.9974	1.227
Grafo 4	2.8478	0.89037	3.3854	1.0597	3.8053	1.1759	3.3854	1.0597	4.0733	1.2593
Grafo 5	2.8852	0.87102	3.4769	1.0467	3.7745	1.2036	3.4769	1.0467	3.5231	1.0856
Grafo 6	2.833	0.85498	3.7208	1.1683	3.8656	1.0873	3.6727	1.0761	4.1984	1.4039
Grafo 7	2.9582	0.88533	3.3787	0.94233	3.3787	0.94233	3.3787	0.94233	3.555	1.1064
Grafo 8	2.8682	0.8834	3.3058	0.98796	3.6868	1.0789	3.3058	0.98796	3.6868	1.0789
Grafo 9	2.8386	0.89972	3.4387	1.1051	4.0885	1.3132	3.4387	1.1051	4.2869	1.4062
Grafo 10	2.8723	0.8779	3.3717	1.0101	3.9052	1.1028	3.3717	1.0101	4.2029	1.3189
Proms	**2.87427**	**0.879853**	**3.46415**	**1.051051**	**3.78067**	**1.134863**	**3.45934**	**1.041831**	**4.0258**	**1.29346**

Tabla 5.2: Comparación de tamaños de rutas para redes de 75 nodos.

En la Tabla 5.2, se muestra la información de la misma manera en que está exhibida en la Tabla 5.1, pero ahora para los grafos generados con 75 nodos en ellos. Al igual que en el caso de los grafos con 50 nodos, al haber 75 nodos, la mejor estrategia fue seleccionar al centro promedio como nodo raíz ya que obtuvo menores longitudes de rutas y eso se refleja en un menor promedio final.

100 Nodos

	1 CENTRO									
100 Nodos	Dijkstra	σ	Grado mayor	σ	Centro	σ	Centro prom	σ	Arbitrario	σ
Grafo 1	2.9814	0.90819	4.0284	1.2419	4.2602	1.2678	3.9254	1.1351	4.2705	1.3757
Grafo 2	2.8409	0.81013	3.6005	1.073	3.8058	1.0631	3.5358	0.97606	4.0078	1.2015
Grafo 3	2.8928	0.85774	3.363	0.98585	3.363	0.98585	3.363	0.98585	4.0878	1.4112
Grafo 4	3.0577	0.907	3.8116	1.135	3.8116	1.135	3.8116	1.135	4.1538	1.3325
Grafo 5	2.9068	0.87484	3.5502	1.049	3.5502	1.049	3.5502	1.049	4.0078	1.2513
Grafo 6	2.9365	0.86256	3.8701	1.1792	3.7782	1.0443	3.7411	1.0669	4.6085	1.481
Grafo 7	2.8907	0.84048	3.4368	1.0639	3.871	1.1434	3.4368	1.0639	4.0078	1.3811
Grafo 8	2.9522	0.88316	3.8112	1.1079	3.8112	1.1079	3.8112	1.1079	4.3335	1.3426
Grafo 9	3.0148	0.89913	3.9278	1.1344	4.3274	1.3031	3.9278	1.1344	4.4519	1.2838
Grafo 10	2.9514	0.87867	3.9114	1.1354	4.0161	1.1356	3.8326	1.0454	4.301	1.3451
Proms	**2.94252**	**0.87219**	**3.7311**	**1.110555**	**3.85947**	**1.123505**	**3.69355**	**1.069951**	**4.22304**	**1.34058**

Tabla 5.3: Comparación de tamaños de rutas para redes de 100 nodos.

En la Tabla 5.3, al igual que en las dos tablas anteriores, se puede observar el mismo tipo de distribución de la información proporcionada pero ahora para el caso de los grafos generados con 100 nodos en ellos. Para los grafos con 100 nodos la estrategia que dio como resultados menores longitudes de ruta fue la del centro promedio.

200 Nodos

	1 CENTRO									
200 Nodos	Dijkstra	σ	Grado mayor	σ	Centro	σ	Centro prom	σ	Arbitrario	σ
Grafo 1	3.1903	0.88926	4.104	1.1288	4.5268	1.2371	4.1443	1.1311	4.7935	1.4288
Grafo 2	3.1723	0.84974	3.897	1.0686	4.3403	1.1591	3.897	1.0686	4.1429	1.1373
Grafo 3	3.129	0.84955	3.9244	1.0441	4.2425	1.1558	3.9244	1.0441	4.3901	1.275
Grafo 4	3.1165	0.8478	3.8835	1.0611	3.8402	0.94801	3.8402	0.94801	4.1372	1.1206
Grafo 5	3.2093	0.8777	3.7171	0.9733	3.7171	0.9733	3.7171	0.9733	3.7493	0.99463
Grafo 6	3.1501	0.85653	3.8107	1.0437	4.5656	1.2558	3.8107	1.0437	4.2669	1.209
Grafo 7	3.2272	0.87404	3.8183	1.0677	4.1916	1.1823	3.8183	1.0677	4.3084	1.3085
Grafo 8	3.1249	0.85125	3.8538	1.0484	3.8538	1.0484	3.8538	1.0484	3.8538	1.0484
Grafo 9	3.0616	0.82812	3.6106	1.0271	4.2161	1.2166	3.6106	1.0271	4.7996	1.5207
Grafo 10	3.2672	0.89098	4.2699	1.2286	4.6199	1.2724	4.0847	1.0404	4.2699	1.2286
Proms	**3.16484**	**0.861497**	**3.88893**	**1.06914**	**4.21139**	**1.144881**	**3.87011**	**1.039241**	**4.27116**	**1.227153**

Tabla 5.4: Comparación de tamaños de rutas para redes de 200 nodos.

En la Tabla 5.4 observamos la misma disposición de la información contenida. También se puede ver una vez más que la estrategia de elección de un nodo raíz que mejores resultados arrojó fue la de elegir al nodo centro promedio como nodo raíz.

Como se pudo observar en las cuatro tablas anteriores, la estrategia que arrojó como resultado las longitudes de rutas más pequeñas en los cuatro diferentes tamaños de las topologías generadas fue la de elegir al centro promedio como el nodo raíz. Este resultado era esperado desde el momento en que se diseñaron los algoritmos de selección de centro y gracias a los experimentos se logró observar que se cumplió lo esperado.

5.1.2. Dos nodos raíz

A continuación se muestran los resultados arrojados al elegir dos nodos raíz para iniciar los etiquetados dentro de las redes con 50, 75, 100 y 200 nodos con los 3 diferentes criterios de selección de raíces. En las tablas se encuentra la información obtenida de los experimentos para calcular las distancias más cortas teniendo en las redes dos nodos raíz. Partiendo de cualquier nodo se calcula la distancia hacia todos los destinos en ambos etiquetados, la ruta que cuente con menor cantidad de saltos entre ambos etiquetados es la ruta que se elige.

50 Nodos

	2 CENTROS					
	Centro y Lejano					
50 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	3.682	1.2089	4.0748	1.4902	3.3665	1.1422
Grafo 2	3.517	1.2041	4.1173	1.4771	3.1267	1.0372
Grafo 3	3.6395	1.1151	4.3282	1.6131	3.3869	1.1036
Grafo 4	3.0867	0.97856	3.6224	1.1758	2.8622	0.93065
Grafo 5	3.1752	0.97715	3.4235	1.1644	2.9668	0.92883
Grafo 6	3.2789	1.1715	3.6259	1.2118	2.8724	0.91366
Grafo 7	3.6565	1.2257	4.9541	1.9191	3.3495	1.1091
Grafo 8	3.381	1.0572	3.9218	1.3454	3.1276	1.0193
Grafo 9	3.6378	1.1985	4.1803	1.488	3.273	1.1476
Grafo 10	2.9575	0.95422	3.6803	1.2709	2.6981	0.80943
Proms	**3.40121**	**1.109093**	**3.99286**	**1.41558**	**3.10297**	**1.014157**
Más Separados						
50 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.5697	1.8261	4.5221	1.7454	3.6293	1.3113
Grafo 2	3.9745	1.4061	4.1173	1.4771	3.2653	1.152
Grafo 3	5.4269	2.5119	4.3776	1.6739	3.7815	1.5105
Grafo 4	4.0068	1.5563	3.6224	1.1758	3.1122	1.0842
Grafo 5	3.9201	1.45	3.4235	1.1644	3.0255	0.96281
Grafo 6	3.6259	1.2118	3.2789	1.1715	2.8724	0.91366
Grafo 7	4.0748	1.5352	4.3759	1.5904	3.5374	1.2704
Grafo 8	3.5884	1.1294	3.7126	1.2865	3.0986	0.98868
Grafo 9	4.2636	1.721	4.3384	1.5692	3.5391	1.3067
Grafo 10	2.9575	0.95422	3.6803	1.2709	2.6981	0.80943
Proms	**4.04082**	**1.530202**	**3.9449**	**1.41251**	**3.25594**	**1.130968**
Arbitrarios						
50 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.017	1.4273	4.2228	1.5715	3.5026	1.2363
Grafo 2	4.1173	1.4771	3.6888	1.3426	3.1998	1.1261
Grafo 3	3.9626	1.374	4.5578	1.6503	3.5281	1.236
Grafo 4	3.8963	1.3232	3.3231	1.0905	3.0264	1.0381
Grafo 5	3.4235	1.1644	3.5986	1.2241	3.0816	1.0535
Grafo 6	3.4014	1.259	3.6037	1.2701	2.932	0.97917
Grafo 7	3.949	1.456	4.199	1.5408	3.6267	1.3174
Grafo 8	3.8759	1.3406	4.1327	1.4212	3.4311	1.2086
Grafo 9	4.2636	1.721	3.7279	1.2462	3.3886	1.1674
Grafo 10	3.5068	1.1001	3.4592	1.1829	3.0502	1.0021
Proms	**3.84134**	**1.36427**	**3.85136**	**1.35402**	**3.27671**	**1.136467**

Tabla 5.5: Comparación de tamaños de rutas para redes de 50 nodos con dos nodos raíz.

La Tabla 5.5 se encuentra dividida en 3 secciones las cuales corresponden a cada uno de los criterios para la elección de dos nodos raíz. El primer criterio es elegir al centro promedio de la red y al nodo más lejano a él. En la primer columna se tienen los promedios de las rutas más cortas hacia todos los destinos dentro del etiquetado realizado por el nodo centro promedio, en la tercer columna está este mismo promedio pero para el caso del nodo más alejado al primer nodo y en la quinta columna se muestra el promedio habiendo elegido a la ruta con menos

saltos de entre ambos etiquetados. En la segunda sección se muestran los promedios de haber elegido a los nodos más separados entre sí dentro de la red, la primer columna corresponde a los promedios obtenidos al etiquetar la red de acuerdo con el primer nodo seleccionado y en la tercera se encuentran los promedios del etiquetado comenzado desde el nodo más alejado al primer nodo, en la quinta columna se encuentra el promedio al haber elegido a la ruta más corta entre ambos etiquetados. La tercera sección corresponde a elegir a dos nodos arbitrarios y en las columnas 1 y 3 se muestran los promedios de las rutas obtenidas dentro de cada uno de los etiquetados, en la quinta columna se muestra el promedio de la elección de la ruta más corta entre ambos etiquetados. También se muestra el valor de la desviación estándar para cada promedio obtenido de cada etiquetado, este valor se observa después de cada columna con promedios de rutas cortas dentro de cada sección. El último renglón de cada sección muestra el promedio final de cada criterio obtenido de los promedios para cada grafo.

75 Nodos

	2 CENTROS					
	Centro y Lejano					
75 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	3.4598	1.0578	3.8941	1.1804	3.1744	0.96894
Grafo 2	3.3525	0.99712	3.9104	1.1855	3.0748	0.91102
Grafo 3	3.7512	1.1354	4.7308	1.5946	3.4272	1.088
Grafo 4	3.3854	1.0597	4.0133	1.3327	3.0766	1.2271
Grafo 5	3.4769	1.0467	4.9522	1.7082	3.2895	0.99435
Grafo 6	3.6727	1.0761	3.943	1.2495	3.2395	0.97068
Grafo 7	3.3787	0.94233	4.4654	1.5568	3.2295	0.91968
Grafo 8	3.3058	0.98796	4.4121	1.4255	3.1385	0.95379
Grafo 9	3.4387	1.1051	4.0885	1.3132	3.1496	0.98987
Grafo 10	3.3717	1.0101	4.7297	1.6969	3.221	0.98902
Proms	**3.45934**	**1.041831**	**4.31395**	**1.42433**	**3.20206**	**1.001245**
Más Separados						
75 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	3.8519	1.302	4.214	1.3846	3.4195	1.1284
Grafo 2	4.197	1.4349	3.987	1.2632	3.3862	1.0979
Grafo 3	4.2814	1.3507	4.1081	1.2939	3.5957	1.1522
Grafo 4	4.2144	1.5081	4.0133	1.3327	3.398	1.1733
Grafo 5	3.6001	1.1142	3.8993	1.2544	3.2558	1.0206
Grafo 6	4.2088	1.475	3.8708	1.1975	3.4136	1.0919
Grafo 7	4.3595	1.7592	4.4372	1.5621	3.6609	1.292
Grafo 8	4.3613	1.4661	3.9241	1.3975	3.4313	1.1827
Grafo 9	4.0885	1.3132	4.5724	1.6585	3.6324	1.243
Grafo 10	3.9345	1.2628	4.2843	1.4404	3.4395	1.1114
Proms	**4.10974**	**1.39862**	**4.13105**	**1.37848**	**3.46329**	**1.14934**
Arbitrarios						
75 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.5113	1.6096	3.7431	1.2657	3.3277	1.1101
Grafo 2	3.9156	1.294	3.6909	1.1107	3.2558	1.0432
Grafo 3	4.3384	1.394	4.5039	1.5153	3.8245	3.8245
Grafo 4	3.9759	1.3165	4.1355	1.3966	3.4906	1.206
Grafo 5	3.8563	1.2754	4.3713	1.5593	3.5046	1.1843
Grafo 6	4.1355	1.2763	3.8708	1.1975	3.5531	1.1404
Grafo 7	3.555	1.1064	3.4306	1.0089	3.2162	0.96143
Grafo 8	3.6227	1.1883	3.3576	1.01	3.1185	0.97032
Grafo 9	4.2488	1.4372	3.7419	1.2161	3.4198	1.1405
Grafo 10	3.8375	1.215	3.7608	1.169	3.4143	1.0512
Proms	**3.9997**	**1.31127**	**3.86064**	**1.24491**	**3.41251**	**1.363195**

Tabla 5.6: Comparación de tamaños de rutas para grafos de 75 nodos con dos nodos raíz.

En la Tabla 5.6 se muestra la misma información obtenida que en la tabla anterior, sólo que en este caso es para los grafos con 75 nodos. En ambas tablas se observa que el mejor resultado se obtiene eligiendo al nodo centro promedio y al nodo más alejado de él.

100 Nodos

En la Tabla 5.7 se muestra la información recabada en los experimentos a los grafos con 100 nodos. La información está desplegada de la misma manera que en las pasadas dos tablas. La elección del nodo centro promedio y el nodo más alejado a él da como resultado un promedio final menor en cuanto a la longitud de las rutas a comparación de los demás criterios.

	2 CENTROS					
	Centro y Lejano					
100 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.2602	1.2678	4.2688	1.3973	3.5741	1.063
Grafo 2	3.5358	0.97606	4.0977	1.179	3.3616	0.97956
Grafo 3	3.363	0.98585	4.2053	1.2072	3.1886	0.96329
Grafo 4	3.8116	1.135	4.6234	1.5722	3.4725	1.0545
Grafo 5	3.5502	1.049	4.0078	1.2513	3.3039	0.99256
Grafo 6	3.7411	1.0669	4.4032	1.328	3.4381	1.0113
Grafo 7	3.4368	1.0639	4.2667	1.4147	3.1241	0.92762
Grafo 8	3.8112	1.1079	4.2478	1.2279	3.4587	1.0271
Grafo 9	3.9278	1.1344	4.5088	1.3547	3.5541	1.0632
Grafo 10	3.8326	1.0454	4.4308	1.3786	3.5347	1.0455
Proms	**3.72703**	**1.083221**	**4.30603**	**1.33109**	**3.40104**	**1.012763**
Más Separados						
100 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.9726	1.2678	4.2494	1.3973	3.7967	1.2652
Grafo 2	5.0699	1.7942	3.6005	1.073	3.3158	0.9994
Grafo 3	3.913	1.1894	4.6485	1.6013	3.5296	1.0864
Grafo 4	4.6234	1.5722	4.2115	1.3696	3.6753	1.2253
Grafo 5	4.8266	1.7375	4.3875	1.3912	3.7328	1.2432
Grafo 6	4.4032	1.328	3.913	1.2428	3.5086	1.0761
Grafo 7	4.7664	1.7086	4.6122	1.5412	3.8905	1.3934
Grafo 8	4.8872	1.6469	4.3933	1.3872	3.8491	1.3063
Grafo 9	4.5088	1.3547	4.2985	1.329	3.7687	1.1983
Grafo 10	4.3319	1.3163	3.8351	1.0907	3.5438	1.0608
Proms	**4.6303**	**1.49156**	**4.21495**	**1.34233**	**3.66109**	**1.18544**
Arbitrarios						
100 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.0486	1.1584	4.2705	1.3757	3.6491	1.134
Grafo 2	3.8058	1.0631	4.5026	1.5654	3.5448	1.0486
Grafo 3	4.1332	1.2039	3.9514	1.34	3.4896	1.1361
Grafo 4	4.1266	1.3715	4.0437	1.2444	3.7444	1.1656
Grafo 5	3.9505	1.308	3.7225	1.1534	3.3346	1.0701
Grafo 6	4.0841	1.298	3.8701	1.1792	3.4599	1.047
Grafo 7	3.9905	1.277	4.1699	1.3257	3.4199	1.0931
Grafo 8	4.1839	1.2322	4.2284	1.2733	3.6236	1.1236
Grafo 9	4.263	1.2485	4.3327	1.3158	3.8174	1.2002
Grafo 10	4.4383	1.4627	4.1253	1.2477	3.7122	1.1236
Proms	**4.10245**	**1.26233**	**4.12171**	**1.30206**	**3.57955**	**1.11419**

Tabla 5.7: Comparación de tamaños de rutas para grafos de 100 nodos con dos nodos raíz.

200 Nodos

	2 CENTROS					
	Centro y Lejano					
200 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	4.1443	1.1311	5.308	1.6825	3.8423	1.0825
Grafo 2	3.897	1.0686	4.7427	1.3544	3.6829	1.0236
Grafo 3	3.9244	1.0441	4.0853	1.0793	3.522	0.93718
Grafo 4	3.8402	0.94801	4.4653	1.2935	3.6717	0.95989
Grafo 5	3.7171	0.9733	4.289	1.1792	3.4995	0.92263
Grafo 6	3.8107	1.0437	5.1156	1.8583	3.589	1.0151
Grafo 7	3.8183	1.0677	5.7035	1.7829	3.6449	1.0407
Grafo 8	3.8538	1.0484	5.2612	1.5347	3.6595	1.012
Grafo 9	3.6106	1.0271	4.931	1.5857	3.3697	0.95434
Grafo 10	4.0847	1.0404	5.0012	1.376	3.8589	1.0125
Proms	**3.87011**	**1.039241**	**4.89028**	**1.47265**	**3.63404**	**0.996044**
Más Separados						
200 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	5.308	1.6825	4.5979	1.3855	4.1277	1.2939
Grafo 2	4.4012	1.212	4.4122	1.3094	3.8954	1.1224
Grafo 3	4.0611	1.0436	4.3824	1.3536	3.6556	0.97955
Grafo 4	4.4135	1.1668	4.4653	1.2935	3.8993	1.0906
Grafo 5	3.7686	1.0074	4.289	1.1792	3.5242	0.94141
Grafo 6	4.9333	2.0958	4.5212	1.4095	3.8858	1.36
Grafo 7	5.3955	1.8374	4.3467	1.2692	3.9619	1.2211
Grafo 8	4.4588	1.4452	4.2335	1.2637	3.7422	1.1132
Grafo 9	5.1847	1.9058	4.3925	1.2896	3.8997	1.2364
Grafo 10	5.1269	1.5591	4.4064	1.246	4.0241	1.1846
Proms	**4.70516**	**1.49556**	**4.40471**	**1.29992**	**3.86159**	**1.154316**
Arbitrarios						
200 nodos	Et1	σ	Et2	σ	$\bar{X}$	σ
Grafo 1	5.3622	1.6466	4.104	1.1288	3.8486	1.1185
Grafo 2	4.4122	1.3094	4.3031	1.2501	3.8006	1.0994
Grafo 3	4.3745	1.2373	4.5974	1.357	3.9426	1.1496
Grafo 4	4.556	1.2718	4.5452	1.3871	4.2737	1.2996
Grafo 5	4.0583	1.1543	4.3122	1.3665	3.7543	1.0684
Grafo 6	3.8107	1.0437	4.526	1.351	3.6005	1.0308
Grafo 7	4.4234	1.2289	4.7335	1.5206	3.9414	1.1651
Grafo 8	4.0581	1.2005	4.9851	1.5856	3.8561	1.1437
Grafo 9	3.6106	1.0271	4.3181	1.1795	3.3867	0.94355
Grafo 10	4.7496	1.4018	4.5274	1.4034	4.1082	1.2712
Proms	**4.34156**	**1.25214**	**4.4952**	**1.35296**	**3.85127**	**1.128985**

Tabla 5.8: Comparación de tamaños de rutas para grafos de 200 nodos con dos nodos raíz.

La información que se muestra en la Tabla 5.8 es la información que se obtuvo durante los experimentos en los grafos con 200 nodos. La manera en que la información está mostrada concuerda con las tablas anteriores y al igual que en ellas, la mejor opción de selección de nodos raíz fue el nodo centro promedio y el nodo más alejado a éste.

Para cada grafo se obtuvo el promedio de la longitud de las rutas encontradas con cada uno de los criterios para la selección de uno y dos nodos raíz, además se calculó la longitud de las rutas con el algoritmo de Dijkstra para poder calcular el promedio de las longitudes más cortas en cada grafo. En cada grupo de grafos con la misma cantidad de nodos, se promedió cada uno de los resultados que se mostraron en las tablas que se presentaron anteriormente. Esos resultados son los que se plasman en la Figura 5.2 en la cual se aprecia que la mejor manera de obtener tamaños de ruta menores es eligiendo a dos nodos raíz, siendo éstos el nodo centro promedio y el más alejado a él. En cuestión de un sólo nodo raíz, se aprecia que el nodo centro promedio es la mejor estrategia. Ambos casos son los más cercanos a los valores obtenidos por

Dijkstra en cada uno de los esquemas que se analizaron. Con base a esos resultados se puede ver que si es buena idea utilizar los recursos de la red para elegir a los nodos que se desempeñarán como raíz. En la Tabla 5.9 se muestran los valores que sirvieron para la realización de la gráfica y que ayudaron a concluir acerca del mejor criterio de elección de un nodo raíz y de dos nodos raíz.

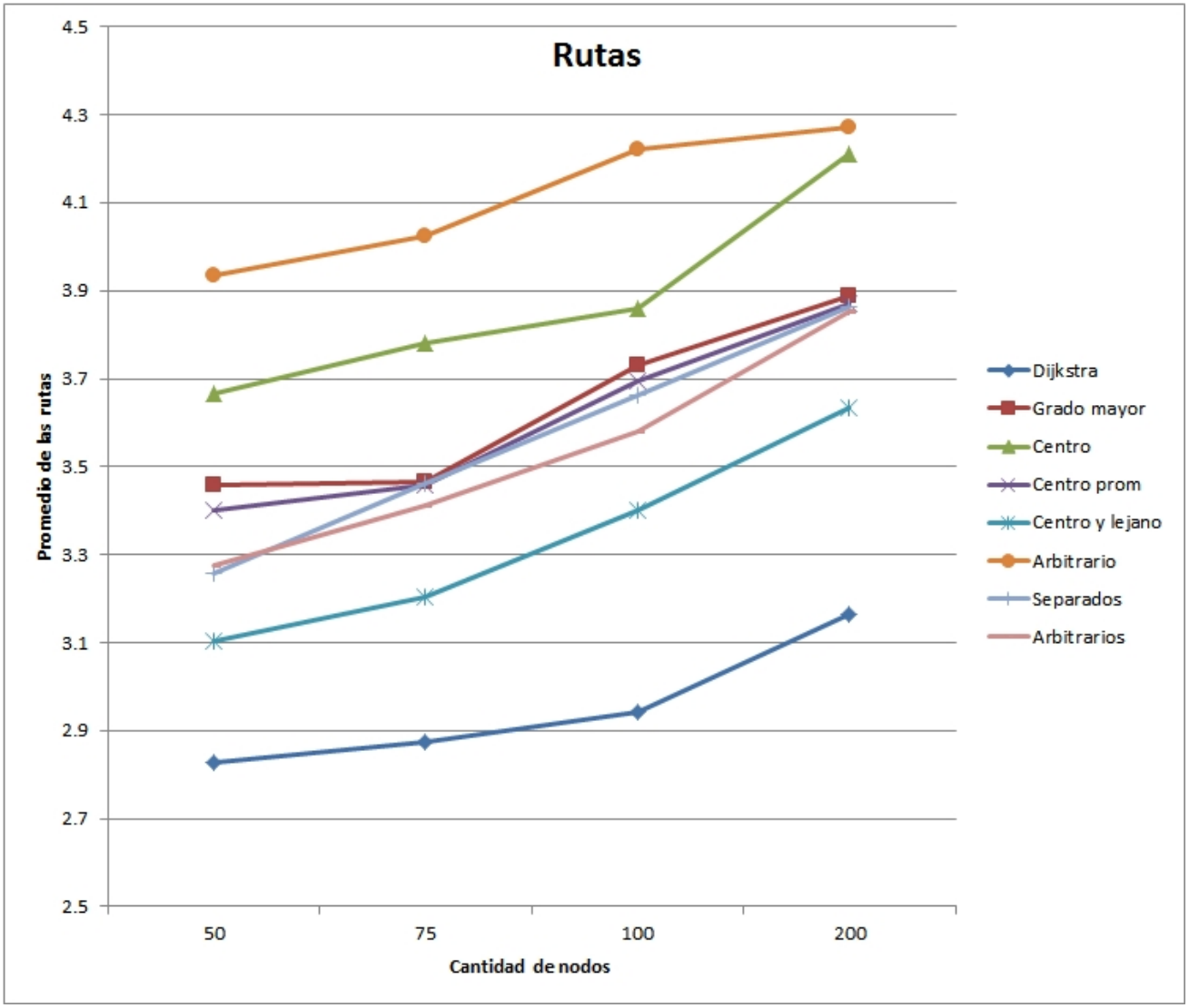


Figura 5.2: Comparación en los promedios de las rutas encontradas.

	1 CENTRO										2 CENTROS					
Nodos	Dijkstra	σ	Grado	σ	Centro	σ	Centro P	σ	Arb	σ	C y l	σ	Separados	σ	Arbs	σ
50	2.8292	0.9269	3.4575	1.1469	3.66428	1.2104	3.4012	1.1091	3.9337	1.4525	3.10297	1.0142	3.2559	1.131	3.2767	1.1365
75	2.8743	0.8799	3.4642	1.0511	3.78067	1.1349	3.4593	1.0418	4.0258	1.2935	3.20206	1.0012	3.4633	1.1493	3.4125	1.3632
100	2.9425	0.8722	3.7311	1.1106	3.85947	1.1235	3.6936	1.07	4.223	1.3406	3.40104	1.0128	3.6611	1.1854	3.5796	1.1142
200	3.1648	0.8615	3.8889	1.0691	4.21139	1.1449	3.8701	1.0392	4.2712	1.2272	3.63404	0.996	3.8616	1.1543	3.8513	1.129

Tabla 5.9: Resultados condensados de los experimentos en relación con las longitudes de las rutas.

5.2. Número de caminos que pasan sobre los nodos

La cantidad de caminos que atraviesan a un nodo fue la métrica decidida para poder ver si en realidad se logra la reducción en el tráfico que pasa por el nodo raíz al momento de realizar el enrutamiento sobre el árbol lógico. En esta parte se mostrarán los resultados de la medición de caminos encontrados en la red cuando hubo un nodo raíz y los diferentes criterios de seleccionarlo y al haber dos nodos raíz también habiendo elegido diferentes nodos para comenzar el etiquetado.

5.2.1. Un sólo nodo raíz

Aquí se podrá observar los resultados con respecto a la segunda métrica con la que se experimentó. En este caso es cuando sólo hay un nodo raíz. En las tablas que se presentarán a continuación se muestra la información de la siguiente manera. En la primera columna se encuentra el número máximo de caminos que pasan por el nodo indicado en la segunda columna, ambas columnas son con respecto a las rutas encontradas en los grafos con el algoritmo de Dijkstra. En la tercera columna se muestra el promedio de veces que los diferentes caminos que atraviesan a los nodos que componen la red. En la cuarta columna se presenta el número máximo de caminos que atraviesan al nodo señalado en la quinta columna y en la sexta columna se muestra el promedio de caminos que atraviesan a los nodos que componen la red. Lo anterior está calculado habiendo seleccionado al nodo con el grado mayor de la red para la inducción del árbol lógico. De la séptima a la novena columnas se muestra la misma información en el mismo orden sólo que en este caso el nodo raíz es el nodo considerado el centro de la red. Desde la décima columna a la doceava se encuentra la información correspondiente al tercer criterio de selección del nodo raíz, el centro promedio, y en las últimas tres columnas se exhibe la información relacionada a la elección arbitraria del nodo raíz. El penúltimo renglón de la tabla muestra el promedio total por cada criterio, calculado con los promedios del conjunto de grafos para cada criterio. En el último renglón se muestra la desviación estándar relacionado con el promedio total de cada criterio.

50 Nodos

	1 CENTRO														
	Grafo			Mayor Grado			Centro			Centro Promedio			Arbitrarios		
50 Nodos	#Máx	Nodo	$\overline{X}$	#Máx	Nodo	$\overline{X}$	#Máx	Nodo	$\overline{X}$	#Máx	Nodo	$\overline{X}$	#Máx	Nodo	$\overline{X}$
Grafo 1	605	5	99.6327	1842	5	139.837	1820	6	128.7347	1820	6	128.7347	1846	4	159.102
Grafo 2	765	11	85.551	2110	11	120.816	1174	4	132.9796	2110	11	120.8163	1598	36	183.1837
Grafo 3	629	5	100.2041	2000	5	126.694	2000	5	126.6939	2000	5	126.6939	2018	5	135.102
Grafo 4	955	1	74.0816	2150	1	100.163	2150	1	100.1633	2150	1	100.1633	1632	3	141.9592
Grafo 5	914	2	82.1224	2064	2	104.408	1576	4	130.3673	2064	2	104.4082	1862	7	116.3265
Grafo 6	1043	2	80.2041	2076	2	109.388	1682	2	114.1224	2076	2	109.3878	2014	2	123.9184
Grafo 7	960	15	102.4082	2078	15	127.51	1682	15	158.2857	2078	15	127.5102	1612	15	163.8367
Grafo 8	627	12	86.3673	1994	7	130.204	1732	5	125.3061	1886	12	114.2857	1646	39	141.0612
Grafo 9	668	16	91.102	2020	16	126.612	1856	2	137.6327	2020	16	126.6122	2028	16	140.4898
Grafo 10	1167	7	76.3265	2148	7	93.9592	1764	1	124.5714	2148	7	93.9592	2154	7	103.1837
Proms	**833.3**		**87.8**			**117.959**			**127.886**			**115.2572**			**140.816**
Desv			10.1929			15.0145			15.01485			12.723302			23.55827

Tabla 5.10: Nodo con mayor número de caminos que lo atraviesan en redes con 50 nodos.

La Tabla 5.10 muestra los resultados obtenidos sobre los grafos con 50 nodos y los criterios de selección de un sólo nodo raíz. Al igual que en la métrica pasada con respecto a un sólo nodo raíz, el criterio de elegir al centro promedio dio como resultado un menor promedio en los caminos de todos los nodos en el grafo.

75 Nodos

	1 CENTRO														
	Grafo			Mayor Grado			Centro			Centro Promedio			Arbitrarios		
75 Nodos	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$
Grafo 1	2328	5	141.973	4932	5	179.568	4932	5	179.5676	4932	5	179.5676	34900	3	258.2162
Grafo 2	1535	20	131.2973	4922	20	171.73	3954	1	207.6757	4922	20	171.7297	3606	6	233.3784
Grafo 3	896	19	138.4054	4778	19	200.838	4132	55	218.8108	4778	19	200.8378	4132	55	218.8108
Grafo 4	1737	10	134.8919	4956	10	174.135	4090	5	204.7838	4956	10	174.1351	4390	2	224.3514
Grafo 5	2251	1	137.6216	4822	1	180.811	4088	1	202.5405	4822	1	180.8108	4156	8	184.1892
Grafo 6	1070	8	133.8108	4708	17	198.622	4578	1	209.1892	4702	8	195.1081	3430	6	233.4865
Grafo 7	2669	2	142.9459	4904	2	173.649	4904	2	173.6486	4904	2	173.6486	3986	9	186.5135
Grafo 8	2091	10	136.3784	4932	10	168.324	4588	6	196.1351	4932	10	168.3243	4588	6	196.1351
Grafo 9	1886	19	134.2162	4944	19	178.027	3938	2	255.4595	4944	19	178.027	3680	4	239.9559
Grafo 10	1927	13	136.6757	4968	13	173.135	4610	3	212.0811	4968	13	173.1351	4158	3	233.8108
Proms	**1839**		**136.822**			**179.884**			**205.989**			**179.5324**			**220.885**
Desv			3.613321			11.1164			22.4013			10.496578			24.48613

Tabla 5.11: Nodo con mayor número de caminos que lo atraviesan en redes con 75 nodos.

En la Tabla 5.11 se presentan los resultados de los experimentos de la segunda métrica en los grafos que cuentan con 75 nodos. Aquí el criterio de elección del nodo raíz a partir del centro promedio arrojó al igual que en el caso anterior un menor promedio de caminos sobre los nodos del grafo, por esa razón se considera la mejor opción. La información fue presentada de la misma manera que la tabla anterior.

100 Nodos

Para los mostrar los resultados obtenidos en grafos con 100 nodos se utiliza la Tabla 5.12 que al igual que las anteriores cuenta con la misma estructura en cuanto a la presentación de la información. La elección del centro promedio como nodo raíz nos permitió observar que el promedio obtenido sobre los experimentos fue más bajo que a comparación de las otros criterios.

	1 CENTRO														
	Grafo			Mayor Grado			Centro			Centro Promedio			Arbitrarios		
100 Nodos	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$
Grafo 1	1734	9	194.1818	8526	9	296.7879	7138	1	319.5	8344	66	286.6869	6944	5	320.5051
Grafo 2	2397	9	180.404	8704	10	254.8485	8252	5	275	8850	9	248.5051	6918	77	294.7677
Grafo 3	3236	20	185.4949	9094	20	231.5758	9094	20	231.6	9094	20	231.5758	7426	20	302.6061
Grafo 4	2555	2	201.6566	8610	2	275.5354	8610	2	275.5	8610	2	275.5354	8198	7	309.0707
Grafo 5	3463	1	186.8687	8746	1	249.9192	8746	1	249.9	8746	1	249.9192	6886	4	294.7677
Grafo 6	1929	17	189.7778	8252	6	281.2727	8794	9	272.3	8714	8	268.6263	7436	4	353.6364
Grafo 7	3589	9	185.2929	9082	9	238.8081	8004	7	281.4	9082	9	238.8081	7216	8	294.7677
Grafo 8	1926	18	191.3131	8766	18	275.4949	8766	18	275.5	8766	18	275.4949	6384	40	326.6869
Grafo 9	1645	20	197.4545	8656	20	286.9293	7428	2	326.1	8656	20	286.9293	8054	60	338.2828
Grafo 10	1359	5	191.2323	8608	11	285.3131	8342	5	295.6	8560	7	277.596	8290	3	323.4949
Proms	**2383**		**190.368**			**267.65**			**280**			**263.968**			**315.8586**
Desv			6.282381			22.24474			28.48			20.10034			20.230188

Tabla 5.12: Nodo con mayor número de caminos que lo atraviesan en redes con 100 nodos.

200 Nodos

	1 CENTRO														
	Grafo			Mayor Grado			Centro			Centro Promedio			Arbitrarios		
200 Nodos	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$	#Máx	Nodo	$\bar{X}$
Grafo 1	6128	21	435.86	35406	21	617.7	31074	3	701.83	36342	10	625.71	30226	8	754.91
Grafo 2	10292	16	432.29	36540	16	576.51	32154	1	664.71	36540	16	576.51	34496	7	625.43
Grafo 3	6627	13	423.67	36918	13	581.96	33366	3	645.25	36918	13	581.96	28874	138	674.62
Grafo 4	7469	9	421.19	36888	11	573.82	37128	10	565.2	37128	10	565.2	36494	8	624.3
Grafo 5	17386	1	439.65	36862	1	540.71	36862	1	540.71	36862	1	540.71	36912	1	547.12
Grafo 6	14012	4	427.87	37002	4	559.32	32188	1	709.56	37002	4	559.32	30222	128	650.11
Grafo 7	15802	3	443.21	37162	3	560.84	28522	1	635.12	37162	3	560.84	29578	4	658.37
Grafo 8	12237	1	422.86	36868	1	567.91	36868	1	567.91	36868	1	567.91	36868	1	567.91
Grafo 9	15446	8	410.26	38096	8	519.51	29384	1	640	38096	8	519.51	26182	189	756.13
Grafo 10	9025	15	451.18	35932	15	650.71	31920	1	720.36	36140	7	613.86	35932	15	650.71
Proms	**11442**		**430.804**			**574.899**			**639.065**			**571.153**			**650.961**
Desv			12.0748			37.0568			63.61604			31.326155			67.92513

Tabla 5.13: Nodo con mayor número de caminos que lo atraviesan en redes con 200 nodos.

En la Tabla 5.13 se muestran los resultados que se obtuvieron de la experimentación con grafos que contaban con 200 nodos. En esta tabla se puede observar que el promedio más bajo fue obtenido al seleccionar al nodo centro promedio como el nodo raíz e iniciador del etiquetamiento. Al observar las 4 tablas que muestran los resultados en los 4 diferentes tamaños de grafos se puede concluir que la mejor opción para seleccionar un nodo raíz de los criterios evaluados es seleccionar al nodo considerado como el centro promedio de la red ya que en los cuatro tamaños de grafos mostró un promedio menor a comparación de los otros criterios.

5.2.2. Dos nodos raíz

A continuación se mostrarán los resultados obtenidos en cuestión de la cantidad de caminos que atraviesan a los nodos cuando en los grafos se cuenta con dos nodos raíz. La información está presentada en tablas, las cuales tienen la siguiente estructura.

Para empezar se divide en 3 grandes grupos de acuerdo al criterio de selección de nodos, el primer es el de elegir al centro promedio con el nodo más lejano a él, el segundo es criterio de elegir a los nodos más separados de la red y el último son la elección de dos nodos arbitrarios. En la primera columna está el número máximo de caminos que pasan por un nodo, dicho nodo es el que está en la segunda columna. En la tercera columna está el promedio de caminos para todos los nodos. Las tres primeras columnas corresponden al primer etiquetado en la red el cuál corresponde al primer nodo seleccionado como raíz del criterio correspondiente. Desde la cuarta columna a la sexta la información corresponde a los mismos datos pero ahora con respecto al segundo etiquetado que se realizó en la red que utiliza como nodo raíz al segundo nodo de cada criterio. En la octava columna se presenta el número máximo de caminos que pasan por el nodo cuya identidad se muestra en la novena columna. Este último nodo es el que obtuvo un mayor número de caminos que pasan por él al haber dos opciones de rutas para los destinos, es decir, al utilizar ambos etiquetados y elegir la distancia más corta entre el origen y los destinos. Finalmente en la última columna está el promedio de caminos que atraviesan a los nodos de las redes utilizando ambos etiquetados, eligiendo siempre la distancia menor. En el penúltimo renglón se aprecia el promedio final al tomar como datos los promedios obtenidos en cada grafo para cada criterio y en el último renglón de cada sección de la tabla está el valor de la desviación estándar vinculado con ese promedio.

50 Nodos

2 CENTROS									
Centro y lejano									
50 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	1820	6	128.7347	1554	4	147.5918	1108	6	113.5918
Grafo 2	2110	11	120.8163	1850	5	149.6327	1424	11	102.0816
Grafo 3	2000	5	126.6939	1718	3	159.7551	1368	5	114.5714
Grafo 4	2150	1	100.1633	1660	7	125.8776	1470	1	89.3878
Grafo 5	2064	2	104.4082	1862	7	116.3265	1496	2	94.4082
Grafo 6	2076	2	109.3878	1988	7	126.0408	1308	2	89.8776
Grafo 7	2078	15	127.5102	1550	25	189.7959	1519	15	112.7755
Grafo 8	1886	12	114.2857	1488	2	140.2449	1274	12	102.1224
Grafo 9	2020	16	126.6122	1820	6	152.6531	1266	16	109.102
Grafo 10	2148	7	93.9592	1886	6	128.6531	1570	7	81.5102
Proms			**115.2572**			**143.657**			**100.943**
desv.			12.723302			21.42227			11.6966
Más separados									
50 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	1602	36	171.3469	1802	4	169.0612	838	4	126.2041
Grafo 2	1704	6	142.7755	1850	5	149.6327	791	6	108.7347
Grafo 3	1564	2	212.4898	1724	4	162.1224	1072	4	133.5102
Grafo 4	1610	4	144.3265	1660	7	125.8776	866	43	101.3878
Grafo 5	1844	4	140.1633	1862	7	116.3265	1188	7	97.2245
Grafo 6	1988	7	126.0408	2076	2	109.3878	1316	2	89.8776
Grafo 7	1822	15	147.5918	1520	6	162.0408	1259	15	121.7959
Grafo 8	1992	6	124.2449	1994	7	130.2041	1018	6	100.7347
Grafo 9	1626	3	156.6531	1552	36	160.2449	715	6	121.8776
Grafo 10	2148	7	93.9592	1886	6	128.6531	1570	7	81.5102
Proms			**145.9592**			**141.355**			**108.286**
desv.	12.7233			21.42227			11.6966		
Arbitrarios									
50 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	1886	6	144.8163	1802	5	154.6939	1097	6	120.1224
Grafo 2	1850	5	149.6327	1808	11	129.0612	1274	11	105.5918
Grafo 3	1704	6	142.2041	1756	48	170.7755	1070	5	121.3469
Grafo 4	1580	42	139.0204	1724	6	111.5102	1082	6	97.2653
Grafo 5	1862	7	116.3265	1834	3	124.7347	1114	7	99.9184
Grafo 6	2082	2	115.2653	1632	6	124.9796	1365	2	92.7347
Grafo 7	1936	15	141.551	1588	3	153.551	1579	15	126.0816
Grafo 8	1810	5	138.0408	1608	37	150.3673	1192	6	116.6939
Grafo 9	1626	3	156.6531	2040	16	130.9388	1527	16	114.6531
Grafo 10	1884	5	120.3265	1744	4	118.0408	1149	7	98.4082
Proms			**136.3837**			**136.865**			109.2816
desv.	31.28104			21.66853			17.02138		

Tabla 5.14: Nodo con mayor número de caminos que lo atraviesan en redes con 50 nodos y dos nodos raíz.

En la Tabla 5.14 se muestra la información descrita anteriormente y con base en ella se puede concluir que la forma de reducir el promedio de caminos sobre los nodos es eligiendo al centro promedio y al nodo más alejado a él como nodos raíz.

La Tabla5.15 contiene en ella los resultados obtenidos en los experimentos realizados sobre los grafos con 75 nodos y contando con dos nodos raíz. Los datos arrojados dan como resultado que la elección de los nodos centro promedio y el nodo más alejado a él reducen el promedio de los caminos por nodos en los grafos que cuentan con 75 nodos en ellos.

75 Nodos

2 CENTROS									
Centro y lejano									
75 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	4932	5	179.5676	4630	6	211.2703	3115	5	158.7297
Grafo 2	4922	20	171.7297	4562	6	212.4595	3271	20	151.4595
Grafo 3	4778	19	200.8378	3334	71	272.3514	3264	19	177.1892
Grafo 4	4956	10	174.1351	4424	8	219.973	3308	10	151.5946
Grafo 5	4822	1	180.8108	3514	73	288.5135	3855	1	167.1351
Grafo 6	4702	8	195.1081	4348	6	214.8378	2632	8	163.4865
Grafo 7	4904	2	173.6486	3550	61	252.973	4027	2	162.7568
Grafo 8	4932	10	168.3243	3676	55	249.0811	3947	10	156.1081
Grafo 9	4944	19	178.027	3938	2	225.4595	3516	19	156.9189
Grafo 10	4968	13	173.1351	4274	1	272.2703	4066	13	162.1351
Proms			**179.5324**			**241.919**			**160.751**
desv.			10.496578			28.83864			7.709243
Más separados									
75 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	4710	5	208.1892	3844	3	234.6216	3104	5	176.6216
Grafo 2	3606	6	233.3784	4254	4	218.0541	2023	4	174.1892
Grafo 3	3786	6	239.5405	4510	3	226.8919	2313	3	189.4865
Grafo 4	3934	4	234.6486	4424	8	219.973	2197	8	175.0541
Grafo 5	4522	1	189.8108	4090	6	211.6486	2912	1	164.6757
Grafo 6	4484	7	234.2432	4324	4	209.5676	2396	4	176.1892
Grafo 7	4524	2	245.2432	3980	4	250.9189	3078	2	194.2432
Grafo 8	4124	1	245.3784	4322	7	213.4595	2535	7	177.4865
Grafo 9	3938	2	225.4595	4338	5	260.7838	2149	2	192.1622
Grafo 10	4366	7	214.2162	4322	5	239.7568	2454	7	178.0811
Proms			**227.0108**			**228.568**			**179.819**
desv.			17.895292			17.51834			9.246325
Arbitrarios									
75 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	4184	4	256.3243	4638	5	200.2432	3155	5	169.9189
Grafo 2	4188	59	212.8378	4852	8	196.4324	2721	8	164.6757
Grafo 3	4066	5	243.7027	4252	2	255.7838	2185	5	206.1892
Grafo 4	3758	53	217.2432	4246	6	228.8919	2294	10	181.8108
Grafo 5	4254	1	208.5135	3646	6	246.1081	3112	1	182.8378
Grafo 6	3660	2	228.8919	4324	4	209.5676	2453	4	186.3784
Grafo 7	3986	9	186.5135	4908	2	177.4324	3768	2	161.7838
Grafo 8	4240	7	191.4595	4928	10	172.1081	3476	10	154.6486
Grafo 9	3732	3	237.1662	4266	60	200.1622	2647	60	176.6486
Grafo 10	4088	4	207.1351	4558	8	201.5405	2842	8	176.2432
Proms			**218.9788**			**208.827**			**176.114**
desv.			22.452013			27.25874			14.618

Tabla 5.15: Nodo con mayor número de caminos que lo atraviesan en redes con 75 nodos y dos nodos raíz.

100 Nodos

Para continuar analizando esta métrica se tiene la Tabla 5.16 donde se presentan los resultados que se obtuvieron al contar con dos nodos como iniciadores del etiquetamiento en redes de 100 nodos. En esta tabla al igual que en las dos anteriores, se puede apreciar que al tener como nodos raíz al nodo centro promedio y al nodo más alejado a él, da como resultado un promedio menor en los caminos que pasan sobre todos los nodos de la red.

2 CENTROS									
Centro y lejano									
100 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	8344	66	286.6869	8288	4	318.4444	4654	66	252.2626
Grafo 2	8850	9	248.5051	8304	4	303.5758	6036	9	231.4343
Grafo 3	9094	20	231.5758	8052	1	314.1212	6789	20	214.4848
Grafo 4	8610	2	275.5354	7114	66	355.0909	5805	2	242.303
Grafo 5	8746	1	249.9192	6886	4	294.7677	6048	1	255.7778
Grafo 6	8714	8	268.6263	8148	4	333.5152	5857	8	238.9293
Grafo 7	9082	9	238.8081	8370	8	320.1414	6248	9	208.1616
Grafo 8	8766	18	275.4949	8378	6	318.2828	5371	18	240.9495
Grafo 9	8656	20	286.9293	7104	55	343.8586	5339	20	250.303
Grafo 10	8560	7	277.596	7986	3	336.2222	5448	7	248.404
Proms			**263.9677**			**323.802**			**238.301**
desv.			20.100335			18.40363			15.96056
Más separados									
100 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	6534	49	389.3131	8288	4	318.4444	5527	4	274.0808
Grafo 2	6394	96	298.8485	8704	10	254.8485	6545	10	226.9495
Grafo 3	8486	8	285.4747	7208	2	357.5556	5272	8	247.899
Grafo 4	7114	66	355.0909	7704	2	314.7273	4347	2	262.1818
Grafo 5	6554	74	375.0101	7566	73	331.9798	3975	73	267.8182
Grafo 6	8148	4	333.5152	7538	17	285.4747	4390	17	245.8384
Grafo 7	6286	97	369.1111	7496	3	354	3610	3	283.2727
Grafo 8	6446	97	380.9495	8314	7	332.5455	4782	7	279.2121
Grafo 9	7104	55	343.8586	6950	56	323.2525	3220	56	271.3333
Grafo 10	7542	4	326.5253	8400	9	277.8384	5323	9	249.2929
Proms			**345.7697**			**315.067**			**260.788**
desv.			34.93106			33.13116			17.79849
Arbitrarios									
100 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	7902	3	298.7677	6944	5	320.5051	4097	3	256.6162
Grafo 2	8252	5	274.9697	7292	75	343.2525	5598	5	249.3939
Grafo 3	8038	4	307.0505	7610	9	289.2323	3898	9	243.9798
Grafo 4	8498	2	306.404	7554	2	298.2828	7161	2	268.9495
Grafo 5	8266	7	289.7879	7582	6	266.8081	4385	1	228.8789
Grafo 6	8196	7	302.2424	8252	6	281.2727	4713	6	241.0707
Grafo 7	6660	71	293.0707	8556	8	310.6465	3940	9	237.1515
Grafo 8	7724	62	312.0202	7124	59	316.3838	3301	62	257.1111
Grafo 9	7926	62	319.7778	7100	57	326.6061	3905	62	276.101
Grafo 10	8214	6	336.9495	7504	71	306.2828	4567	71	265.798
Proms			**304.104**			**305.927**			**252.505**
desv.			17.024709			22.70671			15.08877

Tabla 5.16: Nodo con mayor número de caminos que lo atraviesan en redes con 100 nodos y dos nodos raíz.

200 Nodos

La Tabla 5.17 se presentan los datos arrojados de la experimentación con los grafos que cuentan con 200 nodos. El promedio total menor en relación con los caminos que pasan por un nodo está dado por el criterio de selección donde se eligen al nodo centro promedio y al más alejado a él como nodos raíz para iniciar los etiquetados dentro de la red.

2 CENTROS									
Centro y lejano									
200 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	36342	10	625.71	28278	110	857.29	26265	10	565.62
Grafo 2	36540	16	576.51	33906	4	744.79	26093	16	533.9
Grafo 3	36918	13	581.96	33936	8	613.98	19842	13	501.87
Grafo 4	37128	10	565.2	35202	5	689.6	26053	10	531.66
Grafo 5	36862	1	540.71	31854	4	654.51	25618	1	497.41
Grafo 6	37002	4	559.32	30398	9	819	28688	4	515.22
Grafo 7	37162	3	560.84	26922	104	936	30644	3	526.33
Grafo 8	36868	1	567.91	27292	114	847.98	29135	1	529.24
Grafo 9	38096	8	519.51	28428	101	782.26	29331	8	471.58
Grafo 10	36140	7	613.86	29904	110	796.23	26365	7	568.92
Proms			**571.153**			**774.164**			**524.175**
desv.			31.326155			99.49999			29.74149
Más separados									
200 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	28278	110	857.29	32768	5	715.99	20279	5	622.42
Grafo 2	32908	3	676.84	34512	6	679.03	16865	6	576.18
Grafo 3	35668	6	609.16	36046	9	673.09	19240	6	528.46
Grafo 4	33432	1	679.28	35202	5	689.6	16149	5	576.96
Grafo 5	36912	1	550.96	31854	4	654.51	25348	1	502.31
Grafo 6	34870	4	782.72	30938	5	700.72	21366	4	574.28
Grafo 7	28034	192	874.7	34776	7	665.99	24013	7	589.41
Grafo 8	33208	8	688.3	30596	4	643.47	18680	1	545.7
Grafo 9	28992	105	832.76	30012	7	675.11	18416	7	577.05
Grafo 10	29954	92	821.26	33974	7	677.88	21705	7	601.79
Proms			**737.327**			**677.539**			**569.456**
desv.			111.58774			21.11805			35.2529
Arbitrarios									
200 nodos	#Máx1	Nodo	$\bar{X}$	#Máx2	Nodo	$\bar{X}$	Max	Nodo	$\bar{X}$
Grafo 1	26222	187	868.07	35406	21	617.7	26450	21	566.88
Grafo 2	34512	6	679.03	32498	8	657.32	18988	8	557.31
Grafo 3	35724	5	671.53	32056	3	715.88	20795	5	585.57
Grafo 4	28058	5	707.64	32846	5	705.5	27679	5	651.47
Grafo 5	31890	1	608.61	28744	1	659.12	27742	1	548.11
Grafo 6	37002	4	559.32	27342	6	701.68	27685	4	517.5
Grafo 7	29138	114	681.26	32416	5	742.96	18588	7	585.33
Grafo 8	36694	1	608.56	27962	108	793.03	28493	1	568.37
Grafo 9	38096	8	519.51	32070	116	660.3	27499	8	474.96
Grafo 10	28750	6	746.18	34224	15	701.95	22034	15	618.53
Proms			**664.971**			**695.544**			**567.403**
desv.			99.359368			49.92524			49.26568

Tabla 5.17: Nodo con mayor número de caminos que lo atraviesan en redes con 200 nodos y dos nodos raíz.

Dentro de cada grafo, se calculó el promedio de caminos que atraviesan por cada nodo, ésto se realizó sumando la cantidad total de caminos que cada nodo reportó que atraviesan por él y al final se dividió entre todos los nodos que aportaron caminos. Cabe aclarar que no se consideró como el mismo camino una ruta del nodo a al nodo b que la ruta del nodo b al nodo a, debido a que existen alternativas en las rutas y se puede dar el caso que de ida se elija un camino y de regreso otro, siempre y cuando cuenten con la misma longitud. Posteriormente, con el promedio de cada grafo dentro de cada grupo de grafos con la misma cantidad de nodos, se calculó un promedio, ese promedio de caminos es lo que se muestra en la Figura 5.3

En la gráfica mostrada en la Figura 5.3 lo que se muestran son los promedios totales de cada criterio de acuerdo con la cantidad de nodos en la red, como se puede apreciar el criterio que más se acerca a los valores de los caminos sobre el grafo original es el criterio para elegir dos nodos raíz el cual corresponde al nodo centro promedio y al nodo más alejado a él. Con esta gráfica se comprueba que la elección de nodos raíz con cierto criterio resulta en mejoras en cuanto a las métricas ya que también en el esquema de un sólo nodo raíz se puede ver que la mejor manera de tenerlo es seleccionando al nodo centro promedio como raíz de la red. Los valores expresados en la gráfica se encuentran en la Tabla 5.18 y además se muestra el valor de la desviación estándar σ para cada valor.

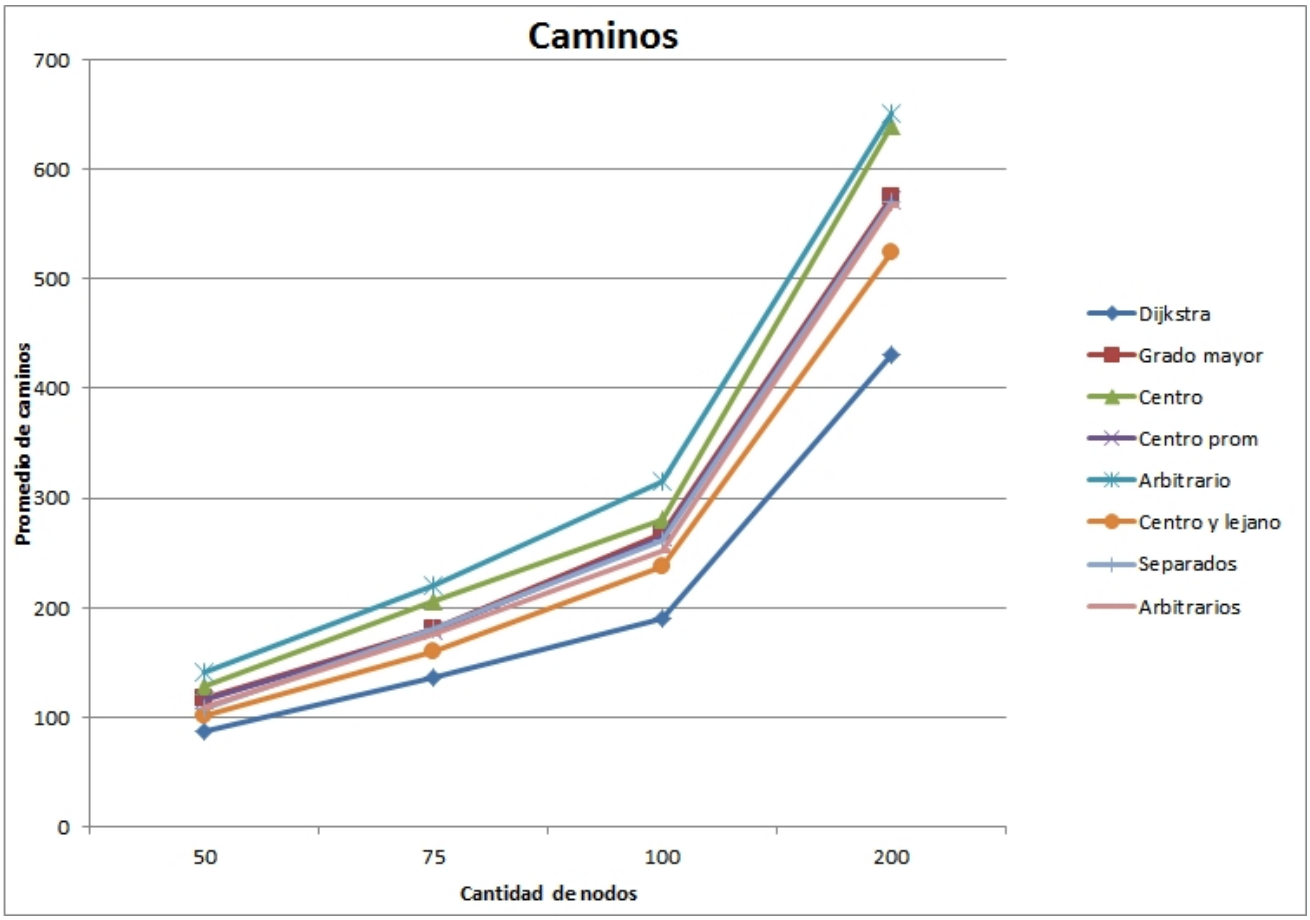


Figura 5.3: Comparación en los promedios de los caminos que atraviesan a los nodos.

	1 CENTRO										2 CENTROS					
Nodos	Dijkstra	σ	Grado	σ	Centro	σ	Centro P	σ	Arb	σ	C y l	σ	Separados	σ	Arbs	σ
50	87.8	10.193	117.96	15.014	127.886	15.015	100.94	12.723	140.82	23.558	100.943	11.697	108.29	17.021	109.28	11.864
75	136.82	3.6133	179.88	11.116	205.989	22.401	179.53	10.497	220.88	24.486	160.751	7.7092	179.82	9.2463	176.11	14.618
100	190.37	6.2824	267.65	22.245	280.226	28.485	263.97	20.1	315.86	20.23	238.301	15.961	260.79	17.798	252.51	15.089
200	430.8	12.075	574.9	37.057	639.065	63.616	571.15	31.326	650.96	67.925	524.175	29.741	569.46	35.253	567.4	49.266

Tabla 5.18: Resultados condensados de los experimentos en relación con los caminos que pasan sobre los nodos.

5.3. Stretch

En esta sección se mostrarán la información arrojada al realizar los experimentos y tomar en cuenta el *stretch* presentado en cada caso. Es bueno recordar la fórmula con la cual se está calculando esta métrica y a continuación se muestra.

$$MAX_{x,y \in N}(\frac{Distancia_{CR}(x,y)}{Distancia_{OPT}(x,y)}) \tag{5.1}$$

5.3.1. Un sólo nodo raíz

A continuación se presentarán las tablas que reúnen los resultados obtenidos al calcular el *stretch* en cada grafo del conjunto de los grafos con un número de nodos igual a 50. Las tablas presentan la siguiente estructura. En cada columna se describe el criterio al cual corresponden los resultados. El número presente en cada celda es el cálculo mayor que se obtuvo dentro de cada grafo, por lo tanto se considera el *stretch* mayor. El penúltimo renglón muestra el promedio obtenido en relación con la métrica que se está abordando en esta sección y dicho promedio se realizó en cada grupo de grafos con la misma cantidad de nodos y el último renglón muestra la desviación estándar para cada promedio calculado. En las columnas de la tabla se tienen todos los criterios planteados para seleccionar a un sólo nodo raíz.

50 Nodos

	1 CENTRO			
50 Nodos	Grado mayor	Centro	Centro prom.	Arbitrario
Grafo 1	6	5	5	6
Grafo 2	6	5	6	8
Grafo 3	5	5	5	5
Grafo 4	4	4	4	6
Grafo 5	4	5	4	5
Grafo 6	5	5	5	6
Grafo 7	5	6	5	7
Grafo 8	5	5	5	5
Grafo 9	5	5	5	6
Grafo 10	4	5	4	5
Proms	4.9	5	4.8	5.9
desv.	0.7	0.4472136	0.6	0.94339811

Tabla 5.19: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de un nodo raíz en grafos con 50 nodos.

En la Tabla 5.19 se muestran la información del cálculo de la métrica *stretch* para los grafos con 50 nodos en ellos. Con los datos proporcionados se puede ver que el máximo valor del *stretch* para este grupo de grafos es de 6 y el *stretch* promedio máximo para el conjunto de grafos sería de 5,7. Los datos presentados muestran que al tener como nodo raíz al nodo centro promedio y al nodo con el mayor grado se obtienen mejores resultados en cuanto al *stretch* en los grafos de 50 nodos.

75 Nodos

	1 CENTRO			
75 Nodos	Grado mayor	Centro	Centro prom.	Arbitrario
Grafo 1	5	5	5	6
Grafo 2	4	5	4	6
Grafo 3	5	6	5	6
Grafo 4	5	5	5	6
Grafo 5	5	5	5	5
Grafo 6	5	5	5	6
Grafo 7	4	4	4	5
Grafo 8	5	5	5	5
Grafo 9	5	6	5	6
Grafo 10	5	5	5	6
Proms	4.8	5.1	4.8	5.7
desv.	0.4	0.53851648	0.4	0.45825757

Tabla 5.20: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de un nodo raíz en grafos de 75 nodos.

En la tabla 5.20 se muestra la información de la misma manera que en la tabla anterior, pero ahora para los grafos con 75 nodos. El *stretch* máximo para este conjunto es de 6 y el valor promedio máximo para este caso es de 5,7. El resultado obtenido aquí es similar a la tabla anterior, lo cual indica que los criterios de elección del nodo con el grado mayor y el nodo centro promedio reducen el promedio total del *stretch* en los grafos con 75 nodos.

100 Nodos

	1 CENTRO			
100 Nodos	Grado mayor	Centro	Centro prom.	Arbitrario
Grafo 1	5	5	5	6
Grafo 2	5	5	5	6
Grafo 3	4	4	4	6
Grafo 4	6	6	6	7
Grafo 5	5	5	5	6
Grafo 6	6	6	5	6
Grafo 7	5	5	5	7
Grafo 8	6	6	6	6
Grafo 9	5	6	5	6
Grafo 10	5	5	5	7
Proms	5.2	5.3	5.1	6.3
desv.	0.6	0.64031242	0.538516481	0.45825757

Tabla 5.21: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de un nodo raíz en grafos de 100 nodos.

En la Tabla 5.21 se puede apreciar que el cálculo del *stretch* máximo para el conjunto de grafos con 100 nodos es de 7 ya que es el valor mayor encontrado en ella y que el valor promedio máximo corresponde a 6,3, esta información se obtuvo de los experimentos realizados. En cuanto al *stretch* en los grafos con los que se trabajó que contaban con 100 nodos, se ve que la estrategia a seguir es seleccionar al nodo centro promedio para obtener mejores resultados.

200 Nodos

	1 CENTRO			
200 Nodos	Grado mayor	Centro	Centro prom.	Arbitrario
Grafo 1	6	7	6	8
Grafo 2	7	7	7	7
Grafo 3	6	6	6	6
Grafo 4	5	5	5	6
Grafo 5	5	5	5	5
Grafo 6	6	6	6	7
Grafo 7	6	5	6	6
Grafo 8	5	5	5	5
Grafo 9	5	6	5	7
Grafo 10	7	6	6	7
Proms	5.8	5.8	5.7	6.4
desv.	0.748331477	0.74833148	0.640312424	0.91651514

Tabla 5.22: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de un nodo raíz en grafos de 200 nodos.

Para los grafos con 200 nodos en ellos, el cálculo del *stretch* máximo corresponde a 8 debido a que es el valor más grande encontrado y el valor del *stretch* promedio es de 6,4. La manera para obtener un promedio menor en cuanto al *stretch* promedio es eligiendo como nodo raíz de la red al nodo considerado el centro promedio, ya que cómo se muestra en la tabla, es el promedio más bajo.

De las cuatro tablas anteriores se puede concluir que en relación con el *stretch*, la manera de obtener mejores resultados (un *stretch* menor) teniendo un sólo nodo raíz en la red es mediante el criterio de selección del centro promedio de la red. Esta estrategia arrojó los promedios más bajos en los diferentes tamaños de las topologías con las que se experimentó.

5.3.2. Dos nodos raíz

De la misma manera que se presentó la información acerca del cálculo del *stretch* en el esquema de un sólo nodo raíz, a continuación se exhiben los resultados obtenidos en el esquema de doble etiquetado. La información se presenta en las tablas de la siguiente manera. En la primer y segunda columna se muestra el valor máximo calculado dentro del primer y segundo etiquetado respectivamente. En la tercer columna se muestra el valor máximo calculado al aplicar la selección de la ruta más corta entre ambos etiquetados. Lo anterior se realiza para cada uno de los 3 criterios de selección que se propusieron para elegir a dos nodos raíz en la red. El resultado del *stretch* máximo se buscará sólo en la columna en la que se mezclan las rutas, es decir sólo se buscará en la tercera columna de cada criterio. En el penúltimo renglón se aprecia el cálculo del promedio de dicha métrica para el grupo de grafos con el mismo número de nodos y al final se muestran los valores de la desviación estándar.

50 Nodos

	2 CENTROS								
	Centro y Lejano			Más Separados			Arbitrarios		
50 Nodos	Et1	Et2	mix	Et1	Et2	mix	Et1	Et2	mix
Grafo 1	5	6	5	7	8	6	6	6	4
Grafo 2	6	7	4	6	7	4	7	5	4
Grafo 3	5	6	4	10	6	5	6	8	5
Grafo 4	4	5	4	6	5	5	6	5	4
Grafo 5	4	5	4	6	5	4	5	5	4
Grafo 6	5	5	5	5	5	5	6	5	4
Grafo 7	5	8	4	6	6	4	5	6	5
Grafo 8	5	6	5	5	5	4	5	6	5
Grafo 9	5	5	4	7	6	5	7	5	5
Grafo 10	4	5	3	4	5	3	5	4	4
Proms			4.2			4.5			4.4
desv.			0.6			0.8062			0.4899

Tabla 5.23: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de dos nodos raíz.

De acuerdo con la información en la Tabla 5.24 el valor del *stretch* para este grupo de grafos es de 6 ya que es el valor más grande encontrado al hacer uso de ambos etiquetados y el valor promedio es 4.5 y que el criterio que logra obtener mejores resultados en esta métrica es seleccionando al centro promedio y al nodo más alejado a él.

75 Nodos

	2 CENTROS								
	Centro y Lejano			Más Separados			Arbitrarios		
75 Nodos	Et1	Et2	mix	Et1	Et2	mix	Et1	Et2	mix
Grafo 1	5	5	4	5	5	5	6	5	5
Grafo 2	4	5	4	6	5	4	5	5	5
Grafo 3	5	7	5	6	6	5	6	7	5
Grafo 4	5	6	4	6	6	5	6	5	5
Grafo 5	5	6	4	5	5	4	6	6	6
Grafo 6	5	5	4	7	5	5	6	5	5
Grafo 7	4	7	4	7	7	6	5	4	4
Grafo 8	5	6	4	6	6	5	6	5	5
Grafo 9	5	6	5	6	7	5	6	6	5
Grafo 10	5	8	5	6	6	5	6	5	4
Proms			4.3			4.9			4.9
desv.			0.45826			0.5385			0.5385

Tabla 5.24: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de dos nodos raíz en grafos de 75 nodos.

Al buscar los valores máximos dentro de cada criterio y en los promedios, se puede concluir que el valor del *stretch* para este caso es de 6 y el valor promedio corresponde a 4.9. Cuando hay 75 nodos en los grafos el criterio que de nuevo resultó ser el mejor de acuerdo a la información obtenida fue eligiendo al nodo centro promedio y al nodo más alejado a él ya que mostró números menores.

100 Nodos

	2 CENTROS								
	Centro y Lejano			Más Separados			Arbitrarios		
100 Nodos	Et1	Et2	mix	Et1	Et2	mix	Et1	Et2	mix
Grafo 1	5	7	5	7	7	6	5	6	5
Grafo 2	5	5	5	8	5	5	5	7	4
Grafo 3	4	5	4	5	7	5	5	6	5
Grafo 4	6	8	5	8	7	6	7	7	7
Grafo 5	5	6	5	8	7	6	6	6	5
Grafo 6	5	6	4	6	7	5	6	6	5
Grafo 7	5	7	4	7	7	6	6	6	4
Grafo 8	6	6	4	7	8	5	6	6	5
Grafo 9	5	6	5	6	6	5	6	6	5
Grafo 10	5	7	5	6	5	4	7	6	5
Proms			4.6			5.3			5
desv.			0.4899			0.6403			0.7746

Tabla 5.25: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de dos nodos raíz en grafos de 100 nodos.

Para los grafos con 100 nodos y tomando en cuenta la información en la Tabla 5.25 los valores del *stretch*y del *stretch* promedio son 7 y 5.3 respectivamente. Los datos obtenidos también muestran que la mejor opción de selección de dos nodos raíz sigue siendo el nodo centro promedio y el nodo más alejado a éste ya que los números son menores que los números obtenidos de los otros criterios

200 Nodos

	2 CENTROS								
	Centro y Lejano			Más Separados			Arbitrarios		
200 Nodos	Et1	Et2	mix	Et1	Et2	mix	Et1	Et2	mix
Grafo 1	6	8	6	8	7	6	8	6	5
Grafo 2	7	7	6	7	8	7	8	7	7
Grafo 3	6	5	5	5	7	5	7	7	6
Grafo 4	5	7	5	7	7	5	6	6	6
Grafo 5	5	6	5	5	6	5	6	8	5
Grafo 6	6	8	5	9	7	7	6	7	5
Grafo 7	6	8	5	8	6	5	6	7	6
Grafo 8	5	8	5	7	7	5	6	7	6
Grafo 9	5	7	5	8	6	5	5	5	5
Grafo 10	6	7	5	8	6	5	8	8	5
Proms			5.2			5.5			5.6
desv.			0.4			0.8062			0.6633

Tabla 5.26: Comparación del *stretch* mayor en cada grafo de acuerdo con el criterio de selección de un nodo raíz en grafos de 200 nodos.

Finalmente, los valores del *stretch* máximo y *stretch* promedio son de 7 y 5,6 respectivamente. La estrategia de selección de dos nodos raíz que dio como resultado un *stetch* promedio menor fue elegir al centro promedio de la red y al nodo más alejado a él. Esta información se encuentra en la Tabla 5.26

Para cada grupo de grafos se sacó un promedio, en el cual contaron los valores máximos encontrados para cada grafo. Ese promedio es el que se plasma en la Figura 5.4 y nos ayudará a ver la comparación del *stretch* promedio de cada criterio de selección tanto de un solo nodo raíz como de dos nodos raíz. Los valores de los promedios junto con sus respectivas desviaciones estándar se encuentran en la Tabla 5.27

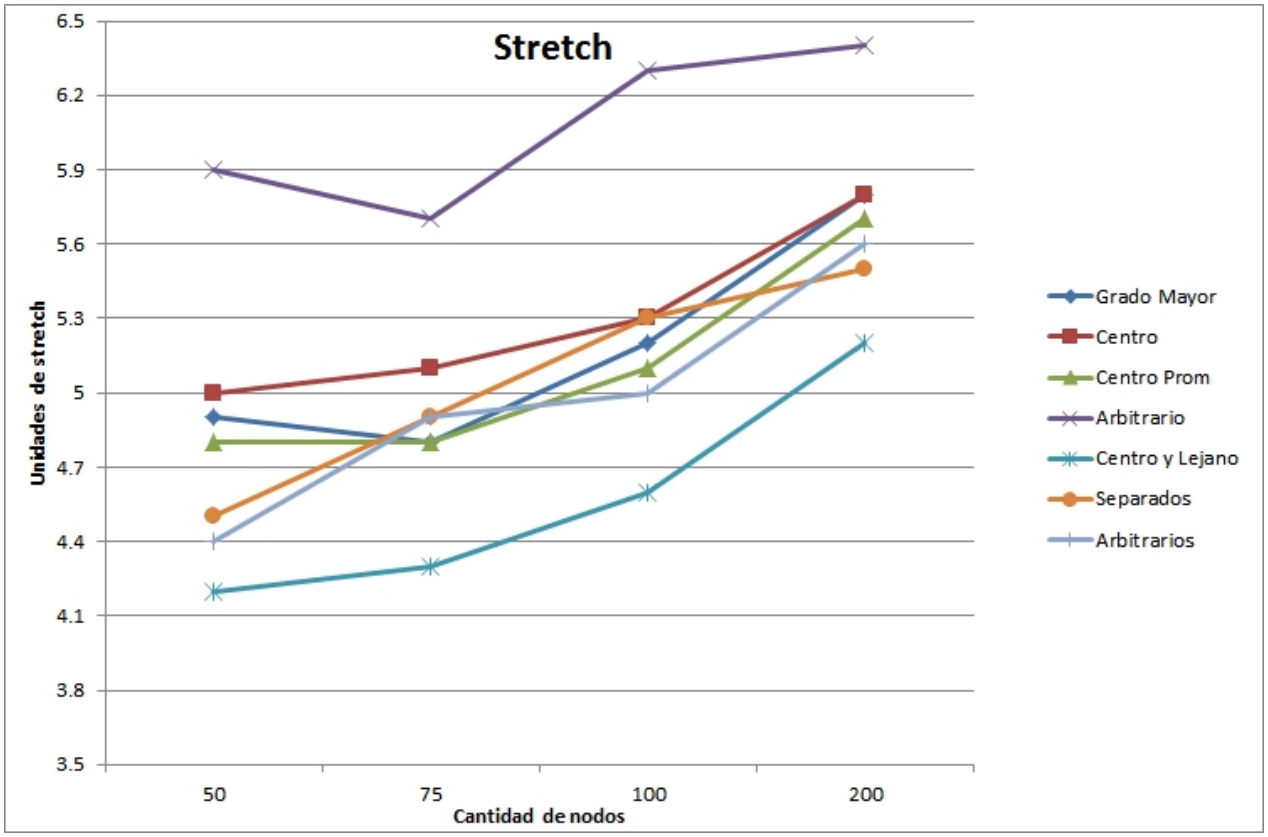


Figura 5.4: Comparación en los promedios del *stretch* para cada grupo de grafos con la misma cantidad de nodos.

	1 CENTRO								2 CENTROS					
Nodos	Grado	σ	Centro	σ	Centro P.	σ	Arb	σ	C y l	σ	Separados	σ	Arbs	σ
50	4.9	0.7	5	0.4472	4.8	0.6	5.9	0.9434	4.2	0.6	4.5	0.8062	4.4	0.4899
75	4.8	0.4	5.1	0.5385	4.8	0.4	5.7	0.4583	4.3	0.4583	4.9	0.5385	4.9	0.5385
100	5.2	0.6	5.3	0.6403	5.1	0.5385	6.3	0.4583	4.6	0.4899	5.3	0.6403	5	0.7746
200	5.8	0.7483	5.8	0.7483	5.7	0.6403	6.4	0.9165	5.2	0.4	5.5	0.8062	5.6	0.6633

Tabla 5.27: Resultados condensados de los experimentos en relación con el *stretch*.

Gracias a la información obtenida se puede concluir que al tener dos nodos raíz se mejoran las métricas que se consideraron. Dichos nodos deben de ser el nodo centro promedio y el nodo más alejado a él ya que se observó que siempre dieron como resultado un *stretch* menor y como se aprecia en la gráfica al sólo contar con un nodo raíz la manera de obtener mejores resultados es con el nodo centro promedio. Con esto se concluye una vez más de la importancia de la selección del y de los nodos raíz ya que presentaron mejores números que al no invertir recursos para la selección. Es necesario aclarar que aun cuando los números obtenidos son lejanos al 1, que en el enrutamiento compacto es la medida ideal del *stretch*, estos valores son tomados de un sólo viaje que se realiza para poder obtener el contenido a diferencia del esquema de enrutamiento tradicional en el cual es necesario realizar más viajes dentro de la red para obtener el contenido deseado.

5.4. Resumen del capítulo

En este capítulo se mostraron los resultados relacionados con las métricas que se seleccionaron para esta tesis. Las métricas a considerar fueron: la longitud de las rutas obtenidas por el algoritmo propuesto, la cantidad máxima de caminos que pasan sobre cualquier nodo y finalmente el tamaño del *stretch*. Gracias a los resultados obtenidos se pudo concluir que invertir recursos de la red para la selección de un nodo raíz como de dos nodos raíz representa mejoras en cuanto a las métricas que se tomaron en cuenta. Para el caso en el que sólo haya un nodo raíz en la red, la estrategia a seguir es seleccionar al centro promedio de la red y cuando existen dos nodos raíz dentro de la misma red, el mejor criterio es seleccionar al centro promedio de la red y al nodo más alejado de él. En el siguiente capítulo se presentará la conclusión del presente trabajo.

Capítulo 6

Conclusiones

A lo largo de este trabajo se ha mostrado el desarrollo del diseño del algoritmo de enrutamiento compacto usando etiquetas basadas en prefijos. Dicho algoritmo se aplica en redes de computadoras sin cambios topológicos. Posteriormente se verificó su funcionamiento mediante una serie de experimentos los cuales ayudaron a caracterizarlo. Parte del trabajo realizado consistió en el diseño de una serie de algoritmos para la selección tanto de un nodo raíz como de dos nodos raíz. Posteriormente se procedió a verificar su correcto funcionamiento, esto mediante una serie de demostraciones realizadas sobre los mismos algoritmos. Por medio de la realización de una serie de experimentos se logró caracterizar el comportamiento de los algoritmos propuestos. Los algoritmos para la selección de centros corresponden a los criterios con los cuales se pensó se obtendrían mejores resultados. Para la elección de un nodo raíz se diseñaron algoritmos basados en los siguientes criterios.

- Selección del nodo con el grado mayor.
- Selección del nodo considerado el centro de la red. Dicho nodo es aquel cuya distancia a su nodos más alejado es mínima.
- Selección del nodo considerado el centro promedio de la red. Dicho nodo es aquel cuya distancia promedio a todos los demás nodos en la red es mínima.

Para el algoritmo propuesto, se requieren de la selección de dos nodos raíz, por lo tanto los algoritmos realizados para elegir a dos nodos dentro de la red como raíces estuvieron basados en los siguientes criterios.

- Selección del nodo considerado el centro promedio de la red y el nodo más alejado a él.
- Selección de los nodos más alejados en toda la red.
- Selección arbitraria de dos nodos.

Una vez que se eligió uno o dos nodos raíces, se continuó con un etiquetamiento o dos, dependiendo del caso, de todos los nodos de la red, ésto mediante un algoritmo descrito en la Sección 4 de este trabajo. El algoritmo de etiquetado respetó las relaciones que existen entre los nodos al inducirse el árbol lógico, es decir, las etiquetas asignadas a los nodos mostraban una relación padre e hijo, siendo la etiqueta del padre el prefijo de la etiqueta del hijo. Una vez que todo está etiquetado, el enrutatmiento se realiza de manera sencilla dentro de la red, simplemente buscando la mayor coincidencia de las etiquetas de los vecinos con la etiqueta destino. Dado este esquema de enrutamiento se puede apreciar que el tamaño de las tablas de enrutamiento es del orden de $O(d)$, donde d es el grado máximo dentro de la red. Gracias al diseño del algoritmo propuesto es que se logró contar con las características mencionadas

anteriormente. En el capítulo 4 es en donde se verificó que el algoritmo funciona de manera correcta y posteriormente en el Capítulo 5 se muestran los resultados a los experimentos para poder caracterizar su funcionamiento.

Basados en los experimentos realizados y de acuerdo con la información recabada y mostrada en el capítulo 5, se puede concluir que en definitivo es mejor utilizar los recursos con los que cuenta la red para poder hacer una elección correcta del nodo que fungirá como raíz, ésto debido a que como se observó, una elección correcta del nodo raíz genera rutas de longitud más corta que un nodo elegido arbitrariamente. Para el caso de un sólo nodo raíz, se concluyó que el criterio que ayuda a reducir el *stretch* es el del nodo centro promedio. Por lo tanto una mejora al esquema considerado como base es que elija a su nodo raíz con base en este criterio.

Con respecto al esquema de doble etiquetado, el mejor criterio de selección de nodos raíz es en donde se elige el nodo centro promedio de la red en conjunto con el nodo más alejado a él. En el Capítulo 5 se observó mediante los datos obtenidos que al elegir a ese par de nodos se obtienen rutas de longitud menores, que en cierta medida se reduce la cantidad de caminos que atraviesan a la raíz y a los nodos cercanos a ella y que fue el criterio que más se acercó en las métricas seleccionadas al enrutamiento de camino más corto. Por lo tanto se concluyó que esa es la mejor manera de hacer la selección de los dos nodos raíz de la red.

6.1. Aportaciones

Una de las aportaciones del presente trabajo está relacionada con el esquema que se consideró como base. La aportación es que se pudo obtener una manera de elegir al nodo raíz mejor que con la que contaba el trabajo, ayudando así a la mejora del mismo.

Otra aportación importante es que se deja una base teórica dentro de un campo de vital importancia en la actualidad como lo son las redes de computadoras y el enrutamiento dentro de ellas. La base que se deja es en el enrutamiento compacto utilizando etiquetas basadas en prefijos y como se dijo es una base teórica la cual puede ser utilizada para desarrollar trabajo teórico dentro del mismo campo, con mejoras al trabajo realizado, de la misma manera que éste ayudó al esquema que se utilizó como base y mejorando aspectos del mismo algoritmo propuesto.

Gracias a los resultados obtenidos es factible seguir experimentando con este algoritmo de enrutamiento que utiliza etiquetas basadas en prefijos, pero ahora en circunstancias más complejas (o reales) que en las del presente trabajo.

Así mismo el enrutamiento compacto que usa etiquetas basadas en prefijos que se plantea en la tesis, se adapta al enrutamiento que se prevee será empleado en el un futuro cercano y éste es el enrutamiento orientado a contenido.

6.2. Trabajo a futuro

A continuación se mencionarán los aspectos a mejorar del presente trabajo.

Como trabajo futuro se contempla mejorar la manera en que el enrutamiento se realiza cuando hay dos nodos raíz. Debido a que sólo el primer nodo calcula la distancia en saltos en ambos etiquetados a su destino y con base en eso hace la decisión de qué ruta tomar, se cree que si cada nodo es capaz de realizar este cálculo la distancia final será más pequeña aún.

Considerar a los cambios topológicos en la red para adaptar el algoritmo propuesto a éstos e implementarlo. El objetivo aquí es doble. Por un lado reducir la tormenta de etiquetado, así como minimizar el número de nodos que tienen que cambiar de etiqueta como respuesta a los cambios topológicos. Lo anterior es particularmente importante si los cambios de etiquetas implican que posteriormente se tendrán que migrar los objetos de datos que son asignados a los nodos.

Analizar y caracterizar el algoritmo propuesto en redes con movilidad de nodos.

Realizar experimentos más realistas considerando los aspectos técnicos de las telecomunicaciones en el entorno de simulación.

Determinar una cota teórica más estrecha para el *stretch* máximo del algoritmo propuesto, en particular cuando se utilizan múltiples etiquetados.

Realizar tanto un estudio teórico como práctico acerca del valor óptimo del número de etiquetados con respecto a las diferentes métricas.

Desarrollar y verificar un algoritmo que encuentre al centro que optimice las diferentes métricas que evalúan la calidad de un etiquetado basado en prefijos.

Bibliografía

[1] Bates, T., Chandra, R., Katz, D., and Rekhter, Y. Multiprotocol extensions for bgp-4.

[2] Bu, T., and Towsley, D. On distinguishing between internet power law topology generators. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2002), vol. 2, IEEE, pp. 638–647.

[3] Cho, K., Choi, J., Ko, D.-i. D., Kwon, T., and Choi, Y. Content-oriented networking as a future internet infrastructure: Concepts, strengths, and application scenarios. *Future Internet Technologies* (2008).

[4] Cyril, G. A survey on interval routing. *Theorical Computer Science* (2000).

[5] David, P., and Eli, U. A trade-off between space and efficiency for routing tables. *Journal of the Association for Computing Machinery* (1989).

[6] Doval, D., and O'Mahony, D. Overlay networks: A scalable alternative for p2p. *IEEE Internet computing 7* (2003).

[7] Eilam, T., Gavoille, C., and Peleg, D. Compact routing schemes with low stretch factor. *Journal of Algorithms* (2003).

[8] Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the internet topology. *SIGCOMM Comput. Commun. Rev.* (1999).

[9] Floyd, R. W. Algorithm 97: Shortest path. *Commun. ACM* (1962).

[10] Fortz, B., and Thorup, M. Optimizing ospf/is-is weights in a changing world, 2002.

[11] Ghosh, R., and Garcia-Luna-Aceves, J. Automatic routing using multiple prefix labels. In *Global Communications Conference (GLOBECOM), 2012 IEEE* (Dec 2012).

[12] Ghosh, S. *Distributed Systems: An Algorithmic Approach.* Chapman & Hall/CRC, 2006.

[13] Gross, J., and Yellen, J. *Handbook of Graph Theory.* Discrete Mathematics and Its Applications. Taylor & Francis, 2003.

[14] Hagouel, J., and Schwartz, M. A distributed failsafe route table update algorithm. In *ICDCS* (1982).

[15] Heineman, G., Pollice, G., and Selkow, S. *Algorithms in a Nutshell.* O'Reilly Media, 2008.

[16] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., and Lewin, D. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (1997), ACM, pp. 654–663.

[17] Kideok Cho, Jaeyoung Choi, D.-i. D. K. T. K. Y. C. Content-oriented networking as a future internet infrastructure: Concepts, strengths, and application scenarios.

[18] Kleinrock, L., and Kamoun, F. Hierarchical routing for large networks performance evaluation and optimization. *Computer Networks (1976) 1*, 3 (1977), 155–174.

[19] Krioukov, D., Fall, K., Brady, A., et al. On compact routing for the internet. *ACM SIGCOMM Computer Communication Review 37*, 3 (2007), 41–52.

[20] Legtchenko, S., Monnet, S., Sens, P., and Muller, G. Relaxdht: A churn-resilient replication strategy for peer-to-peer distributed hash-tables. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* (2012).

[21] Lewin, D. M. *Consistent hashing and random trees: Algorithms for caching in distributed networks*. PhD thesis, Massachusetts Institute of Technology, 1998.

[22] Malkin, G. Rfc 2453: Rip version 2. *Request for Comments 2453* (1998).

[23] Matousek, J., and Nesetril, J. *Invitación a la matemática discreta*. Reverte Editorial Sa, 2008.

[24] Mcquillan, J. M., Richer, I. I., Ieee, M., Eric, and Rosen, C. The new routing algorithm for the arpanet. *IEEE Transactions on Communications* (1980).

[25] Medina, A., Lakhina, A., Matta, I., and Byers, J. Brite: An approach to universal topology generation. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on* (2001), IEEE, pp. 346–353.

[26] Moy, J. Open shortest path first routing protocol (version 2).

[27] Muñoz, V. *Aprendiendo a programar paso a paso con C:*. 2012.

[28] Naor, M., and Wieder, U. Novel architectures for p2p applications: The continuous-discrete approach. *ACM Trans. Algorithms* (2007).

[29] Plaxton, C. G., Rajaraman, R., and Richa, A. W. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems* (1999).

[30] Revenga, J. *Flujo en Redes y Gestión de Proyectos. Teoría y Ejercicios Resueltos*. Netbiblo, 2008.

[31] Rowstron, A., and Druschel, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IN: MIDDLEWARE* (2001).

[32] Rueda, R. *Algoritmos, estructuras de datos y programación orientada a objetos*. Ecoe, 2005.

[33] Rutgers, C. L. H. An introduction to igrp. *url: http://www. cisco. com/en/US/tech/tk365/technologies white paper091 86a008000c8ael. shtml* (1991).

[34] Sampath, D., and Garcia-Luna-Aceves, J. Prose: scalable routing in manets using prefix labels and distributed hashing. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on* (2009), IEEE, pp. 1–9.

[35] Santoro, N., and Khatib, R. Labelling and implicit routing in networks. *Computer Journal* (1985).

[36] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: a scalable peer-to-peer lookup protocol for internet applications. *Networking, IEEE/ACM Transactions on 11*, 1 (2003), 17–32.

[37] Tanenbaum, A. S. *Redes de Computadoras.* 2003.

[38] Tomasik, J., and Weisser, M. ashiip: autonomous generator of random internet-like topologies with inter-domain hierarchy. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on* (2010), IEEE, pp. 388–390.

[39] Vázquez, P., Baeza, J., and Herías, F. *Redes y transmisión de datos.* Universitat dÁlacant (Publicacions), 2010.

[40] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications* (2004).

[41] Zhao, B. Y., Kubiatowicz, J., Joseph, A. D., et al. Tapestry: An infrastructure for fault-tolerant wide-area location and routing.