INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

“ENRUTAMIENTO COMPACTO EN REDES ORIENTADAS A CONTENIDO”

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

URBELINO ANGEL HERRERA LÓPEZ

DIRECTORES DE TESIS: DR. JOSÉ GIOVANNI GUZMÁN LUGO

DR. ROLANDO MENCHACA MÉNDEZ

MÉXICO, D.F. DICIEMBRE, 2012

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 06 del mes de diciembre de 2012 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 en redes orientadas a contenido"

Presentada por el alumno:

HERRERA	LÓPEZ	URBELINO ANGEL
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	0	1	6	5	4

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

Director de Tesis

Dr. José Giovanni Guzmán Lugo

Dr. Rolando Menchaca Méndez

Dra. Nareli Cruz Cortés

Dr. Rolando Quintero Téllez

M. en C. Germán Téllez Castillo

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 el día 07 del mes de Diciembre del año 2012, el (la) que suscribe Urbelino Angel Herrera López alumno (a) del Programa de Maestría en Ciencias de la Computación con número de registro B101654, adscrito a Comunicaciones y Redes de Computadoras, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. José Giovanni Guzmán Lugo y Dr. Rolando Menchaca Méndez y cede los derechos del trabajo titulado Enrutamiento Compacto en Redes Orientadas a Contenido, 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 director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección urbelino.herrera@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Nombre y firma

Resumen

En el Internet actual, el subsistema de enrutamiento está completamente desacoplado del contenido alojado en los servidores. Lo anterior es inherentemente ineficiente ya que impone el uso de otros subsistemas que se encargan de asociar los objetos de datos con las máquinas que los alojan. Por otro lado, en el enrutamiento orientado a contenido se reconoce que los usuarios están interesados en los objetos de datos y no en las máquinas en las que están almacenados. En este nuevo paradigma de enrutamiento, las rutas se establecen hacía el contenido y no hacia una máquina en particular.

En esta tesis se presenta el diseño, implementación y caracterización experimental de un algoritmo de enrutamiento orientado a contenido que está basado en etiquetas. Dichas etiquetas son asignadas a los nodos en función a la topología de la red de forma tal que pueden ser usadas para realizar enrutamiento salto a salto. Para evaluar el desempeño del algoritmo se utiliza el porcentaje de paquetes entregados, la sobrecarga y el retardo de extremo a extremo.

Abstract

In current Internet, the routing subsystem is completely decoupled of the content stored at the end-systems. This is inherently inefficient because it imposes the use of other subsystems in charge of establishing associations between the data objects and the machines that store them. On the other hand, content oriented routing acknowledges that users are interested in the data objects but not in the address of the computer that stores them. In this new paradigm to routing, instead of computing routes between machines, routes are established towards the content (data objects).

In this thesis, we present the design, implementation and experimental characterization of an content oriented routing algorithm which is based on labels. These labels are assigned to nodes in the network based on the topology of the entire net in such a way that can be used to perform hoy-by-hop routing in the label space. In order to evaluate the performance of our protocol, we use packet delivery ratio, overhead and end-to-end delay as performance metrics.

Agradecimientos

En primera instancia quiero agradecer al Instituto Politécnico Nacional por permitir que pudiera desarrollar todos mis estudios académicos desde el nivel medio superior hasta mis estudios de posgrado. Así también quiero agradecer al CIC por las facilidades permitidas y por permitir que realizara mis estudios de posgrado en esta institución.

Un especial agradecimiento a mi familia, sobre todo a mi hermana y a esa persona quien ha estado motivándome a desarrollar y terminar muchos de mis proyectos. También agradezco a mis compañeros de generación, de laboratorio y del CIC, siendo alguno de ellos con quien desarrollé una gran amistad y compañerismo en esta etapa de mi vida.

Así también agradezco a todas las personas que han estado apoyándome a desarrollar este proyecto y a revisarlo sin importar las circunstancias.

This material is based upon work supported by a grant from the University of California Institute for Mexico and the United States (UC MEXUS) and the Consejo Nacional de Ciencia y Tecnología de México (CONACyT).

Contenido

Capítulo 1 Introducción 10
1.1 Antecedentes 10
1.2 Planteamiento del problema 12
1.3 Objetivos 13
1.3.1 Objetivo general 13
1.3.2 Objetivos específicos 14
1.4 Justificación 14
1.5 Beneficios esperados 14
1.6 Organización de la tesis 15
Capítulo 2 Marco Teórico 16
2.1 Modelo de referencia de red 16
2.1.1 Modelo de referencia OSI-ISO 17
2.2 Capa de red 17
2.2.1 Aspectos de diseño de la capa de red 17
2.2.1.1 Algoritmos de enrutamiento 18
2.2.1.1.1 Clasificación 19
2.2.1.2 Semántica de entrega 19
2.2.1.3 Tabla de enrutamiento 20
2.2.1.3.1 Tablas de enrutamiento compacto 21
2.3 Red superpuesta 23
2.3.1 Red superpuesta sobre la Internet 23
2.3.2 Redes Peer-to-Peer (*P2P*) 24
2.3.2.1 Clasificación por grado de centralización 25
2.3.2.1.1 Centralizadas 25
2.3.2.1.2 Híbridas, semicentralizadas, mixtas o no estructuradas 25
2.3.2.1.3 Puras, totalmente no centralizadas o estructuradas. 26
2.4 Tablas hash distribuidas (DHT) 27
2.4.1 Propiedades 27

2.4.2 Estructura 28
2.4.3 Parámetros de medida 28
2.5 Usos actuales del Internet 29
2.6 Redes orientadas a contenido 31
2.6.1 Características 32
2.6.2 Clasificación 32
2.6.2.1 Nombrar al Contenido 33
2.6.2.1.1 Jerárquico 33
2.6.2.1.2 Plano 33
2.6.2.1.3 Basado en atributos 34
2.6.2.2 Localización, entrega y difusión del Contenido (Enrutamiento) 34
2.6.2.2.1 No estructurado 34
2.6.2.2.2 Estructurado 34
2.6.2.3 Elaboración del Caché del contenido 35
2.6.3 Algoritmos de enrutamiento 35
2.6.3.1 Chord 35
2.6.3.1.1 Modelo de sistema 36
2.6.3.1.2 Protocolo Chord 37
2.6.3.2 Tapestry 46
2.6.3.2.1 Conceptos básicos 47
2.6.3.2.2 Localización y enrutamiento 50
2.6.3.2.3 Nodos dinámicos 52
2.6.3.3 Resumen 53
2.7 Resumen 54
Capítulo 3 Diseño 55
3.1 AIR 55
3.1.1 Descripción general 56
3.1.2 Modelo de red 57
3.1.3 Estructura de las etiquetas por prefijos 57
3.1.4 Enrutamiento 59
3.1.5 Topología dinámica 61
3.1.6 Directorio de Servicio Distribuido 63

3.2 Modelo 63
Capítulo 4 Análisis Experimental 65
4.1 Propiedades 65
4.2 Pruebas 65
4.2.1 Métricas 66
4.2.2 Escenarios 67
4.2.3 Simulaciones 68
4.2.4 Paquetes entregados 68
4.2.5 Retardo extremo a extremo 69
4.2.6 Sobrecarga de paquetes de control 71
4.2.7 Características de la propuesta 72
4.2.7.1 Promedio de respuesta de ruta entre la solicitud de rutas 72
4.2.7.2 Promedio de respuesta de ruta entre la solicitud de rutas (contemplando la perdida de paquetes) 73
4.2.7.3 Saltos en las rutas generadas 74
4.2.7.4 Número de vecinos de un nodo 75
4.2.7.5 Solicitudes de ser hijo hacia un nodo 76
4.2.8 Comparativa. AODV con un servidor de nombres (dns) 78
4.2.8.1 Paquetes entregados 78
4.2.8.2 Retardo extremo - extremo 79
4.2.8.3 Sobrecarga de paquetes de control 81
4.2.8.4 Promedio de respuesta de ruta entre la solicitud de rutas 82
4.2.8.5 Promedio de respuesta de ruta entre la solicitud de rutas (contemplando la perdida de paquetes) 83
Capítulo 5 Conclusiones y trabajo a futuro 84
5.1 Logros alcanzados y limitaciones 84
5.2 Aportaciones 85
5.3 Trabajos Futuros 85
Capítulo 6 Bibliografía 87

Capítulo 1 Introducción

El enrutamiento en redes de computadoras es parte esencial para el funcionamiento de las redes de redes, sin éste, no sería posible encaminar información (llamada paquetes), de un origen a un destino. Para dicho propósito, se han desarrollado diferentes modelos de cómo resolver los problemas que se enfrentan al diseñarlos e implementarlos, con esto surgieron y se entablaron varias tecnologías enfocándose a comunicaciones punto a punto, siendo TCP/IP (Protocolo de Control de Transferencia, TCP y Protocolo de Internet, IP) la que más auge ha tenido [**1**]. Dada la masificación del uso del Internet y de sus contenidos, se llegan a presentar problemas con este tipo de implementaciones, tales como la movilidad de la información, la autentificación de los usuarios y proveedores de servicios, así como satisfacer las demandas de recursos que todo esto implica [**1**].

En esta tesis se plantea un protocolo de enrutamiento basado en contenido, el cual será presentado a lo largo de este y subsecuentes capítulos. Cabe mencionar que el tema en el que se basa la propuesta de la tesis ha sido muy poco estudiado en la comunidad científica, por lo que tiene diversas áreas de investigación, desarrollo y aplicación.

En el presente capítulo se plantea una idea general de la propuesta del proyecto, empezando por describir los antecedentes del tema, para después plantear el problema que se está tratando. De la misma forma, se describen los objetivos, la justificación, los logros esperados, las limitantes de la tesis, finalizando con la forma en cómo está organizado el documento.

1.1 Antecedentes

La Internet es una red de redes, que utiliza ciertos protocolos comunes y provee servicios, siendo una de sus principales características que no es controlada por alguien o alguna institución [**1**]. Actualmente, comunicarse con un conocido o familiar utilizando la Internet, a través de correos electrónicos, redes sociales, entre otros servicios es algo muy común, pero para el desarrollo de esto, hubo toda una evolución de protocolos y servicios que nos llevan a conocer lo que es la Internet de hoy en día.

En un principio, a partir de 1950, las ideas del Internet estaban emergiendo como tecnología militar, basándose en la arquitectura de la telefonía, y posteriormente desarrollando protocolos y modelos como lo son el conjunto de protocolos TCP/IP [**2**] y el Modelo OSI [**3**] respectivamente, siendo la ARPANET, *Advanced Research Projects Agency Network*, una de las primeras redes de computadoras, la cual estuvo activa desde 1962 hasta 1990 y orientada a un uso no comercial, es decir, que era utilizado con fines académicos, científicos y gubernamentales.

Cabe mencionar que ARPANET fue una red de investigación patrocinada por el Departamento de Defensa de Estados Unidos de América. Con el tiempo se fueron conectando cientos de universidades, además de las instalaciones del gobierno, utilizando líneas telefónicas arrendadas. Cuando se añadieron las comunicaciones por satélite y las redes de radio, los protocolos existentes tuvieron problemas para interoperar entre sí, por lo que surgió una nueva arquitectura de referencia que sería conocida como el modelo de referencia TCP / IP [**1**].

Uno de los principales problemas del Internet es que en un principio se desarrolló de acuerdo al conocimiento, las necesidades y los recursos que se contaban en ese entonces, tomando como referencia las comunicaciones telefónicas, estableciendo la comunicación punto a punto entre dos entidades, a lo que después con la mejora y establecimientos de estándares, el protocolo fue denominado como TCP/IP. Hoy en día las necesidades y recursos han cambiado, como por ejemplo de 50 kbit/seg (una de las primeras velocidades de transmisión de la ARPANET) a los Gigabits/seg que se han llegado a transmitir hoy en día. También se debe hacer énfasis en que el número de usuarios se ha incrementado considerablemente, ya que antes existían pocos, en comparación de los millones que se cuentan actualmente, además de ser necesario establecer la comunicación entre la IP del nodo destino y la IP del nodo fuente, así mismo, hoy en día se emplean servicios de nombres para poder asociar las direcciones IP y tener un fácil acceso al contenido.

De acuerdo a los usos del Internet, así como de la proyección de éstos [**4**] [**5**], los siguientes puntos suelen tener una prioridad alta para cumplir con los requerimientos de tales usos:

- Transmisión en vivo de audio/video de alta definición.
- Fiabilidad de los usuarios así como de los proveedores de servicios para poder realizar operaciones que traten con información sensible, tal como son los servicios financieros.
- Compartición y creación continúa de información que se mantiene en la red.

Como fue descrito en párrafos anteriores, las necesidades han evolucionado y a pesar de que IP se creó para comunicaciones punto a punto, ha sido empleada en los últimos años como un protocolo de distribución de contenido, lo cual conlleva a un bajo rendimiento del Internet hoy en día, ya que éste no es adecuado para infraestructuras multicast / broadcast. Lo anterior debido a cuestiones como son la asignación de direcciones de multidifusión y a una compleja gestión de grupos, llevando a una implementación limitada y compleja para marcos de trabajo multicast, por lo que son costosas (en términos de retardo de la información) o ineficientes [**6**], ejemplos representativos de ello son:

- Caché de páginas web y redes de entrega de contenido (*Content Delivery Networks*, CDN), los cuales redireccionan transparentemente a los clientes del servicio web a una copia más cercana del contenido.
- Sistemas P2P que permiten a los usuarios buscar e intercambiar archivos de contenido.

De acuerdo a Jaeyoung, et. al, [**6**], los cambios en el uso del internet han creado un conjunto de nuevas oportunidades:

- Desacoplar el contenido del host (o de su ubicación), en la capa de red, con esto se mitigan problemas presentes en la Internet, tales como movilidad y seguridad.
- Remover el modelo de seguridad IP, para cambiarlo a uno que ofrezca seguridad *end to end*, esto significa que se debe tener certeza de que el usuario está firmando digitalmente su contenido. Lo anterior se debe a que si se logra encontrar un mecanismo efectivo, automático y transparente que implemente y maneje la seguridad del contenido, sería una mejor aproximación a la seguridad de la comunicación en conjunto con la seguridad de los datos.
- Liberar a los desarrolladores de la necesidad de reinventar aplicaciones y servicios para mecanismos de entrega más seguros.
- Proveer mayor escalabilidad y eficiencia en la entrega de solicitudes de contenido (esto es soportar multicast, broadcast/anycast de forma natural).

Para resolver los problemas anteriormente mencionados se han trabajado diferentes modelos y estudios para rediseñar la arquitectura del Internet, como es el enfoque de redes orientadas a contenido, el cual es la base para el desarrollo de esta tesis. Es importante destacar que las redes orientadas a contenido son un concepto que tiene poco tiempo de desarrollo ya que las primeras menciones de este tipo de redes datan de hace un poco más de 10 años.

1.2 Planteamiento del problema

A finales de los años 60s y principios de los 70s, cuando las ideas fundamentales sobre las cuales subyacer Internet fueron desarrolladas, la telefonía era el único ejemplo de una red de comunicaciones a gran escala, efectiva y exitosa. En este sentido y aunque la solución de comunicación ofrecida por TCP/IP era única y novedosa, el problema que resolvía era en esencia el de la telefonía, es decir, proporcionaba soporte a conversaciones punto a punto entre dos entidades.

El mundo de las comunicaciones, así como el de las aplicaciones que hacen uso de ellas, ha cambiado dramáticamente desde entonces, por lo que los supuestos sobre los que Internet fue desarrollado ya no son completamente válidos.

Aun cuando el protocolo de Internet (IP) ha excedido todas las expectativas iniciales, fue diseñado primordialmente para soportar conversaciones entre dos puntos, por lo que no es particularmente eficiente como medio para distribuir contenido, aplicación que por mucho domina el tráfico de Internet en el presente.

La naturaleza punto a punto de Internet se ve reflejada en su funcionamiento basado en datagramas: Los datagramas IP únicamente pueden especificar puntos finales de comunicación, es decir, las direcciones IP de la fuente y del destino. Sin embargo, el proceso de obtener la dirección

IP que aloja un objeto de datos en particular requiere que se realicen múltiples viajes a través de la red.

La ineficiencia relacionada con los múltiples viajes a través de Internet para poder acceder a un objeto de datos impone severas restricciones a la escalabilidad de la red. Lo anterior, aunado a la problemática relacionada con los servidores centralizados que implementan tanto los servicios de resolución de nombres como de búsquedas, nos lleva a considerar que es necesario replantearnos la forma en cómo debe funcionar Internet. En concreto, consideramos que el rediseño de Internet debe ser guiado por los siguientes principios:

- **Redes orientadas a contenido**. Para los usuarios de Internet (sistemas de cómputo como humanos) el principal interés está en la información y no en las máquinas que la contienen. Por lo tanto, tiene más sentido trabajar sobre una red donde los algoritmos y las arquitecturas se diseñen teniendo a los datos, y no a las computadoras, como parte central de la red.

- **Escalabilidad**. Los servicios que soportan Internet deben estar diseñados para acomodar al número creciente de dispositivos conectados a la red. Esto exige que el estado que debe mantener un dispositivo para poder participar en tareas de enrutamiento crezca de forma sub-lineal con respecto al crecimiento del número de nodos conectados a la red. Adicionalmente, los servicios como los de resolución de nombres (DNS) no deben estar implementados por servidores centralizados ya que se convierten en un cuello de botella al desempeño de la red.

Una desventaja importante de los sistemas centralizados con respecto a sus contrapartes distribuidos, es que los sistemas centralizados presentan puntos únicos de falla, es decir, que su funcionamiento depende de funciones localizadas en una serie predefinida de servidores. En este sentido, es importante que el Internet del futuro sea completamente tolerante a fallas y que su funcionamiento no dependa de algunos nodos especiales. En otras palabras, las funciones de enrutamiento y resolución de nombres deben ser soportadas por la totalidad de los nodos.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar e implementar un protocolo de enrutamiento orientado a contenido cuyas tablas de enrutamiento crezcan de manera sublineal con respecto al número de nodos en la red.

1.3.2 Objetivos específicos

- Diseñar e implementar un protocolo distribuido de elección de líder que no requiera ningún tipo de conocimiento previo acerca de la topología de la red.
- Diseñar e implementar un protocolo distribuido de etiquetado topológico. El etiquetado debe reflejar la relación padre-hijo en un árbol con raíz inducido sobre el grafo que describe a la red.
- Demostrar que las tablas enrutamiento son proporcionales al grado máximo de la red.
- Demostrar que la relación entre los caminos encontrados por el algoritmo propuesto y los caminos más cortos existentes en la red (*stretch*) es del orden O(log n) donde *n* es el número de nodos que componen la red.
- Desarrollar una serie de experimentos basados en simulaciones realistas para caracterizar el desempeño del algoritmo propuesto por medio de su porcentaje de paquetes entregados, la sobrecarga de control inducida y el retardo extremo a extremo que experimentan los paquetes de datos.

1.4 Justificación

En un principio, el desarrollo de Internet se basó en la arquitectura de comunicaciones telefónicas, esto es, a través de comunicaciones punto a punto. Sin embargo, dada la evolución de la Internet y al ser un medio que cada vez es más accesible para las personas, ha sobrepasado sus funciones originales reflejando sus limitantes en nuestros días, entre las que se encuentra un pobre desempeño en infraestructuras multicast o broadcast, así como presentar problemas respecto a la seguridad en la autentificación de los proveedores y usuarios. Es importante considerar la tendencia del uso de la Internet actualmente, ya que como se refleja en varios estudios [**4**] [**5**] [**7**] el uso de video en vivo (*streaming*) y la compartición de archivos, son servicios que van incrementando la demanda de los recursos, por lo que es importante proponer modelos que traten de solventar estas circunstancias además de permitir una compatibilidad con las infraestructuras existentes para que puedan ser implementados.

1.5 Beneficios esperados

Los beneficios esperados del presente trabajo de tesis son: proporcionar un modelo que reduzca los cuellos de botella del Internet, así como permitir una mejora en la utilización de los recursos de la red, principalmente por la omisión de un servidor de nombres, tal como actualmente lo necesita la Internet. Otro punto importante es el manejo de seguridad de forma nata, ya que se podrían

autentificar a los usuarios y a los proveedores de forma sencilla, además de permitir que exista movilidad por parte de todos los integrantes de la red sin afectar la estructura de la red.

1.6 Organización de la tesis

La organización de la tesis es la siguiente; se cuenta inicialmente con un marco teórico basado en una investigación acerca de los modelos de redes actuales, incluyendo las orientadas a contenido, con algoritmos de etiquetado y tablas de enrutamiento, además de su historia, usos, ventajas y desventajas.

Posteriormente se incluyen la sección para el diseño de la propuesta de solución para el enrutamiento de redes orientadas a contenido con un análisis experimental y pruebas del mismo.

Se incluyen un apartado para las conclusiones, logros, aportaciones y reflexiones de trabajo a futuro.

Finalmente se agrega una sección para las referencias utilizadas en la investigación y desarrollo de esta tesis.

Capítulo 2 Marco Teórico

El estudio de enrutamiento en redes de computadoras ha sido trabajado desde la creación de las primeras redes, existiendo desde entonces diferentes algoritmos y propuestas para satisfacer las diferentes necesidades de los usuarios. En este capítulo se describen los temas para poder profundizar e identificar la razón del porqué realizar el enrutamiento orientado a contenido y cambiar la manera en cómo se realiza el enrutamiento en las redes. Asimismo se presenta la forma en cómo se trabaja actualmente, los problemas que existen con el modelo, y las soluciones que se han llegado a presentar ocupando el enrutamiento orientado a contenido.

2.1 Modelo de referencia de red

Las primeras redes de computadoras fueron diseñadas con base en el hardware, quedando el software en segundo plano [**1**], y dada la proliferación del crecimiento de las redes, empezaron a surgir problemas con la comunicación entre las diferentes redes que había. Debido a que no existía una homogenización, se empezaron a desarrollar y extenderse diferentes protocolos de comunicación, surgiendo varias propuestas para cumplir con dicho objetivo, siendo los siguientes modelos los que tuvieron una mayor aceptación [**1**]:

- Modelo de Referencia OSI ISO, también llamado Modelo de Interconexión de Sistemas Abiertos [**3**]. Este modelo se basa en una propuesta elaborada por la Organización Internacional de Normalización (ISO) como un primer paso hacia la estandarización internacional de los protocolos utilizados en las diversas capas. Se suele llamar como modelo de interconexión de sistemas abiertos debido a que trata de conectar tales sistemas para la comunicación con otros sistemas [**1**].
  El modelo OSI en sí mismo no es una arquitectura de red, ya que no especifica los servicios y protocolos exactos para ser utilizados en cada capa. Simplemente especifica lo que cada capa debe hacer. Sin embargo, ISO también ha elaborado normas para todas las capas, aunque estos no forman parte del modelo de referencia en sí. Cada uno ha sido publicado como estándar internacional por separado [**1**].
- Modelo TCP/IP. Es un conjunto de guías generales para el diseño e implementación de protocolos de red específicos para permitir que una computadora pueda comunicarse en una red. TCP/IP provee conectividad de extremo a extremo especificando como los datos deberían ser formateados, direccionados, transmitidos, enrutados y recibidos por el destinatario [**8**]. Este modelo fue implementado en la ARPANET, y posteriormente en su sucesor, la Internet [**1**].

2.1.1 Modelo de referencia OSI-ISO

El modelo OSI está organizado por varios niveles o capas de la siguiente manera:

- Capa física
- Capa de enlace de datos
- Capa de red
- Capa de transporte
- Capa de sesión
- Capa de presentación
- Capa de aplicación

Siendo la capa de red en la que se tiene un especial interés ya que, como se describe a continuación presenta características interesantes.

2.2 Capa de red

La capa de red se encarga de llevar los paquetes desde el origen hasta el destino. Llegar al destino puede requerir muchos saltos a través de enrutadores intermedios. Esta función ciertamente contrasta con la de la capa de enlace de datos, que sólo tiene la meta de mover tramas de un extremo del cable al otro. Por lo tanto, la capa de red es la capa más baja que maneja la transmisión de extremo a extremo. Para lograr su cometido, la capa de red debe conocer la topología de la subred de comunicación (es decir, el grupo de enrutadores) y elegir las rutas adecuadas a través de ella; también debe tener cuidado al escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores y dejar inactivos a otros [**1**].

2.2.1 Aspectos de diseño de la capa de red

En los siguientes puntos se presentan una descripción de algunos de los problemas que deben enfrentar los diseñadores de la capa de red [**1**]:

- Conmutación de paquetes de almacenamiento y reenvío.
- Servicios proporcionados a la capa de transporte, el cual consta de:
    - Los servicios deben ser independientes de la tecnología del enrutador.
    - La capa de transporte debe estar aislada de la cantidad, tipo y topología de los enrutadores presentes.

  - Las direcciones de red disponibles para la capa de transporte deben seguir un plan de numeración uniforme, aún a través de varias LANs y WANs.
- Implementación del servicio no orientado a la conexión. Si se ofrece el servicio no orientado a la conexión, los paquetes se colocan individualmente en la subred y se enrutan de manera independiente. No se necesita una configuración avanzada. En este contexto, por lo general los paquetes se conocen como datagramas (en analogía con los telegramas) y la subred se conoce como subred de datagramas.
- Implementación del servicio orientado a la conexión. Si se utiliza el servicio orientado a la conexión, antes de poder enviar cualquier paquete de datos, es necesario establecer una ruta del enrutador de origen al de destino. Esta conexión se conoce como CV (circuito virtual), en analogía con los circuitos físicos establecidos por el sistema telefónico, y la subred se conoce como sub-red de circuitos virtuales.

2.2.1.1 Algoritmos de enrutamiento

La función principal de la capa de red es enrutar paquetes de la máquina origen a la máquina destino. En la mayoría de las subredes, los paquetes requerirán varios saltos para completar el viaje. Los algoritmos que eligen las rutas y las estructuras de datos que usan constituyen un aspecto principal del diseño de la capa de red.

El algoritmo de enrutamiento es aquella parte del software de la capa de red encargada de decidir la línea de salida por la que se transmitirá un paquete de entrada. Si la subred usa datagramas de manera interna, esta decisión debe tomarse cada vez que llega un paquete de datos, dado que la mejor ruta podría haber cambiado desde la última vez. Si la subred usa circuitos virtuales internamente, las decisiones de enrutamiento se toman sólo al establecerse un circuito virtual nuevo. En lo sucesivo, los paquetes de datos simplemente siguen la ruta previamente establecida. Este último caso a veces se llama enrutamiento de sesión, dado que una ruta permanece vigente durante toda la sesión de usuario (por ejemplo, durante una sesión desde una terminal, o durante una transferencia de archivos).

Algunas veces es útil distinguir entre el enrutamiento, que es el proceso que consiste en tomar la decisión de cuáles rutas utilizar, y el reenvío, que consiste en la acción que se toma cuando llega un paquete. Se puede considerar que un enrutador realiza dos procesos internos. Uno de ellos maneja cada paquete conforme llega, buscando en las tablas de enrutamiento la línea de salida por la cual se enviará; este proceso se conoce como reenvío. El otro proceso es responsable de llenar y actualizar las tablas de enrutamiento, es ahí donde entra en acción el algoritmo de enrutamiento.

Sin importar si las rutas para cada paquete se eligen de manera independiente o sólo cuando se establecen nuevas conexiones, hay ciertas propiedades que todo algoritmo de enrutamiento debe poseer: exactitud, sencillez, robustez, estabilidad, equidad y optimización. La exactitud y la

sencillez requieren pocos comentarios, pero la necesidad de robustez puede ser menos obvia a primera vista. Una vez que una red principal entra en operación, cabría esperar que funcionara continuamente durante años sin fallas a nivel de sistema. Durante ese periodo habrá fallas de hardware y de software de todo tipo. Los hosts, los enrutadores y las líneas fallarán en forma repetida y la topología cambiará muchas veces. El algoritmo de enrutamiento debe ser capaz de manejar los cambios de topología y tráfico sin requerir el aborto de todas las actividades en todos los hosts y el reinicio de la red con cada caída de un enrutador. Un algoritmo estable alcanza el equilibrio y lo conserva. La equidad y la optimización pueden parecer algo obvias, pero se requiere un punto medio entre la eficiencia global y la equidad hacia las conexiones individuales aunque se debe decidir qué es lo que busca optimizar (minimización del retardo medio de los paquetes, pero también lo es el aumento al máximo de la velocidad real de transporte de la red).

2.2.1.1.1 Clasificación

Los algoritmos de enrutamiento pueden agruparse en dos clases principales: no adaptativos y adaptativos.

- Los algoritmos no adaptativos no basan sus decisiones de enrutamiento en mediciones o estimaciones del tráfico y la topología actuales. En cambio, la decisión de la ruta que se usará para llegar de un nodo I a un nodo J se toma por adelantado, fuera de línea, y se carga en los enrutadores al arrancar la red. Este procedimiento se conoce como enrutamiento estático.
- En contraste, los algoritmos adaptativos cambian sus decisiones de enrutamiento para reflejar los cambios de topología y, por lo general también el tráfico. Los algoritmos adaptativos difieren en el lugar de donde obtienen su información (por ejemplo, localmente, de los enrutadores adyacentes o de todos los enrutadores), el momento de cambio de sus rutas (por ejemplo, cada T segundos, cuando cambia la carga o cuando cambia la topología) y la métrica usada para la optimización (por ejemplo, la distancia, el número de saltos o el tiempo estimado de tránsito).

2.2.1.2 Semántica de entrega

Los esquemas de enrutamiento difieren en sus semánticas de entrega, la cual es representada en la Figura 2.1:

- *Unicast* entrega un mensaje a un nodo específico único.
- *Broadcast* envía un mensaje a todos los nodos de la red.

- *Multicast* envía un mensaje a un grupo de nodos que han expresado su interés en recibir el mensaje.
- *Anycast* entrega un mensaje a cualquiera de un grupo de nodos, típicamente el más cercano a la fuente.
- *Geocast* entrega un mensaje a un área geográfica.

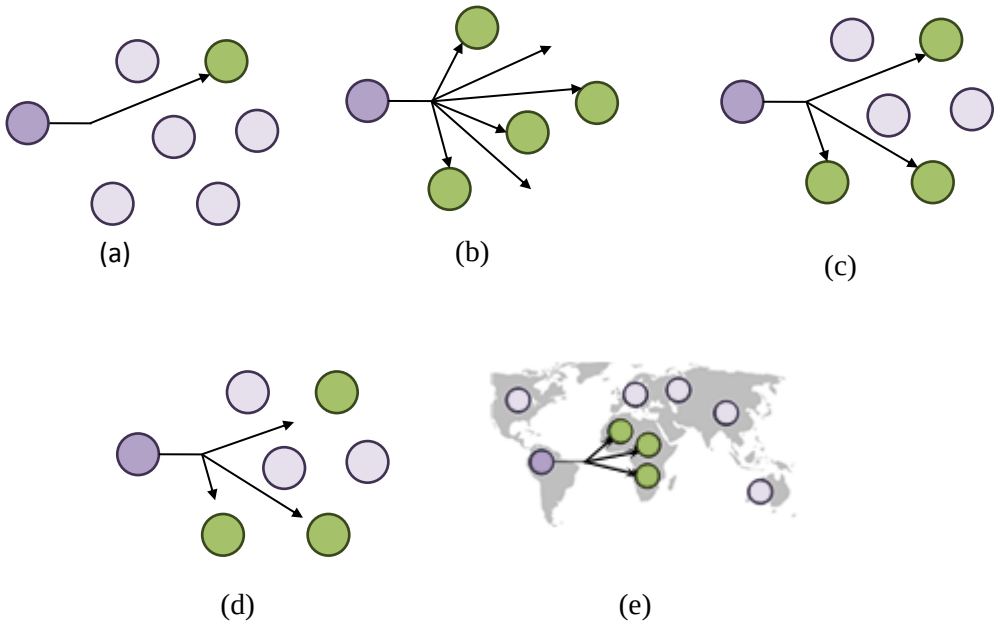


Figura 2.1. Diferentes modos de difusión de paquetes. (a) unicast; (b) broadcast; (c) multicast; (d) anycast; (e) geocast.

2.2.1.3 Tabla de enrutamiento

La tabla de enrutamiento es una tabla de datos almacenada en un router o un nodo en la red que muestra las rutas a los destinos de red particulares, y en algunos casos, alguna otra información adicional asociada con dichas rutas. La tabla de enrutamiento contiene información acerca de la topología de la red inmediatamente alrededor del nodo. La construcción de las tablas de enrutamiento es el principal objetivo de los protocolos de enrutamiento, la cual se realiza utilizando los algoritmos de ruteo basándose en la información que se intercambia entre los vecinos de dicho nodo [**9**] [**10**].

2.2.1.3.1 Tablas de enrutamiento compacto

De acuerdo con [**11**] las tablas de enrutamiento son tradicionalmente tan grandes como el número de destinos n. Esto puede tener altos requerimientos de almacenamiento, así como también realizar las búsquedas en la tabla y procesamiento al reenvío de cada paquete. Si la tabla se puede reorganizar de manera que está indexado por el enlace entrante, y la tabla proporciona el enlace de salida, entonces el tamaño de la tabla se convierte en el grado del nodo, el cual puede ser mucho más pequeño que n.

El factor del *stretch* de un esquema de ruteo r se define por la siguiente ecuación (Ecuación 2.1):

$$\max_{i,j \in N} \left\{ \frac{distancia_r(i,j)}{distancia_{opt}(i,j)} \right\} \tag{2.1}$$

La cual es una medida importante en la evaluación de un compacto esquema de enrutamiento.

2.2.1.3.1.1 Esquemas de enrutamiento

Algunos de los enfoques de diseño compacto de tablas de enrutamiento se describen en las siguientes subsecciones. Todos los métodos que se presentan para el enrutamiento compacto son ricos en problemas algorítmicos de grafos distribuidos, incluyendo la identificación y prueba de límites sobre la eficiencia de las rutas calculadas. El uso de grafos de diferentes topologías proporciona resultados interesantes para los esquemas de enrutamiento que se presentan a continuación.

2.2.1.3.1.1.1 Jerárquico

El gráfico de la red se organiza en grupos de manera jerárquica, donde cada grupo de nodos que tiene un nodo superior designado como *clusterhead* el cual representa el clúster en el siguiente nivel superior en la jerarquía. Existe información detallada acerca del enrutamiento dentro de un grupo, con todos los routers dentro de ese clúster. Si el destino no está en el mismo grupo que la fuente, el paquete se envía al *clusterhead* y superior en la jerarquía, según proceda. Una vez que el *clusterhead* del destino se encuentra en las tablas de enrutamiento, el paquete se envía a través de la red en ese nivel de la jerarquía, y luego hacia abajo de la jerarquía en el grupo de destino. Esta forma de enrutamiento se utiliza ampliamente en la Internet.

2.2.1.3.1.1.2 Etiquetado de árboles

Esta familia de esquemas utiliza una topología de árbol lógico para realizar el enrutamiento. El esquema de enrutamiento requiere el etiquetado de los nodos del grafo de tal manera que todos

los destinos sean alcanzables a través de cualquier enlace y se puedan representar como un rango de direcciones contiguas $[x, y]$. Un nodo con grado deg necesita sólo mantener las entradas deg en su tabla de enrutamiento, donde cada entrada es un rango de direcciones contiguas. Para todos los intervalos de direcciones $[x, y]$, el esquema debe satisfacer la condición $x < y$.

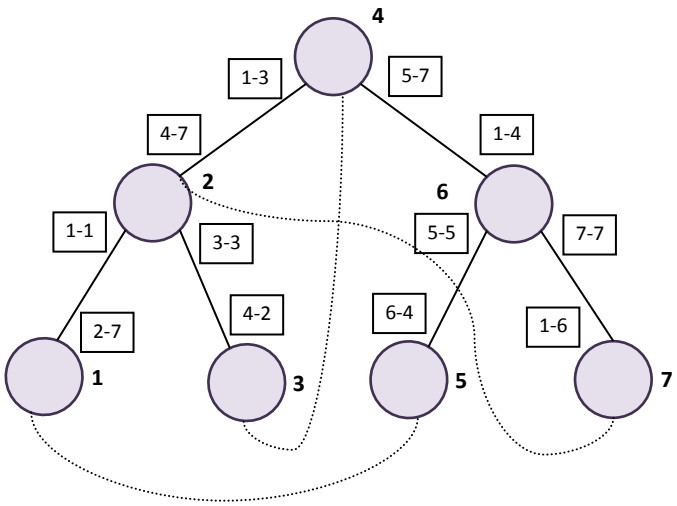


Figura 2.2. Etiquetado de un árbol.

La Figura 2.2 presenta el etiquetado de un árbol con siete nodos. El intervalo de las etiquetas está encerrado en rectángulos; sin intervalos está en líneas de trazos. El etiquetado de árbol puede proporcionar un gran ahorro, en comparación con una tabla de tamaño n en cada nodo. Desafortunadamente, todo el tráfico se limita a los intervalos de árbol lógicos.

2.2.1.3.1.1.3 Intervalos

Los sistemas de etiquetado de árboles sufren del hecho de que los datos pueden enviarse sólo sobre el rango de valores de las etiquetas de lo árboles, desperdiciando el ancho de banda restante en el sistema. El intervalo de enrutamiento se extiende del etiquetado de árboles de modo que los paquetes de datos no necesitan ser enviados sólo en los intervalos de etiqueta en un árbol.

Formalmente, dado un grafo (N, L), un esquema de enrutamiento por intervalo es una tupla (β, γ), donde:

- Etiquetado de nodo. β es un mapeo 1:1 en N, el cual asigna etiquetas a los nodos.
- Etiquetado de rango. El mapeo γ en las etiquetas con un rango L para algún subconjunto de etiquetas de nodo $\beta(N)$ tal que para cualquier nodo x, todos los destinos están

cubiertos $\left(\cup_{y \in Neighbors}\ \gamma\ (x,y)\ \cup \beta(x) = N\right)$ y no existe duplicidad de la cobertura $(\gamma(x,w) \cap \gamma(x,y) = \emptyset \text{ para } w,y \in Neighbors)$.

- Para cualquier nodo fuente s y destino t, debe existir una secuencia de nodos $\langle s = x_0, x_1 \ldots x_{k-1}, x_k = t\rangle$ donde $\beta(t) \in \gamma(x_{i-1}, x_i)$ para cada i entre 1 y k. Por lo tanto, para cada par fuente y destino, debe de existir una ruta de acceso en el nuevo mapeo.

Para mostrar que un sistema de etiquetado intervalo es posible para cada grafo, un árbol con la siguiente propiedad se construye: "no hay intervalos transversales en el gráfico correspondiente." El árbol generado por un recorrido en profundidad siempre satisface esta propiedad. Los nodos están etiquetados por un recorrido en orden previo, mientras que los intervalos están marcados por un esquema más detallado. Ejemplo de ello es el algoritmo de Tapestry presentado en 2.6.3.2.

2.2.1.3.1.1.4 Prefijos

El enrutamiento por prefijo supera los inconvenientes de los intervalos de enrutamiento. En el enrutamiento por prefijos, las etiquetas de los nodos, así como las etiquetas de los objetos son vistas como cadenas. El procedimiento de enrutamiento en un nodo es el siguiente: identificar en su tabla de ruteo las etiquetas almacenadas y determinar aquella cuya etiqueta es el prefijo más largo de la dirección del destino. Este es el registro en el que el paquete se enrutará para ese destino en particular.

Una de las principales características de este esquema es que muchas veces se emplean pequeñas tablas de ruteo, pero con un costo de emplear distancias más largas (en comparación con la ruta óptima) [**12**]. Un ejemplo de este esquema es AIR, que se describe en el apartado 3.1.

2.3 Red superpuesta

Una red superpuesta (*overlay network*) es una red virtual integrada por nodos y enlaces lógicos que es construida sobre una red existente con el propósito de implementar un servicio de red que no está disponible en la red existente [**13**].

2.3.1 Red superpuesta sobre la Internet

Internet es una red superpuesta cuyo objetivo es la conexión de redes de área local y otro tipo de redes. Para su construcción utiliza las redes subyacentes, una serie de líneas de comunicación (como por ejemplo líneas de teléfono o líneas de fibra óptica) y el protocolo IP (nueva capa sobre los mensajes que circulan en las redes subyacentes) [**14**].

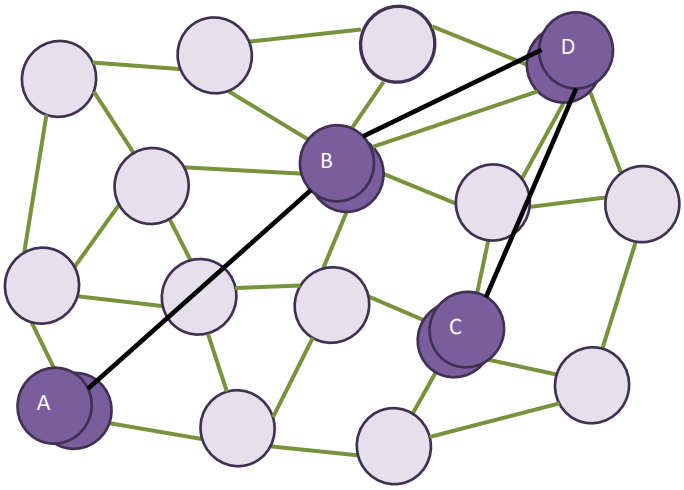


Figura 2.3. Ejemplo de una red superpuesta.

Hoy en día la Internet es la base para muchas redes superpuestas que pueden ser construidas con el fin de permitir mensajes de ruteo que no son especificados por las comunicaciones IP, así como mensajes broadcast y obtención de información de una manera diferente a como se haría si se utilizara el protocolo de IP. Ejemplo de estos son los sistemas P2P, las transmisiones broadcast, entre otros.

Un ejemplo gráfico de como es esta estructura puede observarse en la Figura 2.3, donde se presenta una red superpuesta, sobre una red.

2.3.2 Redes Peer-to-Peer (*P2P*)

Una red *peer-to-peer* también es llamada una red de pares, o una red entre pares o una red punto a punto (*P2P*, por sus siglas en inglés), la cual es una red de computadoras en la que todos o casi todos los nodos (computadoras) involucrados se comportan como iguales entre ellos (es decir no existen sólo clientes y servidores), esto es que actúan como clientes y servidores al mismo tiempo [**13**].

Existen varias características que componen a las redes P2P [**13**] [**15**]:

- *Escalabilidad*. Lo deseable es que cuantos más nodos estén conectados a una red P2P, mejor será su funcionamiento, así cuando los nodos llegan y comparten sus propios recursos, los recursos totales del sistema aumentan. Esto es diferente en una arquitectura del modo servidor-cliente con un sistema fijo de servidores, en los cuales la adición de clientes podría significar una transferencia de datos más lenta para todos los usuarios.
- *Robustez*. La naturaleza distribuida de las redes *peer-to-peer* también incrementa la robustez en caso de haber fallos en la réplica excesiva de los datos hacia múltiples

destinos, permitiendo a los *peers* encontrar la información sin hacer peticiones a ningún servidor centralizado de indexado.

- *Descentralización*. Por definición, las redes P2P son descentralizadas y todos los nodos son iguales. No existen nodos con funciones especiales, por lo tanto, ningún nodo es imprescindible para el funcionamiento de la red, aunque en realidad algunas redes P2P no cumplen con esta característica.
- *Distribución de costes entre usuarios*. Se comparten o donan recursos a cambio de recursos según la aplicación.
- *Anonimato*. Es deseable que en estas redes quede anónimo el autor de un contenido, el editor, el lector, el servidor que lo alberga y la petición para encontrarlo, siempre que así lo necesiten los usuarios.
- *Seguridad*. Es una de las características deseables de las redes P2P y también la menos implementada. Los objetivos de un P2P seguro serían identificar y evitar los nodos maliciosos, el contenido infectado, así como evitar el espionaje de las comunicaciones entre nodos, creación de grupos seguros de nodos dentro de la red, protección de los recursos de la red.

2.3.2.1 Clasificación por grado de centralización

De acuerdo al grado de centralización se pueden clasificar las redes P2P de la siguiente manera:

2.3.2.1.1 Centralizadas

Una red con este tipo cumple con las siguientes características:

- Se rige bajo un único servidor, que sirve como punto de enlace entre nodos y como servidor de acceso al contenido, el cual distribuye a petición de los nodos.
- Todas las comunicaciones, tanto las peticiones como los encaminamientos entre nodos, dependen exclusivamente de la existencia del servidor.

Un ejemplo de este tipo de redes es Napster. Ver Figura 2.4 (a).

2.3.2.1.2 Híbridas, semicentralizadas, mixtas o no estructuradas

Este tipo de P2P presenta las siguientes características:

- Tiene un servidor central que guarda información en espera y responde a peticiones para esa información.
- Los nodos son responsables de hospedar la información (pues el servidor central no almacena la información) que permite al servidor central reconocer los recursos que se desean compartir, y para poder descargar esos recursos compartidos a los usuarios que lo solicitan.
- Las terminales de enrutamiento son direcciones usadas por el servidor, que son administradas por un sistema de índices para obtener una dirección absoluta.

Dentro de las desventajas de este tipo de redes se encuentra que:

- Las consultas no siempre serán resueltas.
- No existe correlación entre un peer y el contenido manejado por este, por lo que no se garantiza que inundar la red encontrará el dato deseado.
- La inundación causa una gran cantidad de tráfico de control en la red.

Algunos ejemplos de una red P2P híbrida son BitTorrent y eDonkey. Ver Figura 2.4 (b).

2.3.2.1.3 Puras, totalmente no centralizadas o estructuradas.

Las redes de este tipo tienen las siguientes características:

- Los nodos actúan como cliente y como servidor.
- No existe un servidor central que maneje las conexiones de red.
- No hay un enrutador central que sirva como nodo y administre direcciones.

Dentro de las ventajas de este tipo de redes se puede enlistar lo siguiente:

- Mantenimiento distribuido de la información.
- Implementa funciones hash para asignar valores a cada contenido y a cada peer en la red.
- Usa un protocolo global para determinar el o los *peers* responsables para la información.

Algunos ejemplos de una red P2P pura son Chord [**16**], Tapestry [**17**], entre otros, ver Figura 2.4 (c).

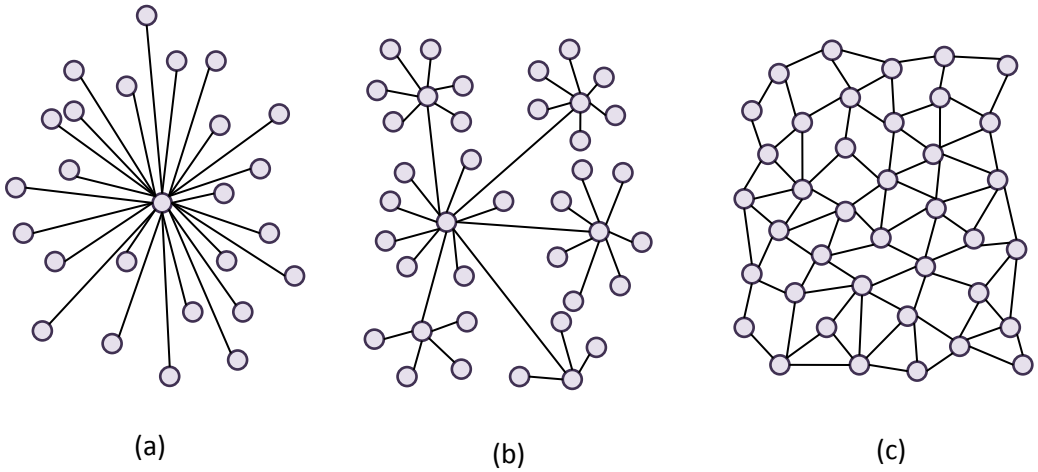


Figura 2.4. Topologías de red. (a) centralizada; (b) mixta; (c) no centralizada.

2.4 Tablas hash distribuidas (DHT)

Las tablas hash distribuidas (Distributed Hash Tables, *DHT* por sus siglas en inglés) son una clase de sistemas distribuidos descentralizados que proveen un servicio de búsqueda similar al de las tablas hash, donde los pares (clave, valor) son almacenados en el *DHT*, y cualquier nodo participante puede recuperar de forma eficiente el valor asociado con una clave dada. La responsabilidad de mantener el mapeo de las claves a los valores está distribuida entre los nodos, de forma que un cambio en el conjunto de participantes causa una interrupción mínima. Lo anterior permite que las *DHTs* puedan escalar a cantidades de nodos extremadamente grandes, y que puedan manejar errores, llegadas y caídas de nodos constantemente [**18**].

Una DHT puede servir como un repositorio de objetos distribuidos, donde se determina la ubicación de un objeto por el valor hash de su nombre. Por ejemplo, las funciones criptográficas de hash como MD5 [**19**] o SHA1 [**20**] crean un mapa de cadenas arbitrarias a 128-bit o 160-bit de los valores hash, respectivamente, los cuales se pueden utilizar para mapear nombres arbitrarios de objeto en $h - bits$ de hash-valores, donde h depende de la función hash que se utiliza [**21**].

Las primeras cuatro *DHTs* CAN [**22**], Chord [**16**], Pastry [**23**] y Tapestry [**17**] surgieron en la misma época durante el año 2001. Desde entonces, esta forma de búsqueda ha sido ampliamente usada, principalmente desde que BitTorrent las incorporó [**18**].

2.4.1 Propiedades

Las *DHTs* destacan las siguientes propiedades:

- *Descentralización*. Los nodos en conjunto forman el sistema sin ningún tipo de coordinación central.
- *Escalabilidad*. El sistema debe funcionar de manera eficiente, incluso con miles o millones de nodos.
- *Tolerancia a fallas*. El sistema debe ser fiable (en cierto sentido), incluso con nodos continuamente uniéndose, retirándose y/o fallando.

Una técnica clave utilizada para lograr estos objetivos es que cualquier nodo debe coordinar con sólo unos pocos nodos en el sistema, la más común $O(\log n)$ de los n participantes, de manera que ante un cambio en la composición sólo es necesaria una cantidad limitada de trabajo.

2.4.2 Estructura

La estructura de una *DHT* puede ser descompuesta en los siguientes tres componentes [**18**]:

- *Espacio de claves abstracto.* Es una serie de números, cadenas, bits que componen al espacio de las claves.
- *Particionamiento del espacio de claves.* Divide entre los nodos el espacio de claves del punto anterior. La mayoría de las *DHTs* utilizan alguna variante de dispersión hash para mapear las claves con los nodos. Esta técnica implementa una función $\delta(k1, k2)$ que define una noción abstracta de la distancia entre la clave $k1$ y $k2$, la cual no está relacionada con la distancia geográfica o la latencia de la red. A cada nodo se le asigna una única clave denominada ID.
- *Red superpuesta.* Cada nodo mantiene una serie de enlaces a otros nodos (sus vecinos o tabla de enrutamiento). En conjunto, estos enlaces forman la red. Un nodo escoge sus vecinos de acuerdo a una estructura específica, llamada topología de la red. Todas las topologías *DHT* comparten alguna variante de la siguiente propiedad: para cualquier clave *k*, pasa que el nodo tiene un ID que posee *k* o tiene un enlace a un nodo cuyo ID es más cercano a *k*, en términos de la distancia definida en el espacio de clave.

2.4.3 Parámetros de medida

De acuerdo a Naor y Wieder [**24**], los parámetros que permiten cuantificar una *DHT* son alguna o algunas de las siguientes métricas:

- *Costo de join / leave*. El servicio debe adaptarse con facilidad a los cambios. Cuando los nodos se unen o abandonan, sólo un pequeño número de nodos deben cambiar su estado. En particular, el grado del grafo que representa la red, debe ser pequeño.

- *Congestión*. Ningún nodo debe ser un cuello de botella en el rendimiento del servicio. La carga incurrida por búsquedas de enrutamiento a través del sistema debe ser distribuida equitativamente entre los nodos participantes.
- *Longitud de ruta de búsqueda*. La ruta de transmisión de una búsqueda debe incluir el menor número de nodos.
- *Tolerancia a fallos*. El servicio debe continuar a pesar de que algunos de los nodos o las conexiones fallan.
- *Almacenamiento en caché dinámico*. Los elementos de datos muy populares pueden causar un cuello de botella en los alrededores de su localidad. Aliviar la congestión alrededor de la zona activa, requiere el servicio para apoyar algunos mecanismos de almacenamiento en caché dinámico, en el que se replica el elemento de datos a otros nodos.

2.5 Usos actuales del Internet

Tal como se mencionó en el apartado 1.1 Antecedentes, los cambios en el uso del Internet han evolucionado de acuerdo a las necesidades y recursos de los usuarios. Para poder saber el alcance así como los usos que está teniendo Internet de forma global, se realizó una búsqueda de información relativa a este, encontrando dos documentos que cumplían con las características buscadas:

- *Cisco: Entering the Zettabyte Era*. Es parte del *Cisco Visual Networking Index* (VNI), una iniciativa en curso para realizar un seguimiento y pronosticar el impacto visual de las aplicaciones de red. El documento presenta algunas de las principales conclusiones del mundial de Cisco, las previsiones de tráfico IP y explora las implicaciones del crecimiento del tráfico IP para proveedores de servicios. [**5**]
- Ipoque Internet Study. Es el proveedor líder europeo de inspección profunda de paquetes (*Deep Packet Inspection*, DPI) para la gestión y el análisis de tráfico de Internet. El estudio incluye datos estadísticos acerca de la popularidad y el comportamiento del usuario para todos los protocolos de red comunes, cubriendo la mayoría de las aplicaciones utilizadas en la Internet actual, como son la navegación web, el streaming, el intercambio de archivos P2P, la mensajería instantánea, la telefonía por Internet y los juegos en línea [**4**].

A pesar de que ambos ofrecen características similares, se escogió el estudio ofrecido por Cisco, ya que los datos empleados son del 2011 [**5**], en comparación con Ipoque, el cual su último estudio data del 2009 [**7**].

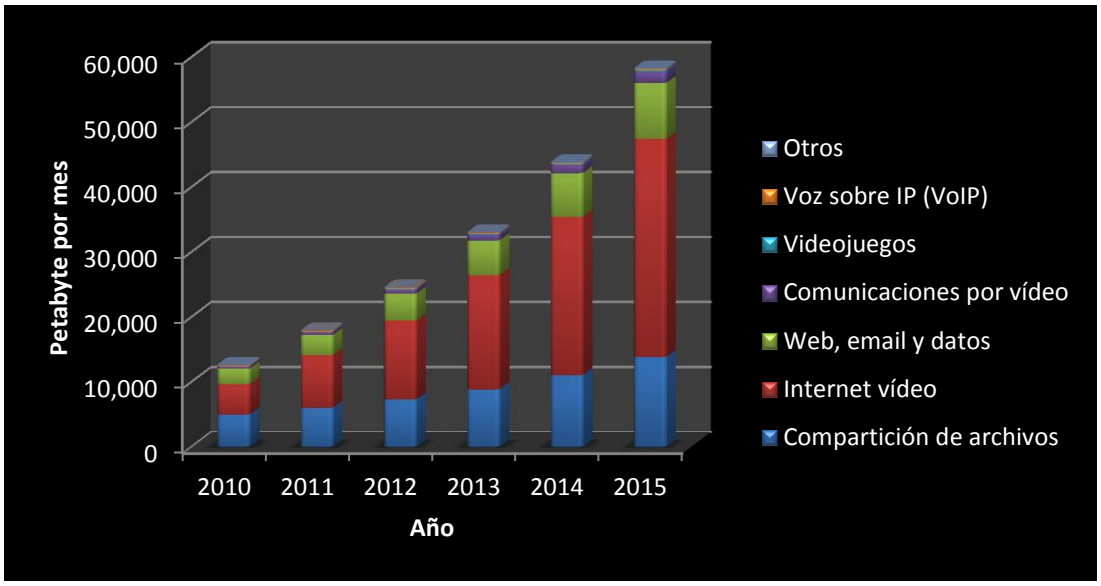
Para el presente documento, es importante citar la Tabla 2.1, la cual en conjunto con la Gráfica 2.1, muestran el comportamiento analizado y pronosticado de Internet para el periodo del 2010-2015. Algunas definiciones de interés de los datos previamente citados son:

- *Web, email, y datos.* Incluye a la web, a los correos electrónicos, a la mensajería instantánea y a otras fuentes de datos (excluyendo la compartición de archivos).
- *Compartición de archivos*. Incluye tráfico peer-to-peer de todos los sistemas reconocidos como P2P como lo es BitTorrent y redes eDonkey, así como tráfico basado en la web para la compartición de archivos.
- *Videojuegos*. Incluye juegos en línea casuales, las redes de los videojuegos de consola y los videojuegos de mundos virtuales.
- *Comunicaciones por video*. Incluye el video sobre Internet de clientes de mensajería, tal como Skype.
- *VoIP.* Incluye el tráfico sobre minoristas de servicios VoIP, así como VoIP basado en computadoras, excluyendo todo lo que involucra al transporte al por mayor de VoIP.
- *Internet video.* Incluye formatos de videos cortos (por ejemplo YouTube), formatos de video largo (Hulu), video sobre Internet en vivo, Internet-video-TV (por ejemplo Netflix a través de Roku), compra y renta de video en línea, accesos y visualización de cámaras web, videovigilancia basada en la web (excluyendo los formatos basados en P2P).
- *Otros.* Cualquier elemento no considerado en las clasificaciones anteriores.

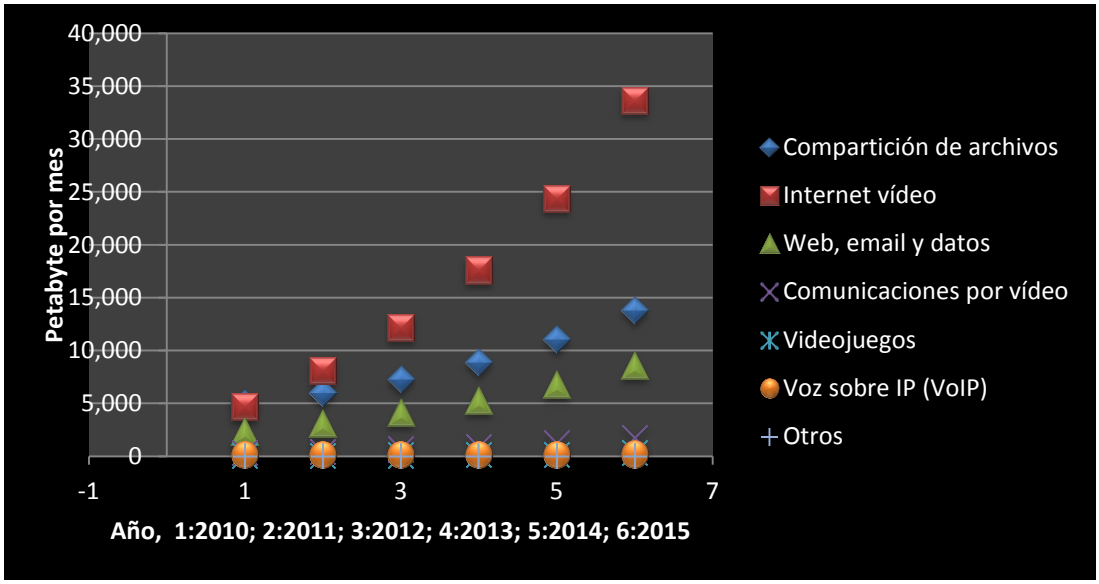
Tabla 2.1. Global Consumer Internet Traffic, 2010 - 2015 (PB per Month)

Clasificación	2010	2011	2012	2013	2014	2015
Compartición de archivos	4,968	6,017	7,277	8,867	11,040	13,797
Internet vídeo	4,672	8,079	12,146	17,583	24,357	33,620
Web, email y datos	2,393	3,113	4,146	5,325	6,769	8,592
Comunicaciones por vídeo	308	442	659	905	1,251	1,736
Videojuegos	49	68	95	133	187	290
Voz sobre IP (VoIP)	138	147	153	157	160	168
Otros	0	1	1	3	8	11

Dentro de las características que interesa destacar de dicha información es la creciente demanda de recursos para internet, video y la compartición de archivos, quienes son los dominantes, tal como se muestra en la Gráfica 2.2. Por lo comentado anteriormente, es importante destacar que los principales usos y necesidades del Internet el protocolo TCP/IP, presenta dificultades a tratar, tal como se explicó en la sección 1.1 Antecedentes.



Gráfica 2.1 Global Consumer Internet Traffic, (2010–2015), apilado por año.



Gráfica 2.2. Global Consumer Internet Traffic, (2010–2015), dispersión anual.

2.6 Redes orientadas a contenido

Las redes orientadas a contenido (*Content-Oriented Networking*, CON por sus siglas en inglés), son identificadas como:

- Content-Centric Networking [**25**] [**26**] [**27**] [**28**] [**29**].
- Content-Oriented Networking [**6**] [**30**] [**31**].
- Content-Based Networking [**32**] [**33**].
- Data-Oriented Networking [**34**] [**35**] [**36**].
- Named Data Networking [**25**].

En los cuales, y de forma general se trata a las redes orientadas a contenido como una alternativa propuesta a la arquitectura actual de las redes de computadoras, siendo su principal base que las comunicaciones en las redes deben permitir al usuario enfocarse al contenido que necesita, en lugar de tener que hacer referencia a una ubicación física de donde se obtienen los datos.

2.6.1 Características

Las redes orientadas a contenido, presentan las siguientes características:

- Una red CON realiza el enrutamiento por su contenido, no por la localización del host, por lo cual:
  - La identificación de los hosts se reemplaza por una identificación del contenido.
  - La localización del contenido es independiente de su nombre.

  Con lo anterior, la localización es independiente del nombre del contenido y el enrutamiento, por lo tanto es libre para su movilidad.
- El modelo base para este tipo de redes, es el paradigma de publicación/suscripción, esto significa que el contenido de una fuente anuncia (publica) un archivo de 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 forma eficiente y escalable.
- La autentificación del contenido puede verificarse fácilmente utilizando una llave criptográfica.

2.6.2 Clasificación

De acuerdo a [**6**] la arquitectura de una CON puede ser caracterizada en tres principales bloques:

- Como nombrar al contenido.
- Como localizar, entregar y difundir el contenido (enrutamiento)
- Como hacer el caché del contenido en la red.

Cada uno de estos bloques se explican a continuación.

2.6.2.1 Nombrar al Contenido

Para poder nombrar al contenido se han desarrollado varias técnicas sobre éste, las cuales son clasificadas en orden jerárquico, plano y basado en atributos.

2.6.2.1.1 Jerárquico

Algunos ejemplos en donde se aplica este formato son en los artículos sugeridos por [**25**] y por [**33**], en donde suelen limitar el nombre de acuerdo a un identificador propuesto dada la URL de la página web, como por ejemplo, se suele decir que ese identificador es el símbolo "/" de las URL (www.ipn.mx/archivos/logo.gif). Lo anterior significa que todo el contenido que es de www.ipn.mx está almacenado en un host para su enrutamiento. Uno de sus principales problemas es la semántica del nombre, por lo que puede llegar a causar confusión dependiendo del área geográfica.

Tabla 2.2. Tabla comparativa de los principales modelos de enrutamiento orientado a contenido, de acuerdo con [6]. Donde *N* se refiere a los nodos, *C* al número de elementos de contenido y *A* el número de atributos en toda la red.

Artículo	Nombre Contenido	Ventajas del Nombrado	Estructura de enrutamiento	Escalabilidad	Mensajes de Control
CCN [**25**]	Jerárquico	Agregable, compatible IP	No estructurado	N(mejor); C(peor)	Alto
DONA [**36**]	Plano	Persistente	Estructurado (árbol)	C	Bajo
PSIRP	Plano	Persistente	Estructurado (jerárquico)	logC	Bajo
TRIAD [**37**]	jerárquico	Agregable, compatible IP	No estructurado	N	Alto
CBCB [**33**]	Pares (atributo, valor)	Búsqueda dentro de la red interna	Basado en la fuente, árbol multicast	2^A	Alto

2.6.2.1.2 Plano

En este formato se emplean nombres planos y en algunos casos de auto-certificación mediante la definición de un identificador del contenido, tal como una llave hash o una llave pública, ejemplo de estos son [**36**] y [**38**]. Una de sus características, es que debido a que se utiliza un nombre

plano, se tiene que implementar un paso intermedio en el proceso, para que se traduzca de ese nombre plano a un texto entendible para que el usuario lo pueda identificar.

2.6.2.1.3 Basado en atributos

Tal como se menciona en [**33**], se puede emplear un par de conjunto de atributos-valores (*attribute-value pairs*, AVP), con lo cual, para el enrutamiento se tiene que verificar esta lista de atributos e identificar la localización. Una de sus principales desventajas es que la semántica puede presentar problemas, dada la localización geográfica, y al ser un par de elementos, los datos tienden a ser amplios.

2.6.2.2 Localización, entrega y difusión del Contenido (Enrutamiento)

De acuerdo con [**6**], existen dos formas de realizar el enrutamiento:

- No estructurado.
- Estructurado.

Los cuales se describen a continuación.

2.6.2.2.1 No estructurado

Esta propuesta define un comportamiento tal como se efectúa en IP, es decir, el anuncio de enrutamiento es realizado por inundación, ya que no existe una estructura para mantener una tabla de ruteo, un ejemplo de lo anterior está dado por [**25**] y [**37**], siendo en el primer artículo citado en donde se implementa un nuevo atributo en las tablas de ruteo para identificar el contenido.

2.6.2.2.2 Estructurado

De acuerdo con [**6**], se han propuesto dos formas:

- Árbol
- Tabla hash

2.6.2.2.2.1 Árbol

El artículo más representativo sobre este tema es tratado en [**36**], en el cual los enrutadores forman un árbol jerárquico y cada enrutador mantiene la información de ruteo de todos los contenidos publicados en sus enrutadores descendientes, además de que el enrutador raíz debe tener la información de ruteo de todos los contenidos en la red.

2.6.2.2.2.2 Tabla hash

En [**38**] se presenta una jerarquía de tablas hash distribuidas (*Distributed Hash Table*, DHT), con las cuales se consigue un enrutamiento de forma equitativa y escalable. Esto quiere decir que si el número de elementos de contenido es C, deben de existir $\log C$ entradas de ruteo. Dentro de las debilidades que presentan las DHT podemos citar que típicamente emplean caminos más largos que un árbol así como que muchas veces el tráfico puede llegar a pasar por lugares donde no es permitido, por lo que se pueden presentar problemas al tratar de enrutar hacia esta información.

2.6.2.3 Elaboración del Caché del contenido

La finalidad de esta característica es mejorar el retraso / rendimiento del desempeño a través de colocar los contenidos más cerca a los usuarios. La base de esta función está dada por las redes de entrega de contenidos (*Content Delivery Networks*, CDN), las cuales han tenido una amplia investigación tal como se muestra en [**39**], [**40**] y en [**41**], además es importante mencionar que este tema se empezó a tratar después de los años noventa [**42**].

2.6.3 Algoritmos de enrutamiento

Pese a existir diferentes propuestas para las redes orientadas a contenido como son los trabajos publicados en [**25**], [**26**], [**27**], [**28**], [**29**], [**6**], [**30**], [**31**], [**32**], [**33**], [**34**], [**35**], [**36**], entre otros; la mayoría se basan en tablas hash distribuidas. Por otra parte, existen dos algoritmos, que son la base para estos algoritmos de enrutamiento, los cuales son Chord [**16**] y Tapestry [**17**], que se describen a continuación.

2.6.3.1 Chord

Chord [**16**], es un protocolo de búsqueda distribuida desarrollado en el 2002 dentro del MIT, el cual propone un mecanismo para localizar de forma eficiente un nodo que almacena un conjunto particular de datos. Chord proporciona soporte para una sola operación: dada una clave, se asigna

en un nodo. La ubicación de los datos se puede implementar fácilmente en la capa superior (aplicación), asociando una llave a cada elemento de datos, y almacenando el conjunto de la clave/dato en el nodo al que se le asigna la clave. Chord adapta el mecanismo en que los nodos entran y salen del sistema, por lo que es posible responder a consultas de información de los nodos, incluso si el sistema está en constante cambio. Los resultados de análisis teóricos y de simulaciones muestran que Chord es escalable, con costes de comunicación y mantenimiento del estado por cada nodo en escala logarítmica con el número de nodos de Chord.

En los siguientes apartados se trata de describir de forma general el funcionamiento del protocolo de Chord, tratando en primera instancia algunas características generales que presenta el protocolo en la sección 2.6.3.1.1 Modelo de sistema, posteriormente se describe el protocolo en la sección 2.6.3.1.2, los conceptos básicos en la sección 2.6.3.1.2.1, y en las siguientes secciones las diferentes características así como algunas mejoras que incrementan la robustez del protocolo.

2.6.3.1.1 Modelo de sistema

Chord simplifica el diseño de sistemas punto a punto (p2p) y aplicaciones basadas en sistemas p2p, abordando los siguientes problemas:

- *Equilibrar la carga.* Chord actúa como una función hash distribuida, extendiendo uniformemente las claves sobre los nodos, lo que proporciona un grado de equilibrio de carga.
- *Descentralización.* Chord es completamente distribuido, ningún nodo es más importante que cualquier otro. Esto mejora la robustez y hace a Chord apropiado para las aplicaciones débilmente organizadas, como son las aplicaciones de punto a punto.
- *Escalabilidad.* El coste de una búsqueda en Chord crece como el logaritmo del número de nodos, por lo que incluso los sistemas muy grandes son factibles. No se requiere ajuste de parámetros para lograr alcanzar la escalabilidad.
- *Disponibilidad.* Chord ajusta automáticamente sus tablas internas para reflejar los nuevos nodos cuando se unen, así como los fallos en los mismos, lo que garantiza que, salvo grandes fallas en la red subyacente, el nodo responsable de una clave siempre se puede encontrar. Esto es cierto incluso si el sistema está en un estado de cambio continuo.
- *Flexibilidad de nombres.* Chord no impone restricciones sobre la estructura de las claves que se busca: el espacio de las claves en Chord es plano. Esto proporciona a las aplicaciones una gran flexibilidad en la forma en que se asignan sus propios nombres a las claves de Chord.

El software de Chord toma la forma de una biblioteca para vincularse con las aplicaciones de cliente y servidor que lo utilizan. La aplicación interactúa con Chord de dos maneras:

- En primer lugar Chord brinda un algoritmo de búsqueda, el cual ofrece la dirección IP del nodo responsable de la llave.
- En segundo lugar, en cada nodo Chord notifica a la aplicación de los cambios en el conjunto de claves que el nodo es responsable. Esto permite que el software de aplicación, por ejemplo, pueda mover los valores correspondientes a sus nuevos destinos cuando un nuevo nodo se une.

La aplicación que utiliza Chord es responsable de proporcionar cualquier tipo de autenticación deseada, almacenamiento en caché, replicación y facilidad de usar nombres de datos.

2.6.3.1.2 Protocolo Chord

Chord usa *consistencia en el hashing* además de una alta probabilidad de equilibrio de carga en la función hash, así como de tener una fracción de claves que deben de ser movidas con una complejidad de:

$$O(1/N) \tag{2.2}$$

Cuando el $N-ésimo$ nodo se agrega o sale de la red hacia una dirección diferente, lo cual es el mínimo necesario para mantener un equilibrio de carga en los nodos.

Un nodo Chord necesita un poco de información sobre los otros nodos para enrutar el contenido, dado que la información es distribuida, resolviendo un nodo la función hash comunicándose con otros pocos nodos. En una red de $N-nodos$, cada nodo mantiene la información de:

$$O(\log N) \text{ nodos} \tag{2.3}$$

y para la búsqueda requiere:

$$O(\log N) \text{ mensajes} \tag{2.4}$$

2.6.3.1.2.1 Consistencia en el hashing

La función de consistencia de hash [**43**] [**44**] asigna a cada nodo tanto una llave así como un identificador de $m-bits$ usando una función base hash (tal como SHA-1) siendo m de una longitud lo suficientemente grande para minimizar que la probabilidad de que la hash de dos nodos o claves sea idéntica. El identificador de cada nodo se encuentra haciendo un hash a la dirección IP del nodo, mientras que el identificador de la clave se produce aplicando la función hash a la clave.

Para lograr consistencia, el hashing asigna las claves de la siguiente manera: los identificadores son ordenados en un círculo de identificación modulo 2^m. La clave k asigna al primer nodo cuyo identificador es igual o sucede al identificador k en el espacio de los identificadores. Este nodo, es llamado el nodo sucesor de la clave k, denotado como $sucessor(k)$, tal como se muestra en la Figura 2.5, en donde se presenta un círculo de identificación con $m = 6$. El círculo de identificación tiene diez nodos y almacena cinco claves. El sucesor del identificador 10 es el nodo 14, por lo tanto la clave 10 será localizada en el nodo 1. De forma similar, las claves 24 y 30 están localizadas en el nodo 32, la clave 38 en el nodo 38 y la clave 54 en el nodo 56, respectivamente.

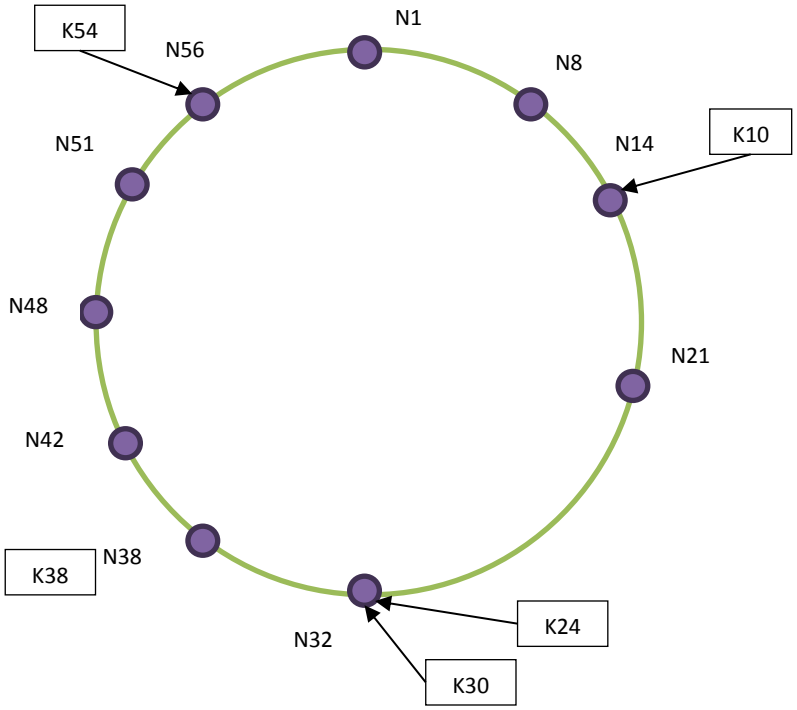


Figura 2.5. Un círculo de identificación que consta de diez nodos y almacena cinco claves.

En el mapa del hashing cuando un nodo n se une a la red, ciertas claves asignadas al sucesor de n ahora se asignan al nodo n. Cuando un nodo n abandona la red, todas las claves son reasignadas al sucesor de n.

Teorema 2.1. Para cualquier conjunto de N nodos y K claves, con una alta probabilidad:

- Cada nodo es responsable por al menos $(1 + \epsilon)K/N$ claves

- Cuando un $(N+1)^{-enésimo}$ nodo se uno o abandona la red, responsable por $O(K/N)$ claves, las cuales cambian de asignación (únicamente para unirse o abandonar el nodo).

En donde el teorema demuestra una cota de

$$\in = O(\log N) \tag{2.5}$$

En lo que respecta a la frase de *alta probabilidad*, se refiere a que se usa k *funciones universales hash*, para proporcionar ciertas garantías. Para Chord en lugar de ocupar esas funciones hash se decidió emplear SHA-1 como su función hash base.

2.6.3.1.2.2 Localización simple de la clave

Las búsquedas de la clave pueden ser implementadas en un anillo lógico de Chord con información de cada nodo respecto a su estado. Cada nodo solo necesita saber como contactar a su actual nodo sucesor en el círculo identificador. Las búsquedas, para un identificador dado, pueden enviarse en el círculo utilizando los punteros de los sucesores hasta encontrar un par de nodos que empaten con el identificador deseado; el segundo elemento en el par es el nodo que realiza la consulta en los mapas. El pseudocódigo que se implementa en la consulta se muestra en la Figura 2.6.

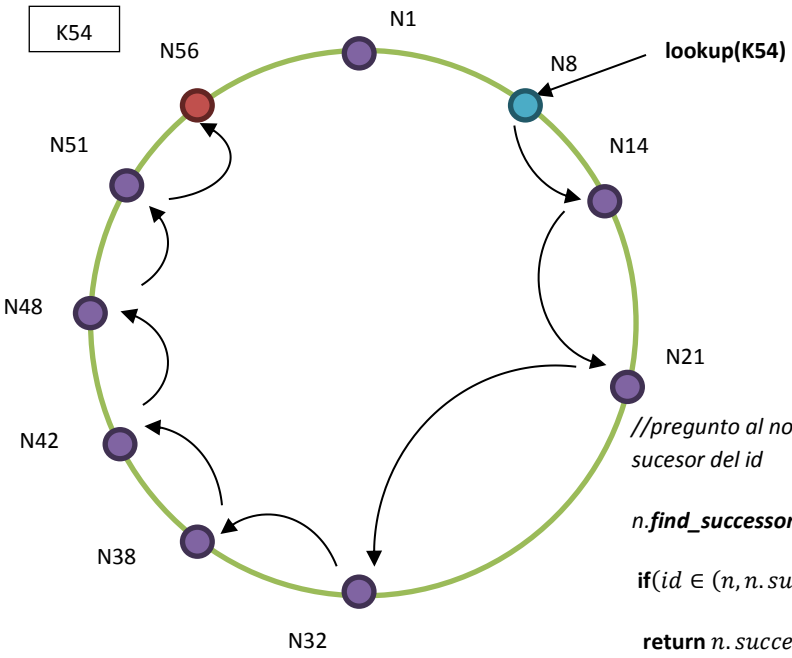
2.6.3.1.2.3 Localización escalable de la clave

El esquema de búsqueda presentado en la sección previa utiliza un número de mensajes lineales de acuerdo al número de nodos. Para acelerar estas búsquedas en Chord se mantiene información de ruteo adicional.

Sea m el número de bits para la clave y los identificadores de los nodos. Cada nodo n, mantiene una tabla de ruteo con (al menos) m entradas, llamada tabla *finger*. La $n-ésima$ entrada en la tabla del nodo n contiene la identidad del primer nodo s, que sucede n por al menos 2^{i-1} en el círculo identificador, esto es:

$$s = successor\left(n + 2^{i-1}\right) \tag{2.6}$$

Donde $1 \leq i \leq m$; además de que todas las operaciones aritméticas son módulo 2^m. Se nombra como nodo s al $i-ésimo$ *finger* del nodo n, y es denotado como $n.finger[i]$, ver Tabla 2.3. Una entrada en la tabla *finger* incluye tanto al identificador de Chord como a la dirección IP (y el número de puerto) de los nodos relevantes. Se debe considerar que el primer *finger* de n es el sucesor inmediato de n en el círculo de identificación y para conveniencia se refiere al primer *finger* como el sucesor.



(a)

//pregunto al nodo n para encontrar al sucesor del id

n.***find_successor***(id)

if$(id \in (n, n.successor])$

return $n.successor$

else

//renvía la consulta al círculo

return successor.find_successor(id)

(b)

Figura 2.6. (a).Muestra un ejemplo en que el nodo ocho realiza la búsqueda de la clave 54. El nodo 8 invoca el método de find_succesor para la clave 54 el cual eventualmente llegara al nodo 56. El resultado presenta también el camino que siguió la consulta para llegar a él. (b) Pseudocódigo para encontrar el sucesor de un nodo con un identificador id.

En la Figura 2.8 se muestra una tabla *finger* del nodo 8, el primer *finger* apunta al nodo 14, como el nodo 14 es el primer nodo que sucede $(8 + 2^0)\ mod\ 2^6 = 9$. De igual manera, el último *finger* del nodo 8 apunta al nodo 42 el cual es el primer nodo que sucede $(8 + 2^5)\ mod\ 2^6 = 40$.

Tabla 2.3. Definición de las variables del nodo n, usando identificadores de m-bits.

Notación	Definición
$finger[k]$	Primer nodo en el círculo que sucede $(n + 2^{k-1})\ mod\ 2^m, 1 \leq k \leq m$
successor	El siguiente nodo en el círculo identificador; $finger[i].node$
predeccessor	El nodo previo en el círculo identificador

La Figura 2.7 presenta el pseudocódigo de la operación *find_successor*, presentando el uso de las tablas *finger*. Si id falla entre n y el sucesor de n, se finaliza *find_successor* y devuelve el nodo

sucesor a n. De otra manera, n busca en sus tablas *finger* para el nodo n' cuyo id es el que precede al identificador id y entonces invoca $find\_succesor$ en n'.

```
//pregunta al nodo n para encontrar al sucesor del id
n.find_successor(id)
 if (key ∈ (n, n.successor])
  return n.successor;
 else
  n'=closest_preceding_node(id)
  return n'.find_successor(id)
```

```
//búsqueda local en la tabla para el predecesor del id
n.closest_preceding_node(id)
 for i = m downto 1
  if (finger[i] ∈ (n, id))
   return finger[i];
 return n;
```

Figura 2.7. Consulta de claves escalable usando la tabla *finger*.

Como ejemplo, se puede considerar el círculo de Chord que aparece en la Figura 2.8(b), en donde el nodo 8 desea encontrar al sucesor de la clave 54. Dado que el *finger* más lejano del nodo 8 que precede a 54 es el nodo 42, el nodo 8 preguntará al nodo 42 para resolver la consulta. Por lo anterior, el nodo 42 determinará el *finger* más lejano en su propia tabla de *finger* para el que precede al nodo 54, esto es el nodo 51. Finalmente encontrará que su propio sucesor, el nodo 56, sucede a la clave 54, la cual regresará el nodo 56 al nodo 8.

Dado que cada nodo tiene entradas en la tabla *finger* en intervalos de potencias de dos alrededor del círculo del identificador, cada nodo puede reenviar la consulta al menos a la mitad de la distancia entre el nodo y el identificador destino, de esta intuición se presenta el siguiente teorema:

Teorema 2.2. Número de nodos que se deben de contactar en una red de nodos Chord.

Con alta probabilidad (o bajo los supuestos de dureza estándar de la función hash) el número de nodos que deben contactarse para encontrar un sucesor en una red de n-nodos es:

$$O(\log N) \tag{2.7}$$

2.6.3.1.2.4 Unión de nodos y estabilización

Para garantizar que la ejecución de las búsquedas es correcta cuando los nodos participantes cambian, Chord debe asegurarse que cada apuntador sucesor del nodo esté actualizado, para lo cual se utiliza el protocolo de estabilización. Chord verifica y actualiza las entradas de la tabla *finger* usando una combinación de *fingers* existentes (y posiblemente desactualizadas) así como apuntadores sucesores correctos.

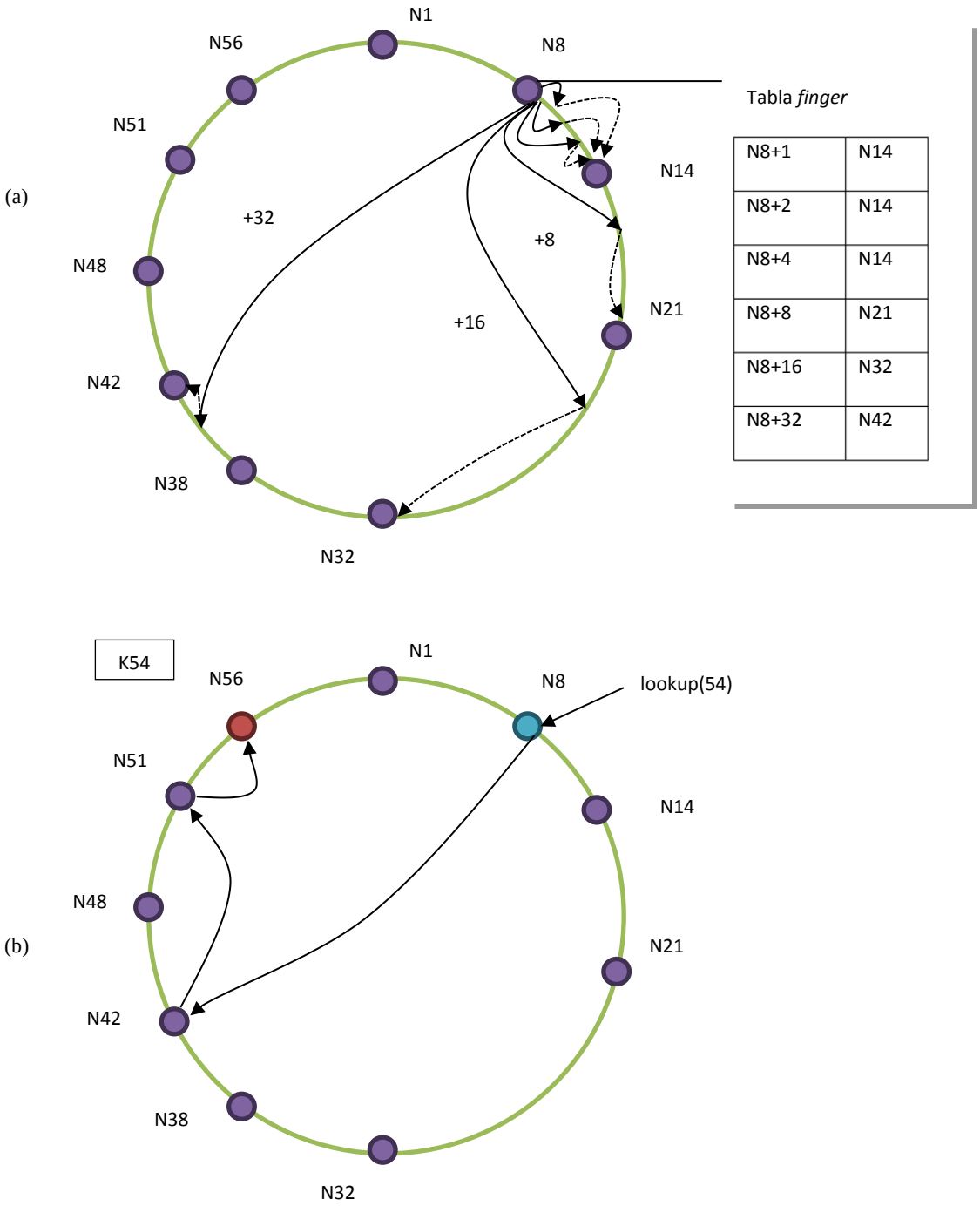


Figura 2.8. (a) Entradas en la tabla finger para el nodo 8. (b) El camino de una consulta para la clave 54 empezando en el nodo 8, usando el algoritmo de la Figura 2.7.

Al unirse algún nodo se puede afectar alguna región del anillo de Chord, y al realizarse una búsqueda antes de que se haya terminado la estabilización puede presentarse alguno de los siguientes comportamientos:

- El caso común es que todas las entradas de la tabla *finger* que estén involucradas en la búsqueda se encuentren razonablemente actualizadas, y por lo tanto la búsqueda encuentre al sucesor correcto en $O(\log N)$ pasos.
- El segundo caso se presenta cuando los punteros al sucesor son correctos, pero las entradas *finger* son incorrectas, lo cual nos llevará a búsquedas correctas, pero pueden realizarse lentamente.
- En el tercer caso, los nodos en la región afectada presentan apuntadores a sucesores incorrectos o las claves no han sido migradas a los nuevos nodos que se incorporaron y posiblemente falle la búsqueda, por lo que la capa superior del software de Chord notaría que los datos deseados no fueron encontrados, presentando la opción de reintentar la búsqueda después de una pausa. Dicha pausa puede ser corta, ya que la estabilización actualiza los punteros de los sucesores de manera rápida.

En la Figura 2.9 se presenta el pseudocódigo para las uniones de nuevos nodos y estabilización. Cuando un nodo n comienza a unirse invoca a $n.join(n')$, donde n' es cualquier nodo conocido en el anillo de Chord. La función $join()$ pregunta a n' encontrar al sucesor inmediato de n. Por si mismo, la función $join()$ no hace que el resto de la red tenga conocimiento de n.

Cada nodo ejecuta la función $stab()$ periódicamente, siendo de esta forma en cómo el sistema aprende de nuevos nodos agregados. Cuando el nodo n ejecuta $stab()$, le pregunta a su sucesor por la existencia del sucesor del predecesor de n, permitiendo que el sucesor pueda cambiar su predecesor de n. El sucesor realiza esto sólo si sabe que no existe algún predecesor más cercano que n.

Como ejemplo simple, se supone que el nodo n se une al sistema y su id queda entre los nodos n_p y n_s. Durante la ejecución de la función $join()$, n declara n_s como su sucesor, adicionalmente n copia todas las claves con un id mayor o igual a su id desde n_s. Cuando el nodo n_s es notificado por n, declara a n como su predecesor. Cuando n_p ejecuta nuevamente la función $stab()$, preguntará a n_s por su predecesor (el cual es ahora n); entonces declarará a n como su sucesor. Finalmente n_p notificará a n y n declarará a n_p como su predecesor. En este punto todos los predecesores y sucesores son correctos. La Figura 2.10 representa el proceso de unión, cuando el id de n es 26 y los id de n_s y n_p son 21 y 23 respectivamente.

```
//construcción de la tabla finger de n'
n.build_fingers(n')
 i_0 := [log(successor − n)] + 1; // primer finger no-trivial
for each i ≥ i_0 index into finger[];
 finger[i] = n'.find_successor(n + 2^(i−1));

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

```
//periódicamente verifica al sucesor inmediato //de n' e informa al sucesor respecto a n
n.stabilize()
 x = successor.predecessor;
 if (x ∈ (n, successor))
  successor = x
successor.notify(n);

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

Figura 2.9. Pseudocódigo de la estabilización en Chord.

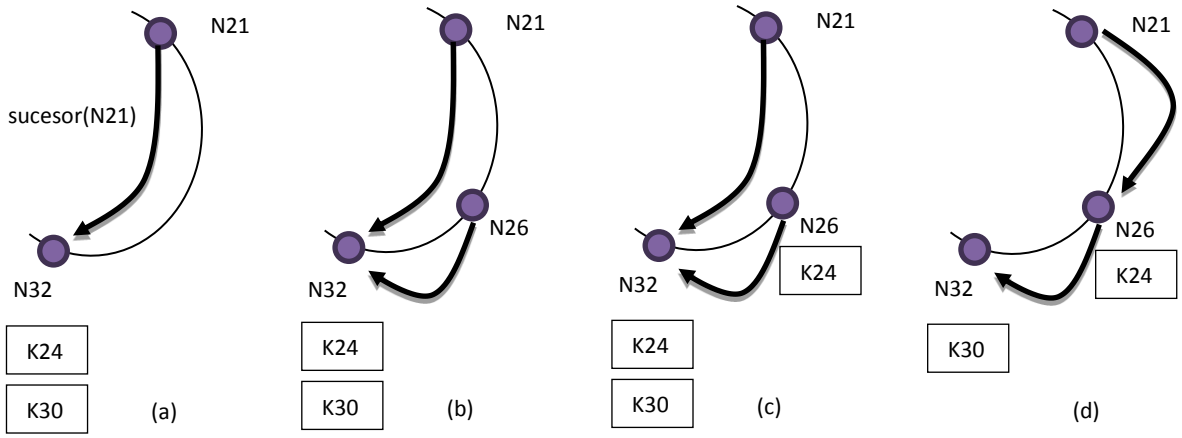


Figura 2.10. Ejemplo que ilustra la operación de unión. El nodo 26 se una al sistema entre los nodos 21 y 32; los arcos representan la relación del sucesor. (a) Estado inicial, el nodo 21 apunta al nodo 32; (b) el nodo 26 encuentra a su nodo sucesor (esto es el nodo 32) y apunta hacia él; (c) nodo 26 copia todas las claves entre 26 y 31 del nodo 32; (d) el procedimiento de estabilización actualiza el sucesor de los nodos 21 al 26.

Tan pronto como los apuntadores a los sucesores son correctos, las invocaciones a la función $find\_predecessor()$ funcionarán correctamente. Los nuevos nodos agregados a la red que no han sido añadidos a las tablas *finger* 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 2.3. Conectividad en una red Chord en una secuencia de uniones.

Si cualquier secuencia de operaciones de unión es ejecutada en intervalos dentro de una estabilización, entonces en algún tiempo después de la última unión los apuntadores al sucesor formaran un círculo con todos los nodos en la red.

El Teorema 2.3 permite identificar que después de un tiempo para cualquier nodo es posible contactar a cualquier otro en la red siguiendo a los apuntadores de los sucesores.

2.6.3.1.2.5 Falla y replicación

La precisión del protocolo de Chord recae en el hecho de que cada nodo conoce a su sucesor. Un sucesor incorrecto llevará a búsquedas incorrectas. Para incrementar la robustez en Chord cada nodo mantiene una lista de sucesores de tamaño r, la cual contiene los primeros r sucesores, entonces si el sucesor inmediato de un nodo no responde, el nodo puede sustituir la entrada desde la lista de sucesores. Una implementación como la descrita anteriormente puede tener un r de tamaño fijo el cual debe de ser de $2\log_2 N$.

Para la función de estabilización (ver Figura 2.9), la lista de sucesores se estabiliza de la siguiente manera: el nodo u reconcilia su lista con sus s sucesores copiando la lista l de s, agregando a s al frente de l y eliminando el último elemento. Si el nodo n nota que su sucesor ha fallado, este lo remplaza con la primer entrada en la lista de sucesores y concilia 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 el sucesor a un nuevo sucesor.

En cuanto a la función de *closest_preceding_node* (Figura 2.7), las búsquedas no son únicamente en las tablas *finger*, sino también en la lista de sucesores para el predecesor inmediato de *id*. Además de lo anterior, el pseudocódigo necesita soportar fallas en los nodos. Si un nodo falla durante la función de *find_successor*, la búsqueda procederá, después de un tiempo dado, en tratar de encontrar el mejor predecesor de acuerdo a los nodos de la tabla *finger* y de la lista de sucesores.

Descrito en los párrafos anteriores se establece que un anillo de Chord es estable si cada nodo de la lista de sucesores es correcto y se declaran los Teorema 2.4 y Teorema 2.5.

Teorema 2.4. De la búsqueda de sucesores implementando una lista de sucesores en una red inicialmente estable.

Si se usa una lista de sucesores de tamaño $r = O(\log N)$ en una red que es inicialmente estable y cada nodo falla con una probabilidad igual a $1/2$, existe una alta probabilidad de que *find_successor* determinará el sucesor más cercano cuando se realice la búsqueda de la clave.

Teorema 2.5. Del tiempo esperado en la búsqueda de una red inicialmente estable implementando una lista de sucesores.

En una red que es inicialmente estable, si cada nodo falla con una probabilidad de $1/2$, entonces el tiempo esperado para realizar la función de *find_successor* es $O(\log N)$.

2.6.3.1.2.6 Salida voluntaria de algún nodo

Dado que Chord es robusto en cuanto a las fallas, un nodo que voluntariamente ha abandonado el sistema puede ser tratado como una falla en el nodo. De cualquier manera, existen dos modificaciones que pueden mejorar el rendimiento de Chord cuando un nodo abandona la red voluntariamente, las cuales son:

- La primer mejora se refiere a que cuando un nodo n está por abandonar la red, el nodo en cuestión puede transferir sus clases a su sucesor antes de que este se aparte de la red.
- La segunda mejora es que n puede notificar a su predecesor p y su sucesor s antes de abandonar la red, lo que ocasionaría que el nodo p removiera a n en su lista de sucesores y agregaría el último nodo de la lista de sucesores de n a su propia lista; de igual manera, el nodo s remplazaría su predecesor con el predecesor de n. Entonces se asume que n envía su predecesor a s y el último nodo de su lista de sucesores a p.

2.6.3.2 Tapestry

Es un prototipo de infraestructura para la localización y ruteo el cual se caracteriza por ser descentralizado, escalable, tolerante a errores y adaptativo, que fue diseñado principalmente para ser una capa para un mejor manejo de p2p. Fue desarrollado en la Universidad de Berkeley, originalmente se publicó como un reporte técnico en el 2001 [**17**], y posteriormente como un artículo de revista [**45**].

Se basa en ruteo por sufijos, similar a los arboles de Plaxton [**46**], que entre sus características destacan:

- Inserción dinámica de nodos.
- Mapeo dinámico del nodo raíz.
- Redundancia en la localización y ruteo.
- Protocolos con tolerancia a fallos.
- Adaptativo y auto configurable para la red dinámica (*soft-state*).
- Soporta objetos móviles.

2.6.3.2.1 Conceptos básicos

Antes de describir los algoritmos que componen al protocolo se describirán de forma breve las características básicas del protocolo.

2.6.3.2.1.1 Nombre de espacio (nodos y objetos)

Los nodos participantes en la capa de Tapestry son asignados como *nodeIDs* uniformemente a través de un largo identificador de espacio, además de que más de un nodo puede ser alojado por un nodo físico. Los puntos finales dentro de la aplicación (objetos), reciben un *GUID* (*Globally Unique Identifier*, por sus siglas en inglés) el cual es seleccionado dentro del mismo espacio de identificadores. Tapestry utiliza una longitud de nombre de 160 bits en base hexadecimal, por lo que idealmente existe un espacio de 2^{80} nombres antes de presentarse alguna colisión. Tapestry asume que los *nodeIDs* y los *GUIDs* son distribuidos uniformemente en el campo de nombre utilizando un algoritmo de seguridad tal como lo es SHA-1. Se dice que el nodo N tiene un *nodeID* N_{id} y un objeto O tiene un *GUID* O_G.

2.6.3.2.1.2 Localización y publicación de objetos

Tapestry mapea dinámicamente cada identificador G a un único nodo vivo denotado como la raíz del identificador G_R, por lo que el nodo raíz es responsable de almacenar la localización de los objetos. Si un nodo N existe con $N_{id} = G$, entonces este nodo es la raíz de G. Debido a esta regla, la búsqueda y la publicación son responsabilidad de los nodos raíz.

El proceso de publicación consiste en enviar un mensaje hacia el nodo raíz. Por cada salto en el camino, el mensaje publicado almacena la información de localización en la forma de mapa, esto es <Object-ID(O), Server-ID(S)>, los cuales son simples apuntadores al servidor S donde O será almacenado, y no es una copia del objeto por si mismo. Para el almacenamiento de réplicas de objetos, estas son almacenadas de acuerdo a la latencia que presentan. En la Figura 2.11 (a) se presenta un pseudocódigo para la publicación de objetos.

//enruta hacia la raíz virtual del ID=ObjectID
$For\ (i = 0, i < \log_2(N), i+= j)\ \{$
j es el número de bits de la base del ID (por ejemplo para dígitos hexadecimales, j=4).
Inserta la entrada en el nodo más cercado que empareje con los últimos i bits.
Si no existe algún emparejamiento, deterministamente escoge una alternativa.
Encuentra el nodo real cuando no quedan rutas.

(a)

//Realiza el mismo camino que (a), excepto que busca por la entrada en cada nodo
$For\ (i = 0, i < \log_2(N), i+= j)\ \{$
Busca en la caché de objetos almacenados
Una vez encontrado enruta vía IP o por objeto de Tapestry

(b)

Figura 2.11. (a) Pseudocódigo para la publicación de objetos. (b) Pseudocódigo para la búsqueda de objetos.

Durante el proceso de búsqueda, los clientes envían mensajes a los objetos. Un mensaje que tiene como destinatario al nodo O, es inicialmente encaminado a la raíz de O. En cada salto, si el mensaje encuentra un nodo que contiene la localización para O, inmediatamente es dirigido al servidor que contiene el objeto, de otra manera el mensaje es reenviado cada salto más cercano a la raíz. Si el mensaje alcanza el nodo raíz, se garantiza que se encuentra una ruta para la localización de O. En la Figura 2.11 (b) se presenta un pseudocódigo para la búsqueda de objetos.

2.6.3.2.1.3 Tabla de vecinos

Para entregar mensajes cada nodo mantiene una tabla de ruteo la cual contiene los *nodeIDs* así como las direcciones IP de los nodos con los que se comunica (a estos nodos se les conoce como vecinos del nodo local). Un nodo *N* tiene una tabla de vecinos con múltiples niveles, donde cada nivel contiene enlaces a nodos que empatan con un prefijo hasta cierta posición en el *ID*, y contiene un número de entradas igual a la base de los *ID*. La $i-ésima$ entrada en el $j-ésimo$ nivel es el ID y la localización del nodo más cercano que empieza con un prefijo $(N, j-1)+i$. Por ejemplo, la novena entrada del cuarto nivel de un nodo, por ejemplo el nodo 325AE es el nodo más cercano con un ID que empieza con 3259. En la Figura 2.12 se muestra una malla de un nodo, indicando el nivel con el que es emparejado hacia los nodos que esta enlazado, en donde un nodo con un mayor nivel significa un mayor emparejamiento.

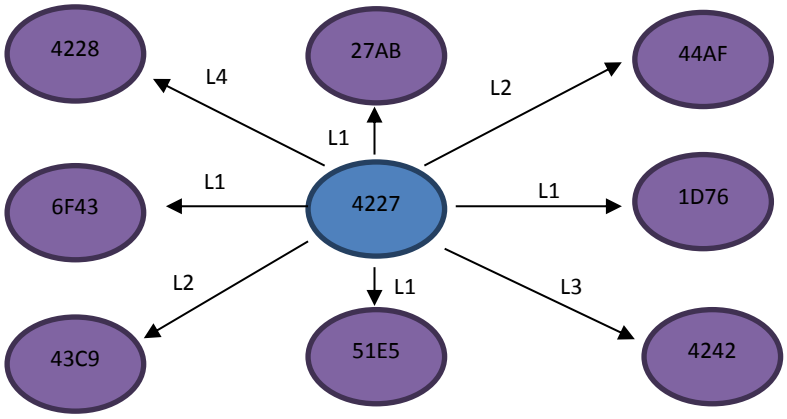


Figura 2.12. Malla de ruteo para el nodo 4227, describiendo los niveles para construir la tabla de vecinos.

En la Tabla 2.4, se presenta un esbozo de como está constituida la tabla de vecinos de la Figura 2.12.

Tabla 2.4. Tabla de vecinos para el nodo 4227, de acuerdo a la malla de la Figura 2.12.

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

2.6.3.2.1.4 Nodo Tapestry

Un nodo en Tapestry consta de cuatro elementos:

- Tabla de vecinos. Tal como se describió en la sección anterior, son entradas de vecinos cercanos cuyo *ID* empata en cierto criterio con su *ID* actual.
- Lista de backpointers. *ID* de nodos que se refiere a este como su vecino.
- Punteros de localización de objetos. Tuplas en la forma de ObjectID, NodeID, para la localización más eficiente de ObjectID.
- Monitores Hotspot. Tuplas en la forma de ObjectID, NodeID, Frecuencia. Las cuales permiten le creación de copias de objetos.

En la Figura 2.13 se presenta un ejemplo de un nodo Tapestry.

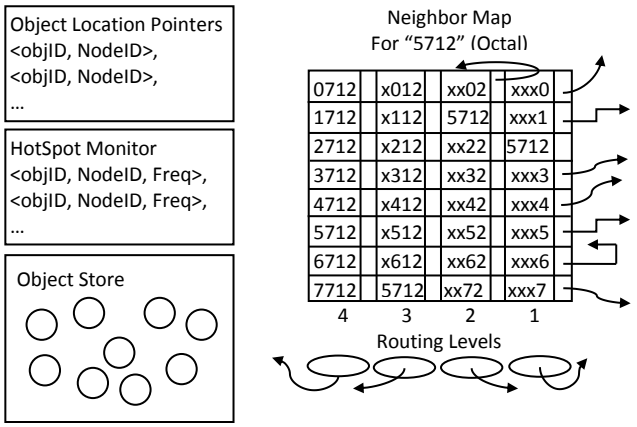


Figura 2.13. Representación completa de un nodo Tapestry.

2.6.3.2.1.5 Ruteo por sufijos

Cuando se enruta hacia G_R, los mensajes son reenviados por vecinos cercanos cuyo *nodeIDs* son progresivamente más cercanos (esto es por tener un emparejamiento de prefijos más cercanos) hacia G en el espacio de los *ID*. En el $n-ésimo$ salto, se llega al nodo más cercano a través de la función $hop(n)$, el cual comparte el sufijo de una longitud de n dígitos.

Ejemplo: 5324 enruta a 0629 vía:

5324-> 2394 -> 1429->7629->0629

2.6.3.2.2 Localización y enrutamiento

El mecanismo de localización de Tapestry es similar al esquema de localización de Plaxton, donde cada nodo que enruta al nodo raíz almacena la localización de las réplicas más cercanas de forma más flexible en comparación de Plaxton, permitiendo que la aplicación defina la operación de selección, como por ejemplo escoger a la caché más reciente.

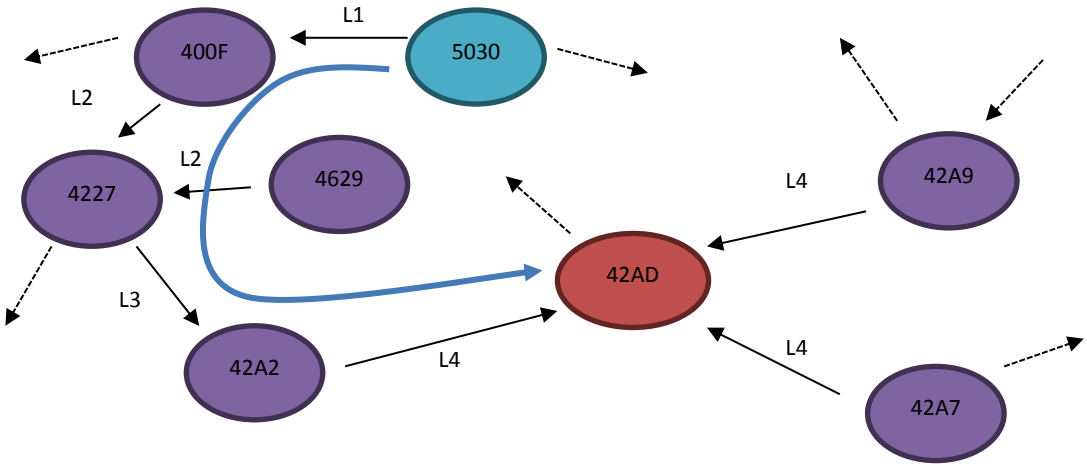


Figura 2.14. Camino de un mensaje en Tapestry. El camino que toma un mensaje originado desde el nodo 5230 destinada al nodo 42AD en una malla Tapestry.

En la Figura 2.14 se presenta una ruta que tomó un mensaje a través de la infraestructura Tapestry. El camino para el $n - ésimo$ salto comparte un prefijo de longitud $\geq n$ para el destino ID, siendo que Tapestry revisa en su $(n+1) - énesimo$ nivel en la tabla de vecinos para el registro emparejando con el próximo dígito destino en el ID. Este método garantiza que cualquier nodo en el sistema puede ser alcanzado a lo más en $\log_\beta N$ saltos lógicos, en un sistema con un espacio de nombres N, IDs de base β y asumiendo que son consistentes las tablas de vecinos.

2.6.3.2.2.1 Enrutamiento surrogate

Un nodo responsable es un nodo con el mismo ID al del objeto, pero dicho nodo no suele existir. Cuando un dígito no puede ser emparejado, Tapestry busca por el dígito más cercano (determinístamente) en la tabla de ruteo, si ese dígito tampoco existe busca el siguiente y así sucesivamente hasta encontrar uno, lo cual se conoce como enrutamiento *surrogate*, donde cada nodo existente ID es mapeado a algún nodo vivo con un ID similar. Un Ejemplo de este procedimiento se presenta en la Figura 2.15.

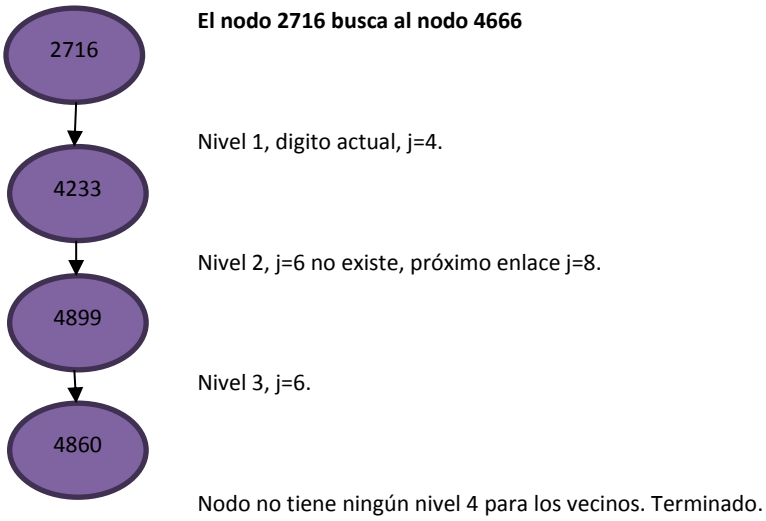


Figura 2.15. Ejemplo del enrutamiento surrogate, cuando el nodo 2716 trata de localizar al nodo 4666.

2.6.3.2.2.2 Tolerancia a las fallas en el ruteo

Para detectar fallas en los enlaces y en los servidores durante el tiempo de operaciones, Tapestry puede basarse en tiempos fueras (*timeouts*) de TCP. Adicionalmente, cada nodo Tapestry emplea un *backpointers,* el cual envía periódicamente pulsaciones en paquetes UDP a los nodos con los que son vecinos, detectando el *ID* de cada mensaje de los nodos a los que llegan los *backpointers*, con lo que se puede detectar rápidamente un fallo, así como también tablas de vecinos corruptas.

Para operaciones bajo fallas, cada entrada en la tabla de vecinos mantiene de respaldo dos nodos vecinos (los más cercanos además del vecino principal). Cuando el vecino primario falla, se decide utilizar los vecinos alternativos.

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 fallas en los nodos y enlaces son reparados relativamente en un tiempo corto, se mantiene un periodo de segunda oportunidad. Al momento de recibir un mensaje de satisfactorio, indica que la falla ha sido reparada y el puntero de la ruta original es marcado como válido nuevamente.

2.6.3.2.2.3 Tolerancia a las fallas en la localización

Para prevenir problemas en la localización de algún objeto se implementan múltiples nodos raíces, esto es que se concatena una pequeña y constante secuencia de valores *salt* a cada *objectID*, entonces al resultado se le aplica la función hash y se identifican las raíces apropiadas. Estas raíces son usadas vía enrutamiento *surrogate* durante el proceso de la publicación para insertar la

información de localización en Tapestry. Cuando se localiza un objeto, Tapestry realiza el mismo proceso de hash con el destino de *objectID*, generando un conjunto de raíces para la búsqueda.

2.6.3.2.3 Nodos dinámicos

Tapestry incluye un número de mecanismos para mantener las tablas de ruteo consistentemente y asegurar la disponibilidad de objetos. A continuación se describen los tres tipos de comportamiento que se pueden presentar.

2.6.3.2.3.1 Inserción de nodos

Existen cuatro componentes que se presentan en la inserción de un nodo N en la red Tapestry:

a) Necesidad de conocer. Los nodos son notificados de N, ya que N se registró en una entrada vacía de sus tablas de ruteo.
b) N se puede convertir en la nueva raíz objeto para los objetos existentes. Las referencias a esos objetos deben de ser movidas a N para mantener la disponibilidad de los objetos.
c) Los algoritmos deben de construir una tabla de ruteo óptima para N.
d) Los nodos cerca a N son notificados y pueden considerar usar N en sus tablas de ruteo como una optimización.

La inserción del nodo empieza cuando N es un *surrogate* de S (la raíz del nodo que N_{id} mapea en la red existente. S encuentra p, por la longitud del prefijo más largo que su *ID* comparte con N_{id}. S envía un mensaje multicast de reconocimiento, que llega a todos los nodos existentes que comparten el mismo prefijo por medio de atravesar el árbol en base a sus *nodeIDs*. Tan pronto los nodos reciben los mensajes, ellos agregan N a sus tablas de ruteo y transfieren las referencias de punteros raíces locales como sea necesario, con esto se cumplen los puntos (a) y (b).

Los nodos alcanzados por medio del mensaje multicast de contacto de N se convierten en el primer conjunto de vecinos a ser usados en su tabla de ruteo durante la construcción de esta. Después, N realiza una búsqueda iterativa de los vecinos más cercanos empezando a llenar tablas de ruteo de nivel p, recortando la lista a los nodos k más cercanos, y solicita a estos k nodos que envíen sus apuntadores al nodo anterior (*backpointers*) de ese nivel. El conjunto de nodos resultante contiene todos los nodos que apuntan a cualquiera de los k nodos del previo nivel de ruteo y se convierte en el próximo junto de vecinos. Entonces N decrementa a p y repite el proceso hasta que todos los niveles son llenados, esto completa el punto (c). Los nodos contactados durante la iteración de los algoritmos utilizan a N para optimizar sus tablas de ruteo donde sean aplicables, completando el punto (d).

2.6.3.2.3.2 Eliminación voluntaria de nodos

Si el nodo N abandona Tapestry de forma voluntaria, este avisa al conjunto de nodos D que son *backpointers* de N su intención, junto con el nodo que es el remplazo para cada nivel de ruteo de su propia tabla de ruteo. Cada uno de los nodos notificados republican el tráfico, tanto para N como su remplazo. Mientras que N enruta las referencias de los nodos raíces objetos hacia sus nuevas raíces y notifica a los nodos N cuando ha terminado.

2.6.3.2.3.3 Eliminación involuntaria de nodos

En una red dinámica las fallas en la red existen debido a fallas en los enlaces o particiones de la red, además de la movilidad de los nodos ya que pueden entrar y abandonar la red en periodos cortos de tiempo. Tapestry permite que la disponibilidad de los objetos y enrutamiento en tales ambientes construyendo tablas de enrutamiento redundantes y referencias de localización de los objetos.

Para mantener la disponibilidad y redundancia, los nodos usan mensajes periódicos para detectar enlaces desaparecidos y fallas en los nodos.

2.6.3.3 Resumen

En las dos secciones anteriores (2.6.3.1 y 2.6.3.2) se describieron 2 de los 4 primeros algoritmos que ocuparon DTH, ya que estos han sido implementados y desarrollados con diferentes mejoras y variaciones. En la Tabla 2.5 se presenta un resumen de la complejidad que presentan los algoritmos Tapestry, Chord, CAN y Pastry para la creación de rutas, la identificación de vecinos, y los mensajes que se deben de crear para insertar un nuevo nodo.

Tabla 2.5. Cuadro comparativo de parámetros para la creación de rutas, el estado de vecinos, y la complejidad de mensajes que se deben de hacer para una inserción de nodo, en los algoritmos de Tapestry, Chord, CAN y Pastry.

Propiedades	Tapestry	Chord	CAN	Pastry
Parámetro	Base b	Ninguno	Dimensión d	Base b
Longitud de la ruta lógica (enrutamiento)	$log_b\ N$	$log_2\ N$	$O\left(d * N^{1/d}\right)$	$log_b\ N$
Estado vecinos	$b\ log_b\ N$	$log_2\ N$	$O\ (d)$	$b\ log_b\ N + O(b)$
Routing Overhead (RDP)	0 (1)	0 (1)	0 (1) ¿?	0 (1) ¿?
Mensajes para insertar	$O\left(log_b{}^2\ N\right)$	$O\left(log_2{}^2\ N\right)$	$O\left(d * N^{1/d}\right)$	$O\ (log_b\ N)$

2.7 Resumen

En el capítulo actual, se presentó el por qué la importancia del desarrollo de la tesis, así como la definición y amplio margen de trabajo sobre las diferentes áreas en que se pueden trabajar, además de presentar las bases con las que se puede trabajar el enrutamiento orientado a contenido.

Capítulo 3 Diseño

En el presente capítulo se describe el modelo a seguir para poder realizar el objetivo planeado. Primero se detalla del algoritmo de enrutamiento base así como sus propiedades, para después mostrar como esta constituido el algoritmo.

3.1 Algoritmo de Enrutamiento Basado en Prefijos (EbP)

En esta sección se describirá un algoritmo recientemente propuesto para MANETs (redes móviles ad hoc) el cual utiliza etiquetado por prefijos y tablas hash distribuidas, el cual se propone como la implementación de este en redes tipo Ethernet y como base para el enrutamiento orientado a contenido.

La propuesta del algoritmo esta basado en dos propuestas, muy similares llamada AIR y PROSE, los cuales se describen a continuación.

AIR (Automatic Integrated Routing, por sus siglas en inglés) [**47**], es un algoritmo que se publicó en el 2009 el cual se basa en otro artículo propuesto en ese mismo año por los mismos autores [**48**], el cual implementa un algoritmo llamado PROSE (Prefix Routing Over Set Elements, por sus siglas en inglés).

PROSE es un protocolo de enrutamiento escalable en MANETs basado en la combinación del uso de etiquetas de prefijo y de hash distribuidas. En PROSE, los nodos utilizan mensaje vecino a vecino para poder etiquetarse a sí mismos con etiquetas de prefijo que permite un enrutamiento implícito de cualquier nodo a cualquier destino de red. Los nodos implementan una tabla hash distribuida para almacenar las asignaciones entre los identificadores de nodo (por ejemplo, un MAC o dirección IP) y sus etiquetas de prefijo. Con lo anterior los destinos publican su existencia y las fuentes deben de suscribirse a sus destinos.

Mientras que AIR es un enfoque unificado para enrutamiento unicast y multicast en redes móviles ad hoc. En AIR, los nodos ejecutan un algoritmo de enrutamiento distribuido para asignar etiquetas de prefijo a sí mismos. Las etiquetas se asignan de tal manera que el enrutamiento unicast o multicast es en automático, ya que una ruta desde un nodo a un destino está definido por las etiquetas de prefijo del nodo, así como que ningún nodo debe de conocer un camino completo a cualquier destino.

3.1.1 Descripción general

Cada nodo en la red de EbP se supone que tiene una etiqueta de posición independiente, que llama identificador único global (GID, por sus siglas en inglés) y EbP asigna a cada nodo una etiqueta posicional.

AIR enruta dos fases separadas. En la primera fase, se construye el grafo acíclico dirigido etiquetado (LDAG, por sus siglas en inglés) que forma la estructura básica de enrutamiento y las etiquetas de cada nodo utilizando un sistema de etiquetado basado prefijo. En la segunda fase se establece anclas para cada nodo y estas anclas se hacen responsables de la asignación entre GID y las etiquetas de posición.

Un algoritmo de elección distribuido determina el origen o raíz de la LDAG, y se lleva a cabo durante la primera fase. El etiquetado de los nodos se lleva a cabo al mismo tiempo que el mensaje del nodo raíz se difunde en la red de EbP. Cada nodo hijo en el LDAG está ordenado en función de su GID y se le asigna una etiqueta posicional basada en este orden. Las etiquetas posicionales contienen, como su prefijo, la etiqueta de posición de su padre y su sufijo único ordenado. Las etiquetas posicionales se propagan usando mensajes de saludo *(hello)* y los nodos con múltiples etiquetas posicionales de padres comunes se les asigna una etiqueta primaria posicional en base a su primer asignación. Los nodos realizan la caché de estas múltiples etiquetas posicionales, ya que son indicadores de varias rutas entre los nodos.

La segunda fase de EbP comienza cuando cada nodo calcula la posición de su etiqueta de anclaje aplicando una función hash a su propio GID tal como SHA-1. La elección de la función de hash es clave para tener una distribución equilibrada de los anclajes a través de la red para evitar particiones debido a fallos correlacionados. Cada nodo determina la etiqueta de posición de su ancla y agrega un registro a su mensaje de saludo que el cual describe la asignación de su GID a su etiqueta de posición. Los mensajes que se reenvían con esta información permiten que los nodos involucrados agreguen la información de mapeo en sus respectivas tablas y avanzar hacia el siguiente nodo que tiene un prefijo más cercano para el ancla. La agregación mensaje de *hello* de estas notificaciones reduce el número de mensajes de sobrecarga realizados al crear el mapeo de nodos, pero no aumentar el tamaño del mensaje. Este no es mucho más largo que el tamaño de mensaje que está limitado por la longitud máxima de la ruta, la cual sólo crece a un ritmo de

$$\log N/2 \qquad (3.1)$$

El intervalo de los mensajes de saludo también se mantiene mediante la introducción de un mayor tiempo de espera, durante el cual no se envían mensajes de saludo, a menos que haya un cambio en la etiqueta de posición del nodo.

Cada nodo actualiza los mensajes de *hello* a través de sus vecinos hacia el ancla. En el caso de que la etiqueta posicional hash sea inexistente, ausente o este caída debido a un fallo, el último nodo que está más cerca del prefijo se marca como el ancla. Esto es correcto, porque para llegar al nodo con la etiqueta de posición completa, cualquier paquete tendría que pasar por el prefijo más próximo. Una vez que el anclaje se determina para un GID dado, el ancla almacena la asignación y sirve como el directorio para ese nodo. Un ancla puede asignar varios nodos.

Para enrutar sobre la LDAG de nodos etiquetados se considera que todo nodo origen debe de aplicar la función hash al GID del destino conocido para obtener la etiqueta de posición del anclaje. Entonces el nodo de origen inicia una solicitud de etiqueta de posición hacia la ancla del nodo destino. El ancla responde con la asignación más reciente de su tabla y se inicia una ruta entre el origen y el destino y se debe de considerar que este camino no tiene que pasar por el ancla. El algoritmo de enrutamiento sigue una estrategia de enrutamiento *greedy*, reenviando el paquete al nodo que mejor se ajuste al prefijo. En el peor de los casos esto implicaría la elección del nodo padre como el próximo salto, hasta que se alcance el prefijo máximo y luego enrutar hacia el hijo más cercano hasta que el destino se alcance.

3.1.2 Modelo de red

Dado un grafo dirigido $G\ (V, E)$, donde E es el conjunto de aristas que representan las conexiones entre dos nodos y V es el conjunto de vértices que representan el conjunto de nodos. G se construye de tal manera que todos los enlaces E son acíclicos y el resultante es un gráfico dirigido acíclico (DAG, por sus siglas en inglés) la cual contiene su raíz en algún nodo R. Además, los enlaces se supone que son bidireccionales y los costes de los enlaces no son negativos. Los nodos se suscriben o eliminan del DAG arbitrariamente, cambiando la topología dinámica.

Sea Σ el alfabeto que contiene un número finito de símbolos y Σ^* el conjunto de todas las cadenas de Σ de tal manera que $|\Sigma| \geq 2$. Entonces una etiqueta L y una etiqueta de prefijo de Σ^* se definen como:

- Una etiqueta L de Σ^* es una cadena única del alfabeto Σ^* y cada enlace, en cualquier caso, tiene un único alfabeto de Σ.
- Una etiqueta de prefijos L de alguno nodo y es una etiqueta de Σ^* tal que $L = p(x)$ donde $p(x)$ es la etiqueta de prefijos del $parent(y) = x$.

3.1.3 Estructura de las etiquetas por prefijos

Cada nodo etiqueta a sus enlaces con cada uno de sus vecinos con una letra w desde Σ. Si w_i representa la letra asignada al enlace $i - ésimo$ de cualquier nodo, entonces:

$$w_i \rightarrow \{w_i \in \Sigma \mid w_i \neq w_{i+1} \forall i \leq d-1\} \quad (3.2)$$

Donde d es el grado del nodo. Por lo tanto, una única letra es asignada a cada enlace de conexión de cualquier nodo a sus vecinos.

La lógica del etiquetado etiqueta a cada nodo en el LDAG de forma como se realiza un algoritmo *breadth first*. Dado que el LDAG puede ser organizado como un árbol de k-elementos, con k siendo el grado del LDAG, cada hijo (hasta *k*), es asignado con una etiqueta de prefijos Λ de la siguiente manera:

Una etiqueta por prefijos Λ del nodo y es una palabra de Σ^* tal que

$$\Lambda = \Lambda_{parent} \odot l' \quad (3.3)$$

Donde Λ_{parent} es la etiqueta de prefijos obtenida por el padre y $\odot$ es el operador de concatenación, siendo Λ_{parent} concatenado con un sufijo único sobre k de Σ a la forma de Λ.

Las etiquetas de prefijos de los nodos definen la relación de predecesor en el LDAG, representada por $\hookleftarrow$, tal que para dos nodos s y d en el LDAG:

- $s \hookleftarrow d$: s precede d y entonces d puede ser alcanzada recorriendo los subárboles de s.
- $d \hookleftarrow s$: d precede s y entonces d puede ser alcanzada desde s recorriendo los ancestros de s.
- $d \approx s$: d y s son nodos tal que $|\Lambda_d| = |\Lambda_s|$ además de que ambos comparten el mismo ancestro. En este caso d puede ser alcanzado recorriendo hacia el ancestro en común hasta que $r \hookleftarrow s$ se encuentra, donde r es el ancestro en común y entonces ir hacia el subárbol más cercano mientras $r \hookleftarrow d$ se mantenga y d sea alcanzado. Entonces $d \approx s \Rightarrow r \hookleftarrow s \Rightarrow r \hookleftarrow d$.
- $dr \approx s$: EL cual es un caso especial del punto anterior, ya que $|\Lambda_d| \neq |\Lambda_s|$ y el único ancestro en común es el nodo raíz.

Los nodos construyen y mantienen la LDAG enraizada a un nodo raíz elegido mediante mensajes de *hello* intercambiados entre los vecinos que están a un salto. Cada mensaje de saludo especifica el identificador del nodo de la raíz, un número de secuencia monótona creciente asignado por el nodo raíz, el prefijo y el identificador del nodo de la etiqueta del nodo emisor, y una lista de tuplas con la asignación de etiquetas a los prefijos de los identificadores de nodos. El algoritmo de elección raíz debe de ser elegible como el nodo con la mayor cobertura a un salto. A medida que la red empieza a organizarse con las etiquetas de prefijo, el proceso de etiquetado puede producir

LDAGs múltiples. Cuando un nodo está en la interface de dos LDAGs, la etiqueta de cada uno de ellos y se compara la etiqueta lexicográficamente más grande es elegido como el dominante.

Considérese el ejemplo en la Figura 3.1. Las etiquetas de prefijo para los nodos se asignan en el alfabeto $\Sigma = 0;\ 1;\ 2$. La raíz del árbol se le asigna una letra de Σ y los nodos conectados a la raíz están etiquetados 00 y 01 sucesivamente. En el siguiente nivel, cada nodo se le asigna una etiqueta de prefijo único a través del enlace combinado con el prefijo de su padre. Los nodos E y M están etiquetados 001 y 010, respectivamente.

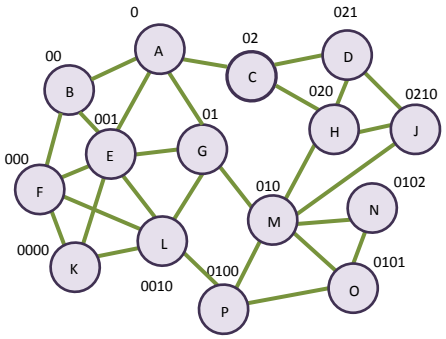


Figura 3.1. Etiquetado por prefijos en una red.

3.1.4 Enrutamiento

Para enrutar a un destino d, s nodo elige el enlace a cualquiera de sus vecinos en dos saltos que ofrece la longitud máxima de la etiqueta prefijo que coincide con la etiqueta de prefijo del destino. Esto es simplemente una lógica congruente de máximo prefijo, que selecciona el siguiente salto utilizando una estrategia *greedy* que considera la vecindad de dos saltos de un nodo, y se puede encontrar caminos más cortos que el tradicional árbol de prefijo de enrutamiento mediante el aprovechamiento de la diversidad de un LDAG en comparación a un árbol de prefijo. Por ejemplo, si existe una etiqueta en la zona de dos saltos que es lexicográficamente más cerca del destino, el siguiente salto se elige de manera que se reenvía el paquete a ese nodo en lugar de enrutamiento a través de prefijos en el árbol. Una forma de asegurar de que esta estrategia *greedy* no se encuentra con un mínimo local es seleccionando al azar el siguiente salto cuando todos los nodos ofrecen el mismo prefijo coincidente. Si L_s, L_d son las etiquetas de alguno nodo de origen y nodo destino, entonces en cada nodo x en el camino desde el origen al destino se ejecuta el Algoritmo 3.1.

Se requiere: Raíz etiquetada, $n > 1$
  if $L_s \neq L_d$ **then**
    L_x= nodo con prefijo Max coincidente de la tabla de enrutamiento
    reenvía el paquete al nodo x
    $L_s = L_x$
  else
    paquete enrutado
  end if

Algoritmo 3.1. Enrutamiento por prefijos.

De la relación predecesor inducida por las etiquetas de prefijo en el LDAG, un determinado nodo selecciona el siguiente salto a una etiqueta prefijo de acuerdo con los cuatro casos posibles permitidos por la relación predecesor. El siguiente lema (Lema 3.1) muestra que el algoritmo converge en rutas correctas.

Lema 3.1. El enrutamiento basado en prefijos es correcto en una red conectada sin cambios en la topología.

Demostración. Por la definición de las etiquetas de prefijo, cada nodo tiene una etiqueta prefijo único, y el LDAG está libre de bucles de construcción. Sin pérdida de generalidad, se considera una fuente s y un destino d. Ya que pertenecen a la LDAG, deben cumplir con uno de los cuatro casos de la relación antecesor, y cada nodo elegido como próximo salto a partir de s hacia d también debe satisfacer la misma relación. Debido a que el LDAG es finito y no cambia, debe haber por lo menos una ruta sin bucles de s a d en la LDAG.

Sea L la representación de una tupla (S_n^a, Λ_n^a), donde S_n^a presenta la secuencia de números que es originada en a siendo reenviada por el nodo n, y Λ_n^a representa la etiqueta por prefijos del nodo actual con respecto a a. S es un entero monotónicamente creciente, mientras que Λ es una palabra de Σ^*. Se define al operador $\prec$ como un conjunto de identificadores de pares ordenados L. Si L_x^a, L_y^a son dos tuplas tal que:

$$\{L_x^a \prec L_y^a \mid L_{x,y}^a \in \Sigma\} \tag{3.4}$$

$$if \left\{(S_x^a < S_y^a) \vee \left[(S_x^a = S_y^a) \bigwedge (\Lambda_x^k \prec \Lambda_y^j)\right]\right\} \tag{3.5}$$

Para satisfacer la afirmación de que esta tupla establece un orden entre los nodos, se muestra que el operador $\prec$ es anti-reflexivo y transitivo en el universo de Σ^*, Lema 3.2.

Lema 3.2. La relación $\prec$ es un orden parcial de tuplas sobre el universo de Σ^*.

Demostración. La primer parte del lema es intuitivo, porque dos tuplas son iguales sólo si el número de secuencia y la etiqueta de prefijo son los mismos. Entonces $L_x^a \prec L_y^a$ o $L_y^a \prec L_x^a$, sólo uno de los cuales puede contener y es por lo tanto anti-reflexiva. Para demostrar que la transitividad se mantiene, se consideran tres etiquetas y para conveniencia se llaman L_1, L_2, L_3 tal que $L_1 \prec L_2$ y $L_2 \prec L_3$. De la ecuación se observa que $S_1 < S_2 < S_3$ como S son monotónicamente incrementales. Esto implica que $S_1 < S_3$. Esto implica que si $S_1 \not\prec S_3$ entonces $L_1 \prec L_3$ sólo si $S_1 = S_3$ y $\wedge_1 < \wedge_3$ (la comparación es en naturaleza lexicográfica). Dado que se sabe que $\wedge$ es válido por transitividad, la relación $\prec$ es un ordenamiento sobre el espacio de Σ^* y establece una relación sucesor-predecesor.

Una vez demostrado que las etiquetas de prefijo preservar un orden, se muestra en un teorema más general en el que un algoritmo de enrutamiento converge a caminos sin bucles, incluso en la si la topología es dinámica.

Teorema 3.1. Un esquema de enrutamiento por prefijos con números de secuencia 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 los nodos que se encuentra a una distancia de un salto. Del Lema 3.2 se determinó que cada nodo etiquetado se ordena con respecto a la longitud de la etiqueta por prefijo lexicográfico y un número de secuencia. Por lo tanto, si el enrutamiento crea un camino entre dos nodos, la estrategia *greedy* está garantizado mejorar alguna métrica (número de saltos, la distancia, carga, etc.) hacia un destino en cada salto. Descrito lo anterior, un esquema de enrutamiento por prefijos con números de secuencia enruta correctamente, incluso si cambia la topología.

3.1.5 Topología dinámica

La inserción de un nodo en cualquier punto en un subárbol es simple ya que afecta únicamente a los nodos de forma local. Mientras la propiedad de unicidad de la etiqueta se mantiene y la etiqueta está asignada correctamente, los nuevos nodos se pueden añadir con un coste constante. Los nodos que salen de la red, sin embargo, tienen un procedimiento más complejo que afecta a la función de servicio de directorio, el cual se trata en la sección 3.1.6.

Si el nodo y se une a la red en algún nodo x, el nodo x tiene una etiqueta válida y el número de secuencia más reciente. Mientras que el nodo y, puede no tener una etiqueta o una etiqueta con

un número de secuencia inferior que el número de secuencia actual. Se pueden considerar cada uno de estos casos de inserción:

- *Sin etiqueta*: y no tiene etiqueta previa y x asigna una etiqueta a este nodo, determinando el contador local y asignando una etiqueta que no ha sido asignada para el nuevo enlace con el nodo y. L_y contendrá la etiqueta de x, L_x y añadiendo la letra adicional a la misma con la ayuda del contador.
- *Etiquetado y con menor identificador de nodo padre*: Si y tiene una etiqueta pero su identificador de nodo padre es menor que el nodo x, entonces este asigna una etiqueta y actualiza el valor del nodo padre.
- *Etiquetado y con mayor o igual identificador de nodo padre*: En este caso, el nodo espera para una actualización en su identificador de nodo padre antes de determinar si el nuevo nodo necesita etiquetado o si por sí mismo tiene que restablecer su etiqueta, ya que ahora tiene una trayectoria más reciente a la raíz. Si L_y sigue siendo más elevado comienza un restablecimiento de etiqueta y si no, se procede a etiquetar el nodo como en el caso anterior. Si sigue siendo igual y no comparte el mismo padre, entonces se considera como una trayectoria horizontal hacia algún otro subárbol con prefijo L_y.

El pseudocódigo es mostrado en el Algoritmo 3.2.

Se requiere: Raíz etiquetada R; Número total de nodos, $n > 1$
Se requiere: contador local, c_x
while nueva actualización de S llega **do**
if L_y es un nuevo nodo OR $L_y \rightarrow S_y^A < L_x \rightarrow S_x^A$ **then**
cambiar la etiqueta de y a $L_x + k$ donde $k \cong c_x$
incrementar c_x
else
esperar por la siguiente actualización de S
end if
end while
if $L_y \rightarrow S_y^A == L_x \rightarrow S_x^B$ **then**
el nuevo nodo tiene la última etiqueta
else
nodo y tiene una etiqueta mas reciente
inicia el restablecimiento de etiqueta en x
end if

Algoritmo 3.2. Unión de nodo.

Descrito lo anterior, se establece el Lema 3.3.

Lema 3.3. Los nodos que se unen a la red adquieren sus etiquetas correctamente.

3.1.6 Directorio de Servicio Distribuido

El servicio de directorio distribuido (DDS) es un esquema hash basado en la dirección de consulta que une algún identificador global a las etiquetas que están asignados localmente. El servicio de directorio asigna anclajes a cada uno de los nodos de la red, la etiqueta de la cual se determina es establecida aplicando la función hash al identificador global. El servicio de directorio en cada ancla se mantiene gracias a las actualizaciones periódicas de los nodos en los que estos anclajes están designados.

El servicio de directorio distribuido asegura que un sólo nodo no estará encargado con la tarea de mantener un directorio y divide los requisitos de almacenamiento a través de la red. La elección de la función de hash determina la eficiencia en la distribución de los nodos de anclaje. La función hash toma un identificador global de entrada y produce otra salida. El nodo más cercano a la salida se designa como el ancla para ese nodo con un identificador global dado.

El ancla recibe actualizaciones periódicas desde el nodo incluidos los mensajes de actualización de la etiqueta del nodo. La falla de empate con cualquier nodo exactamente, resulta en definir a algún nodo predecesor que tiene un prefijo de superposición como el ancla. Esto garantiza que si cualquier ancla se reposiciona en la LDAG, los nodos anteriores manejar las actualizaciones dinámicas hasta que un nuevo nodo es identificado.

3.2 Modelo

El algoritmo propuesto, ocupa la misma ideología de etiquetado que AIR, es decir se basa en prefijos. Para poder desarrollar el algoritmo, se considera que:

- Todos los nodos cuentan con un identificador único y a partir de este, se elige al que tenga el mayor identificador.
- Una vez elegido al nodo líder, a partir de los vecinos que lo rodean (los que se encuentran a una unidad de distancia) se empieza a establecer un orden de padre-hijo en donde el padre es quien esta a una unidad menor que el hijo, y el hijo esta a un salto del padre, siempre y cuando el padre pueda aceptar al hijo (es decir que tenga capacidad para almacenarlo como su hijo).
- Se elige la filosofía de etiquetado por prefijos de EbP, tal como se describió en 3.1.
- Una vez etiquetado el nodo y después de recibir varias notificaciones del mismo padre y líder, se considera que esa parte de la red se ha estabilizado y procede a publicarse (informar al nodo ancla) de la existencia a partir de una función hash pública.

- Para suscribirse a algún nodo, se aplica la función hash al identificador del nodo, para poder solicitar la etiqueta de éste y poder entablar un camino hacia él.
- El enrutamiento se va realizando salto a salto, el nodo que se visita es quien determina hacia donde es el siguiente salto en base a la etiqueta destino.

Con los puntos anteriores, se cumple con las características de un modelo de enrutamiento orientado a objetos, tan como se describió en el tema 2.6 Redes orientadas a contenido.

Capítulo 4 Análisis Experimental

En el presente capítulo se describe en primera instancia, las propiedades que respaldan al algoritmo planteado, para posteriormente describir las métricas que se utilizaran en el algoritmo propuesto, continuando con la descripción de los escenarios para finalizar con el análisis de los resultados presentados.

4.1 Propiedades

En la siguiente sección se presentan dos teoremas que respaldan el comportamiento del algoritmo planteado.

Teorema 4.1. Las tablas de enrutamiento son del orden $O(d)$ donde d es el grado máximo de la red.

Demostración. El enrutamiento se realiza salto a salto, comparando la etiqueta destino con la de los uno-vecinos. Por lo tanto cada nodo sólo debe conocer las etiquetas de los nodos con los que comparte un enlace.

Teorema 4.2. La relación entre los caminos encontrados por el algoritmo propuesto y los caminos más cortos existentes en la red (stretch) es $O\ (\log n)$.

Demostración. La relación padre-hijo del algoritmo de etiquetado induce un árbol de profundidad $O\ (\log n)$. Por lo tanto, el camino más largo en esta estructura es $O\ (\log n)$. Como el camino más corto posible es de un solo salto, entonces el *stretch* está acotado por $O\ (\log n)$.

4.2 Pruebas

En esta sección se presentan las diferentes pruebas que se realizaron al protocolo propuesto. Para esto se definieron varias métricas y escenarios de comportamiento, los cuales se describen a continuación.

4.2.1 Métricas

Para el análisis del algoritmo se presentan 3 métricas que se utilizaron para la caracterización del algoritmo.

- Porcentaje de paquetes entregados (*PDR, Packet Delivery Ratio*), el cual mide la efectividad del algoritmo para establecer y mantener rutas que conecten a la fuente con sus receptores. Siendo que el porcentaje de paquetes entregados esta dado por la siguiente fórmula:

$$\sum_{i=1}^{n_s} \frac{n_r(i)}{n_m(i)} \tag{4.1}$$

  donde n_s es el número total de paquetes enviados, $n_r(i)$ es el número de paquetes que se reciben del paquete i, y $n_m(i)$ es el número de paquetes enviados por el nodo *i*.

- Retardo extremo a extremo, el cual es el tiempo promedio que les toma a los paquetes de datos desplazarse desde el nodo fuente hasta llegar a su destino. Esta métrica esta relacionada tanto con la longitud de los caminos que conectan a las fuentes con los receptores, así como los niveles de contención del medio y de la congestión de las colas de los paquetes. Esta métrica esta dada por:

$$\frac{1}{n_r}\sum_{i=1}^{n_s} t_r(i) - t_s(i) \tag{4.2}$$

  donde n_r es el número total de paquetes recibidos, $t_s(i)$ es el tiempo en el que el paquete i fue enviado, y $t_r(i)$ es el tiempo en el que el paquete i fue recibido por el nodo.

- La sobrecarga de control es el número total de paquetes de control que tramite un algoritmo de enrutamiento para establecer y mantener las rutas.

4.2.2 Escenarios

Para las pruebas se hace uso del protocolo MAC IEEE 202.11, para acceder al medio, los nodos dependiendo de su función, envían paquetes de control ya sea broadcast o únicas, así también estos nodos son estáticos, es decir en ningún momento se desplazan de su lugar.

En el desarrollo de las pruebas los nodos fueron establecidos aleatoriamente, permaneciendo en el lugar asignado por los 80 segundos que dura la ejecución de los experimentos. Mientras que para los nodos fuentes de tráfico, fueron designados aleatoriamente, los cuales tenían un tiempo de iniciación de consumo de recursos entre los segundos 51 al 59, dejando el tiempo previo para el conocimiento de la red por parte del algoritmo; y designando el tiempo posterior para el termino de envío de paquetes.

Para generar los datos a transmitir, se hace uso de la aplicación *CBR (Constant Bit Rate*), el cual permite establecer la frecuencia de transmisión, el destino así como el tamaño de los paquetes. Cabe mencionar que CBR no presenta alguna otra aplicación más que generar paquetes de datos.

Para poder realizar y presentar los resultados, se hace uso del simulador NS2 en su versión 2.35, el cual provee simulaciones realistas de la capa física y es un estándar de facto para la evaluación de protocolos de comunicaciones.

De forma general se hicieron simulaciones con un tamaño fijo de los paquetes enviados (1024 bytes), así como de variar el número de paquetes que se envían por segundo (32, 16, 12), en ambientes de flujos concurrentes (5, 10, 30 y 60) para dos modelos de estado de nodos (uno de 60 nodos y uno de 30 nodos) en un área de 1000 x 1000 metros, tal como se presenta en la Tabla 4.1.

En cuanto a la comparación del algoritmo propuesto, dado que no se encontró implementación de algún método para el enrutamiento orientado a contenido en NS2, se optó por compararlo con el protocolo AODV [**49**], pero realizando una modificación que consiste en primer instancia a que se realice una llamada a un nodo propuesto (el cual simula que es el servidor de nombres), una vez que se haga la solicitud al nodo y este haya respondido, se procede como segunda instancia a enviar los paquetes al nodo destino.

Tabla 4.1. Descripción de los escenarios de simulación con un tamaño de paquete de 1024bytes.

Número de nodos	30 y 60 nodos	**Posición de los nodos**	Estática
Área de simulación	1000m x 1000m	**Tamaño de paquetes**	1024bytes
Tiempo de simulación	30s	**Flujos concurrentes**	5, 10, 30 y 60
Número de paquetes enviados por segundo	32, 16 y 12	**Tráfico**	CBR

4.2.3 Simulaciones

Tal como se describió en temas anteriores, a continuación se presentan los resultados obtenidos, variando las posiciones de los generadores de tráfico. Cabe hacer hincapié que los resultados presentados son el promedio de diez ejecuciones del algoritmo.

4.2.4 Paquetes entregados

De acuerdo a lo que se presenta en la Figura 4.1 y la Figura 4.2 se puede observar que a mayor número de elementos en la red, existe la posibilidad de que sea menor la relación de número de paquetes enviados sobre los recibidos. También se puede notar que en los flujos concurrentes más altos se llega a estabilizar el número de paquetes entregados, esto debido a la saturación de paquetes de control, por lo que los nodos cercanos pueden establecer su conexión y consumir los datos.

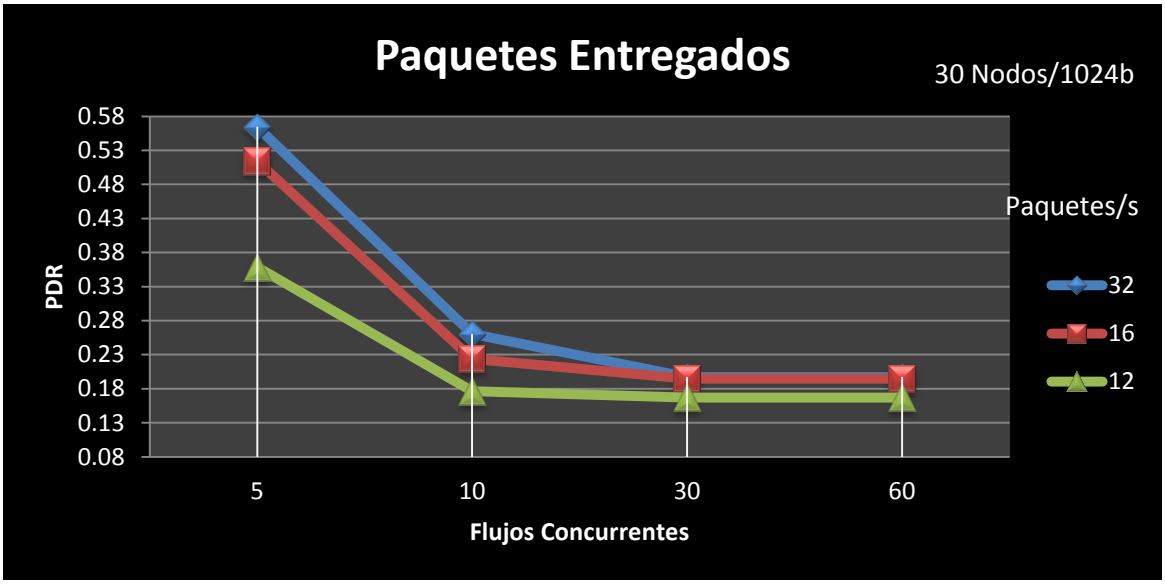


Figura 4.1. Porcentaje de paquetes entregados en una red de 30 nodos con una capacidad de 1024bytes cada paquete, en 5, 10, 30 y 60 flujos concurrentes con una velocidad de envío de 32, 16 y 12 sobre segundo en paquetes CBR.

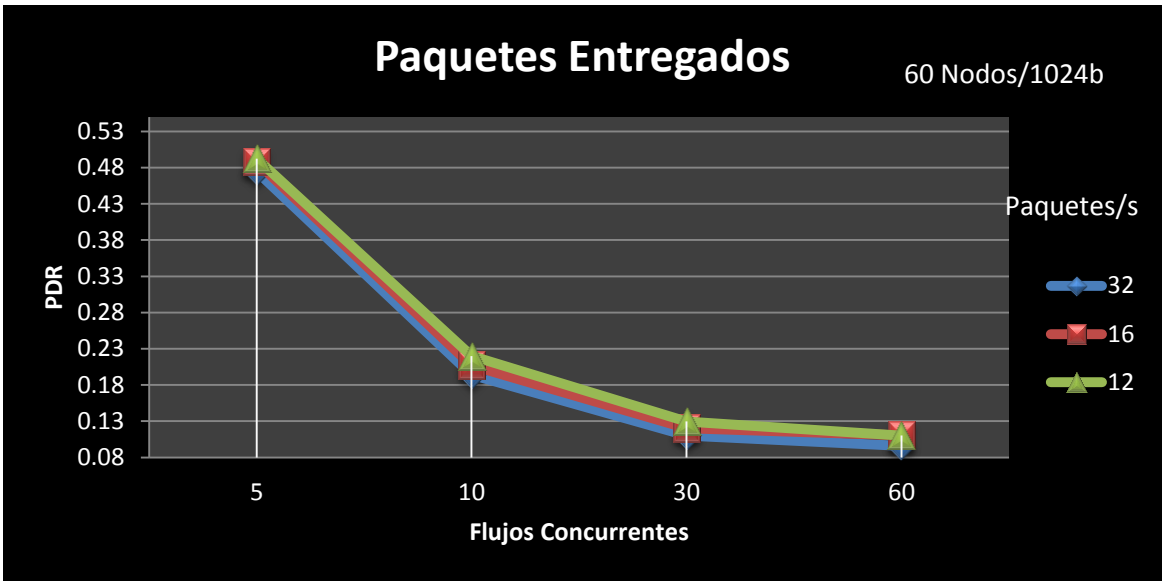


Figura 4.2. Porcentaje de paquetes entregados en una red de 60 nodos con una capacidad de 1024bytes cada paquete, en 5, 10, 30 y 60 flujos concurrentes con una velocidad de envío de 32, 16 y 12 sobre segundo en paquetes CBR.

4.2.5 Retardo extremo a extremo

En cuanto al análisis por el retardo de la entrega de paquetes, se puede observar que el menor tiempo tanto para la Figura 4.3 y la Figura 4.4 se presenta cuando existe un menor número de flujos concurrentes, el cual empieza aumentar conforme aumenta el número de flujos hasta alcanzar un grado máximo (como por ejemplo el pico en 10 flujos concurrentes de la Figura 4.3), y empieza a decrecer hasta alcanzar un comportamiento estable (tal como se muestra en la Figura 4.3 entre los flujos concurrentes de 30 y 60), esto debido a que se satura el uso de la red y únicamente se transmite a los nodos cercanos y hacia los nodos que se conocen en la red, por lo que se esperaría que si se realizaran más simulaciones, en cuanto a un mayor número de flujos concurrentes, la Figura 4.4 llegue a comportarse de la misma forma que la Figura 4.3.

Ahora bien los picos presentados en todas las figuras de este análisis (es decir la Figura 4.3 y la Figura 4.4) son el punto máximo o el punto más cercano (en cuanto a las referencias) que se puede tener en el algoritmo dadas las condiciones para poder tratar de entablar rutas antes de decaer por el tráfico de paquetes de control.

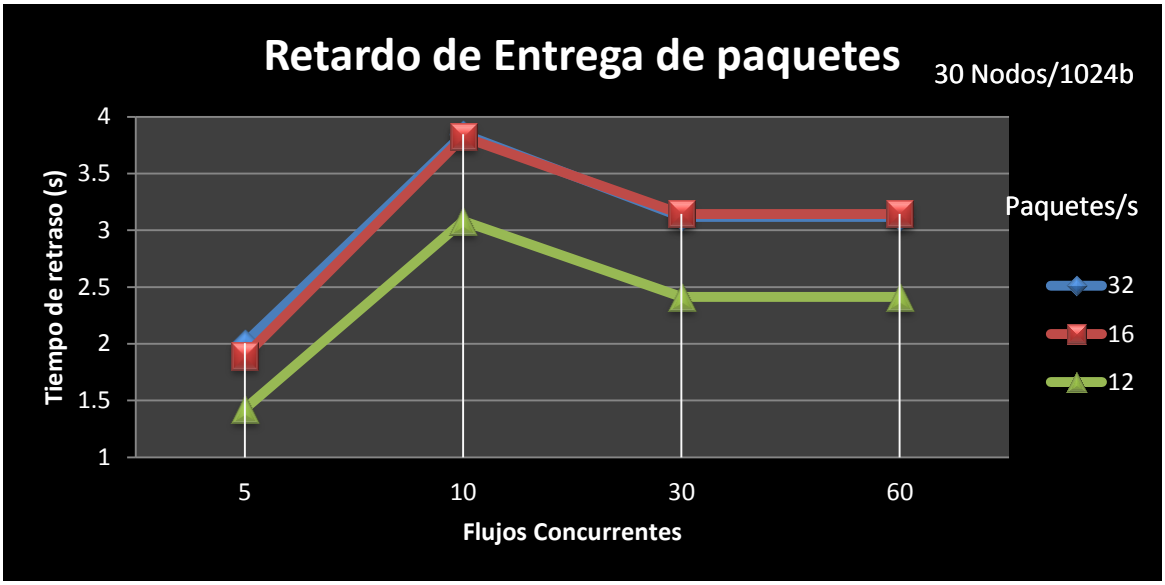


Figura 4.3. Tiempo de retraso en segundos para el retardo de entrega de paquetes en una red con 30 nodos y paquetes CBR de una capacidad de 1024bytes a una velocidad de envío de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

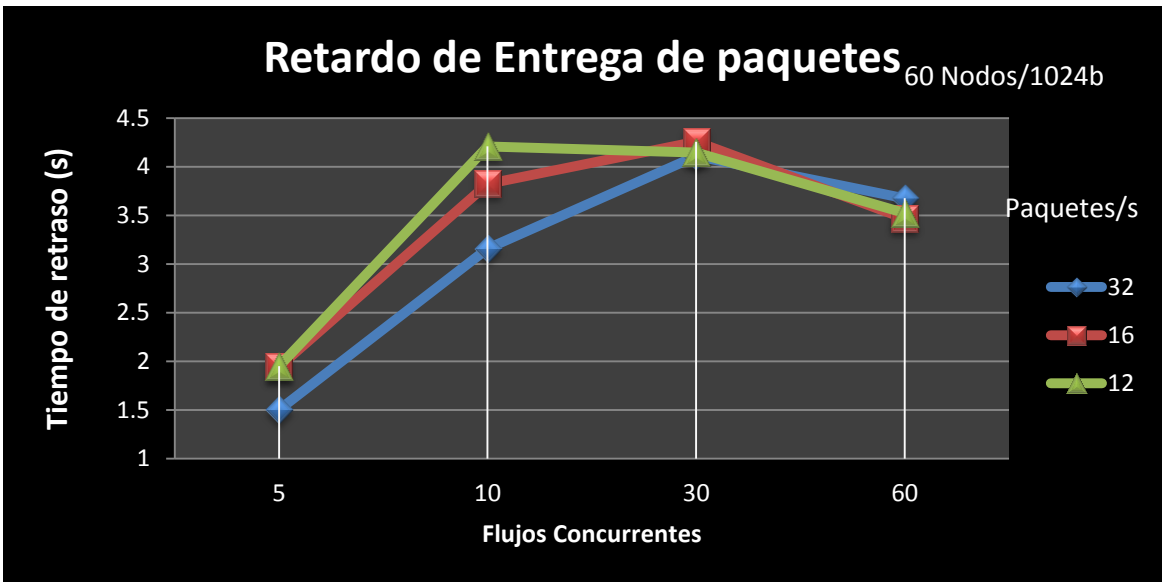


Figura 4.4. Tiempo de retraso en segundos para el retardo de entrega de paquetes en una red con 60 nodos y paquetes CBR de una capacidad de 1024bytes a una velocidad de envío de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.6 Sobrecarga de paquetes de control

En cuanto a la sobrecarga de control se puede ver en la Figura 4.5, que se llegó al límite de envío de paquetes de control, ya que este no incrementa o es muy poco, y eso esta acompañado por los resultados de la Figura 4.1, y la Figura 4.3; mientras que la Figura 4.6, presentan que aún no llegan a saturar la red, pero dado el comportamiento de la entrega de paquetes en la Figura 4.2, se puede predecir que esta en el punto o cerca de saturar la red. Mientras que se puede observar en la Figura 4.5 y la Figura 4.6 que dependiendo del número de flujos concurrentes incrementa también el número de paquetes de control.

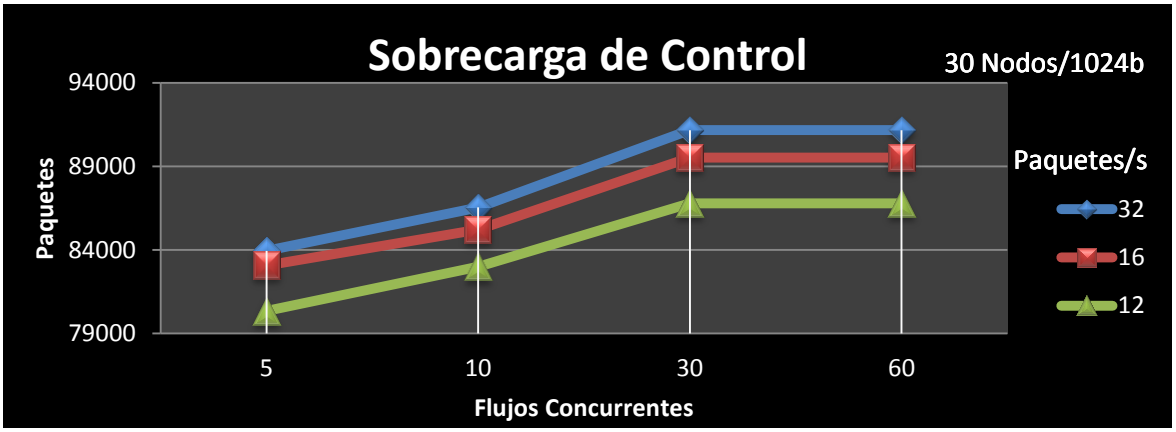


Figura 4.5. Número de paquetes de control enviados en una red de 30 nodos con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

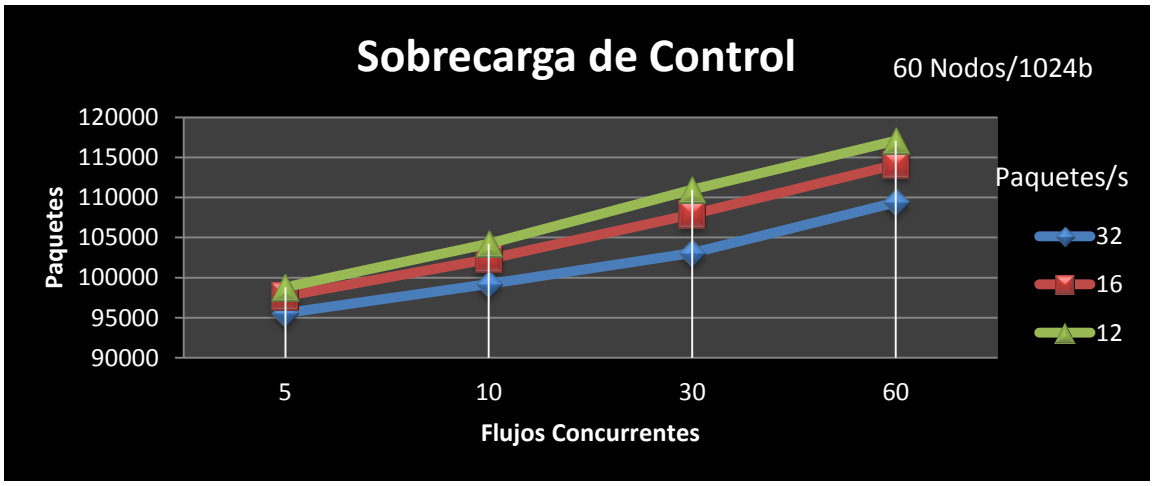


Figura 4.6. Número de paquetes de control enviados en una red de 60 nodos con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.7 Características de la propuesta

A continuación se mencionan y explican cinco características del modelo propuesto, las cuales son el promedio de la respuesta de ruta entre la solicitud de rutas (contemplando y no la pérdida de paquetes dado la saturación de la cola de estos), el número de saltos en las rutas generadas, el número de vecinos de un nodo y el máximo de solicitudes de ser hijo hacia un nodo.

4.2.7.1 Promedio de respuesta de ruta entre la solicitud de rutas

De acuerdo con la Figura 4.7 y la Figura 4.8, se puede observar que entre más flujos concurrentes existen en la red, se encuentra una menor respuesta de solicitud de ruta, ubicándose el cociente entre el 0.92 y el 0.41 en la experimentación, comportándose de acuerdo a la saturación de la red, ya que de acuerdo a la Figura 4.5, la Figura 4.7 presenta también un comportamiento estable entre los flujos concurrentes de 30 y 60. Lo cual presenta un comportamiento interesante ya que si la cola de paquetes se encuentra saturada, empieza a desechar los más antiguos.

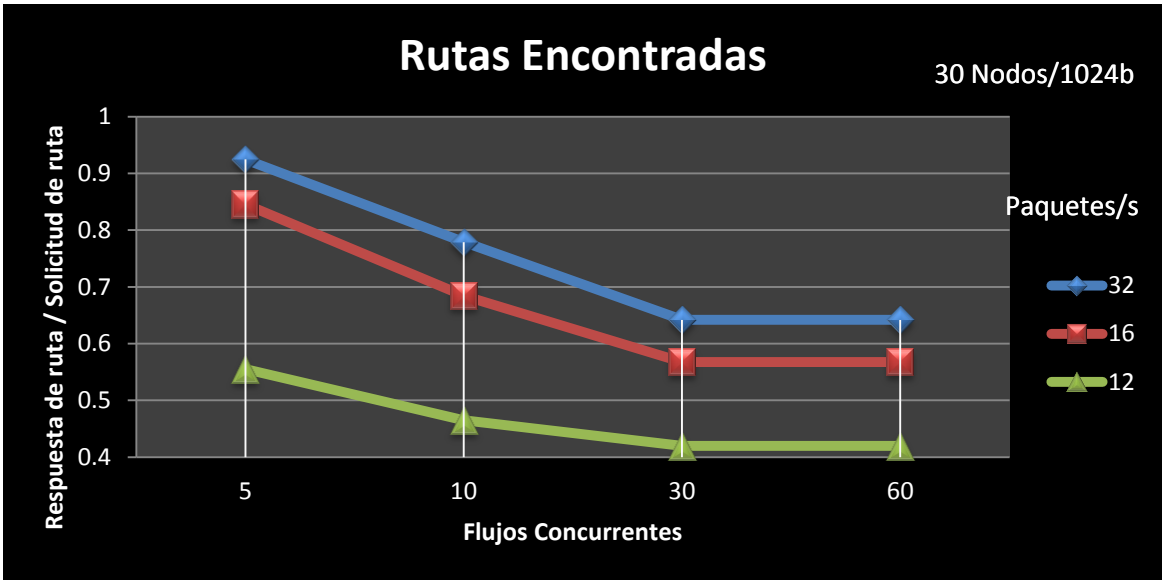


Figura 4.7. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 30 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

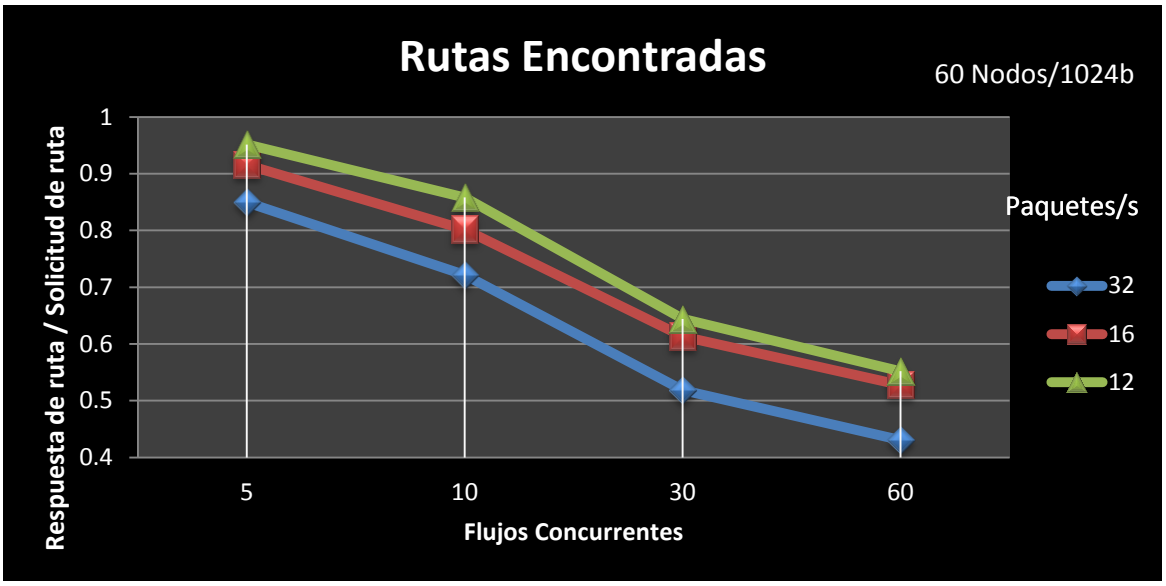


Figura 4.8. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 60 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.7.2 Promedio de respuesta de ruta entre la solicitud de rutas (contemplando la perdida de paquetes)

Si se consideran los paquetes que se desechan (por saturación en la cola de paquetes) y se elimina su solicitud, se presenta un comportamiento casi perfecto al encontrar en casi todos los casos siempre la ruta deseada, tal como se muestra en las Figura 4.9 y Figura 4.10.

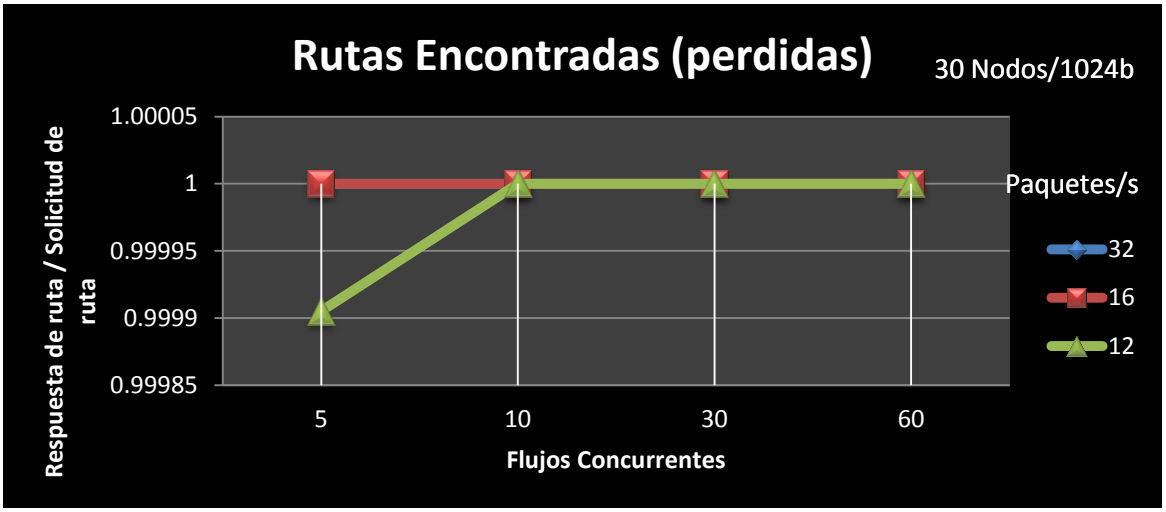


Figura 4.9. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 30 nodos, omitiendo los paquetes perdidos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

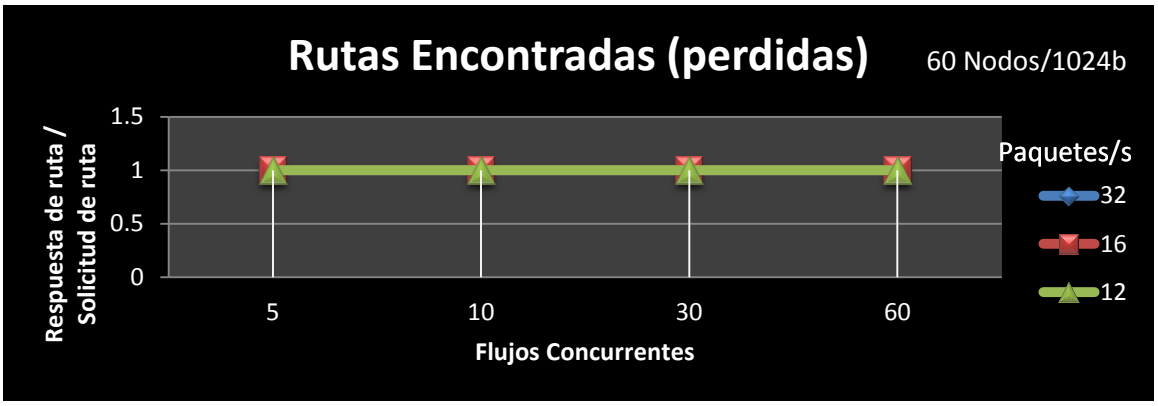


Figura 4.10. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 60 nodos, omitiendo los paquetes perdidos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60

4.2.7.3 Saltos en las rutas generadas

En la Figura 4.11 y la Figura 4.12 se observa el comportamiento promedio de saltos que presenta el algoritmo al recorrer las rutas generadas para enviar los paquetes CBR como flujos de tráfico, siendo de manera general dos unidades de salto el promedio dado, además de obtener en las simulaciones un máximo de 14 unidades como el mayor número de saltos para recorrer una ruta generada, la cual se presentó en la red de 60 nodos. El comportamiento estable de la Figura 4.11 se debe a que se encuentra saturada la red y sólo se envían a los nodos cercanos y/o conocidos, tal como se ha explicado en secciones anteriores.

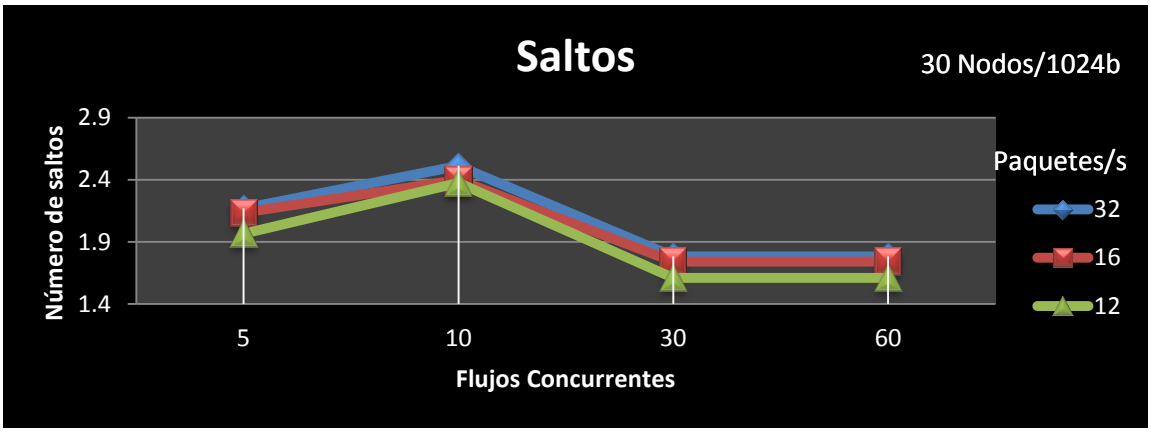


Figura 4.11. Número de saltos promedio que se dan al cursar las rutas generadas para enviar paquetes CBR, en una red de 30 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

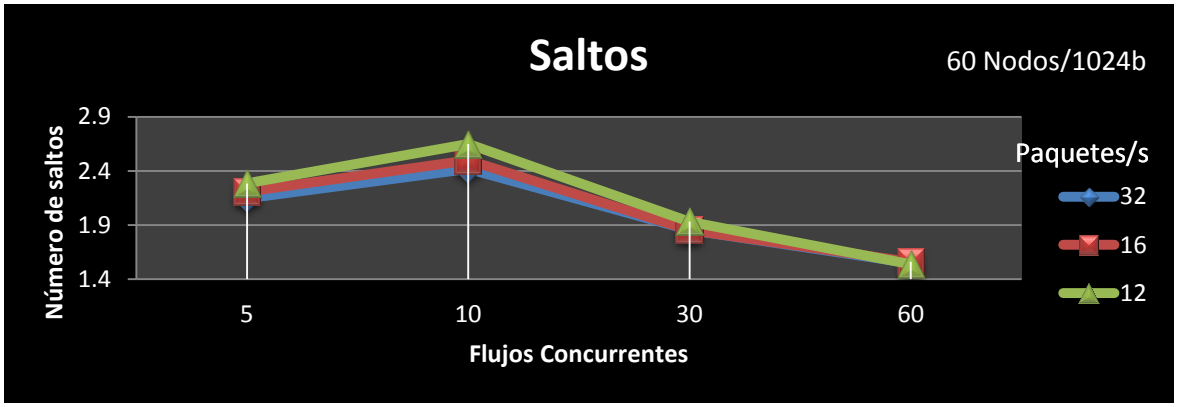


Figura 4.12. Número de saltos promedio que se dan al cursar las rutas generadas para enviar paquetes CBR, en una red de 60 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.7.4 Número de vecinos de un nodo

Una de las características que se revisó fue la saturación de nodos vecinos, ya que a mayor saturación puede que se tarde un nodo en encontrar a su nodo padre, dado que pueden estar saturados (la implementación limita a que un nodo tenga un máximo de 16 hijos). De la Figura 4.13 y la Figura 4.14, se puede observar que la red con 30 nodos presenta una mayor saturación de nodos en comparación con la de 60 nodos, respectivamente. Los picos presentes en la Figura 4.13 y la Figura 4.14, se debe a la estructura de la red, en este caso se debe a un posicionamiento aleatoria de los nodos involucrados en dicha red.

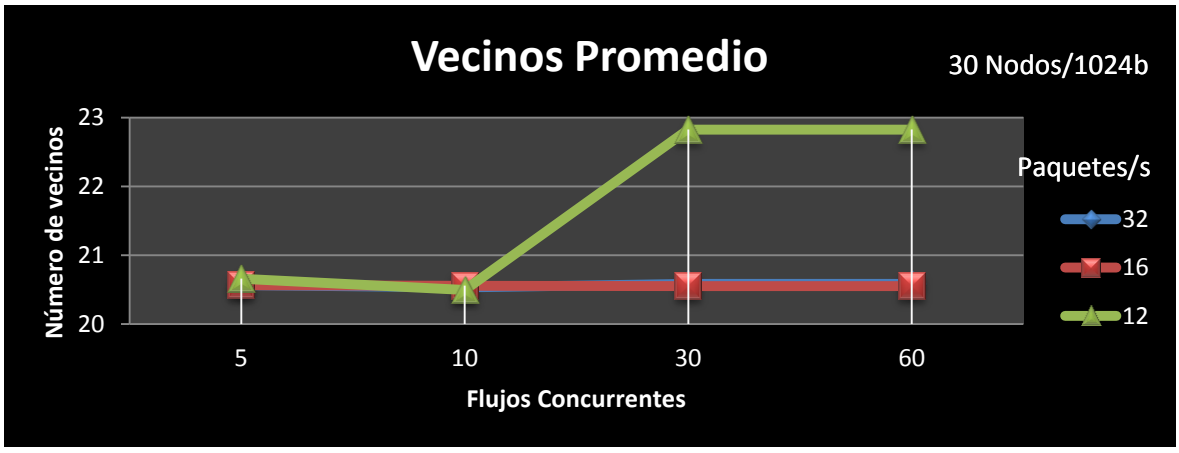


Figura 4.13. Número de vecinos promedio que tiene un nodo en una red de 30 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

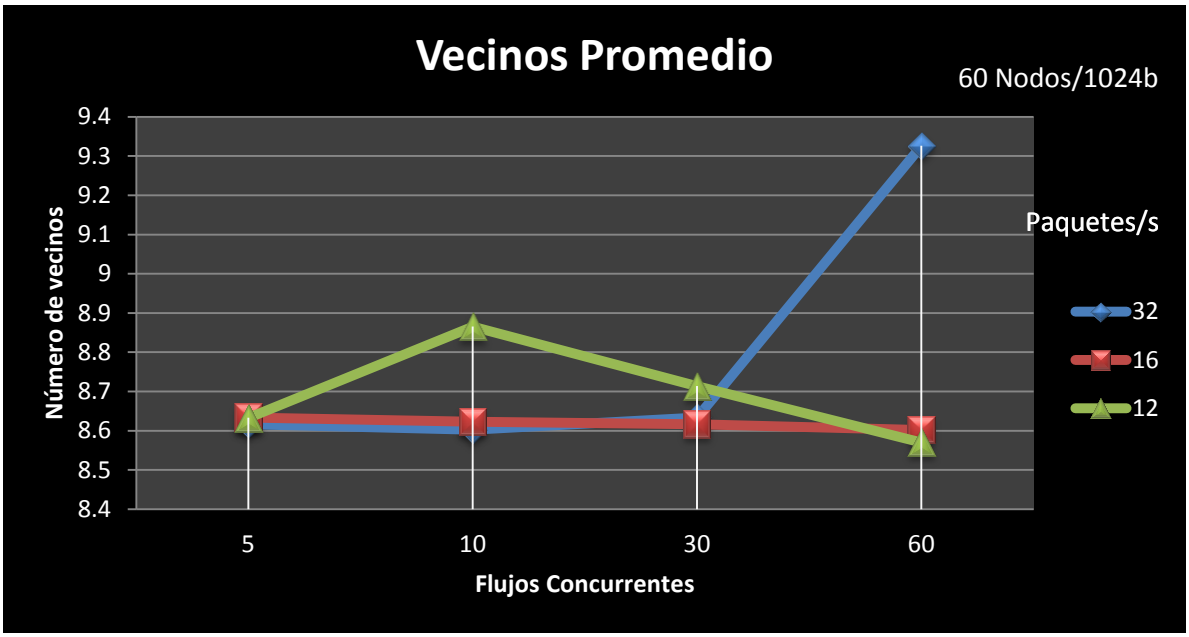


Figura 4.14. Número de vecinos promedio que tiene un nodo en una red de 60 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.7.5 Solicitudes de ser hijo hacia un nodo

Complementando con la sección anterior, se presentan en la Figura 4.15 y la Figura 4.16 las solicitudes máximas promedio de hijo hacia un nodo existente en la red dada, aunque existiera una solicitud mayor de nodos a la soportada, lo que realiza el algoritmo es procesar dicha solicitud y notificar que el nodo no puede ser su padre para que busque a otro padre candidato, así también el algoritmo detecta si no se ha recibido respuesta de la solicitud, y en caso de que el tiempo de holgura para la respuesta haya expirado busca otro padre candidato. Dado el posicionamiento aleatorio que tuvieron los nodos en la Figura 4.15 se encuentra una mayor solicitud de ser hijo hacia un padre candidato. Los picos presentados se deben al posicionamiento aleatorio de los nodos, lo cual indica que existe una gran cantidad de nodos vecinos en cierta región de la red.

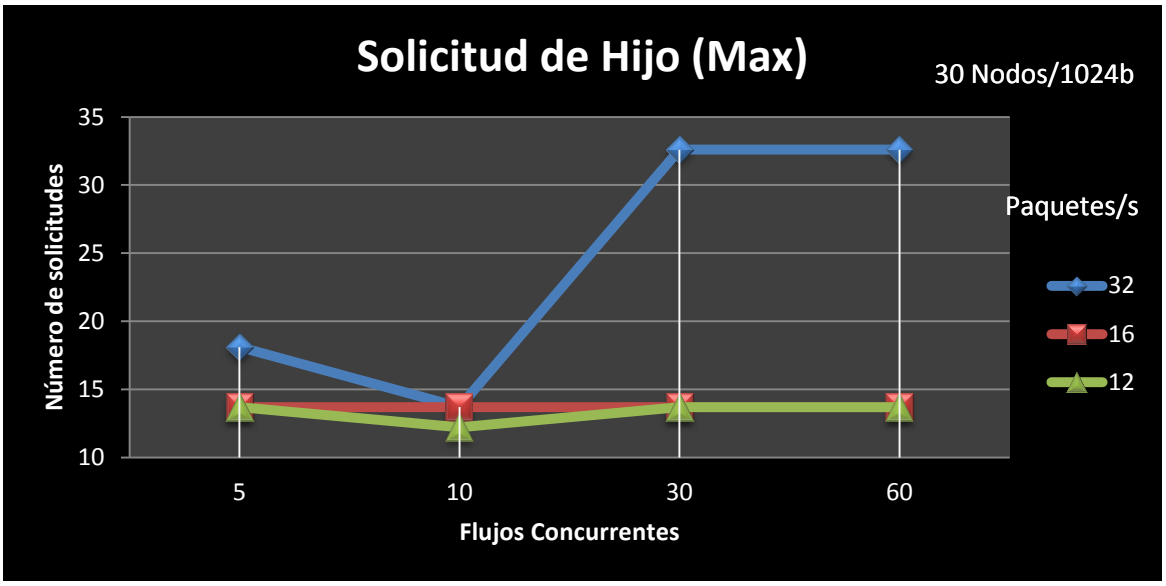


Figura 4.15. Número de solicitudes promedio de hijo que tiene un nodo en una red de 30 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

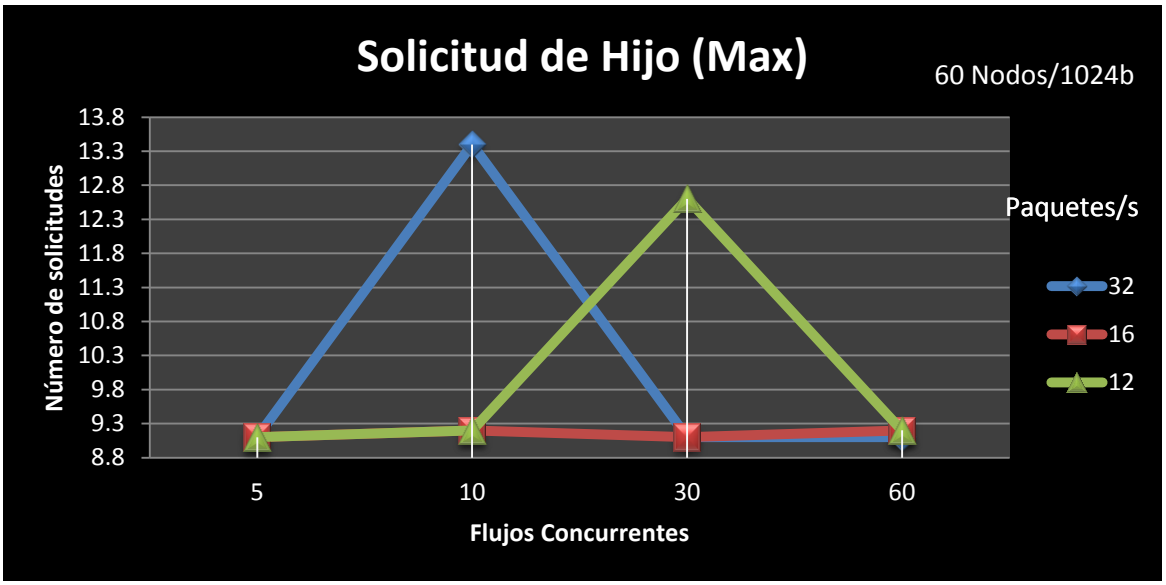


Figura 4.16. Número de solicitudes promedio de hijo que tiene un nodo en una red de 60 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.8 Comparativa. AODV con un servidor de nombres (dns)

En las siguientes secciones se presentan algunas características de la implementación del algoritmo de enrutamiento AODV con un servidor de nombre, comparados con la propuesta de esta tesis.

4.2.8.1 Paquetes entregados

En la Figura 4.17 y la Figura 4.18 se presenta el cociente de paquetes entregados para una red de 30 y 60 nodos respectivamente, el cual, comparado con los presentados por el algoritmo propuesto (Figura 4.1 y la Figura 4.2) muestran un mejor resultado, siendo casi del doble. La razón se encuentra en que AODV hace un manejo de colas de paquetes, mientras que la propuesta, dada su implementación, solo desecha los paquetes antiguos; así también se debe a que AODV realiza el conocimiento de la red únicamente cuando se desconoce el destino y/o ha expirado el tiempo de vida de la ruta.

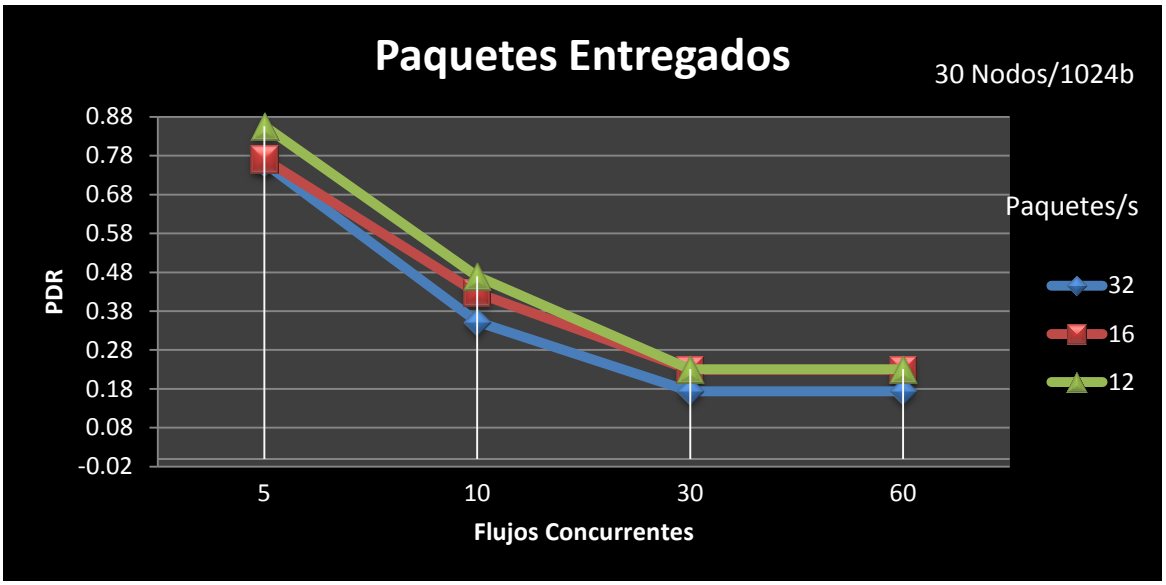


Figura 4.17. Porcentaje de paquetes entregados en una red de 30 nodos con una capacidad de 1024bytes cada paquete, en 5, 10, 30 y 60 flujos concurrentes con una velocidad de envío de 32, 16 y 12 sobre segundo en paquetes CBR.

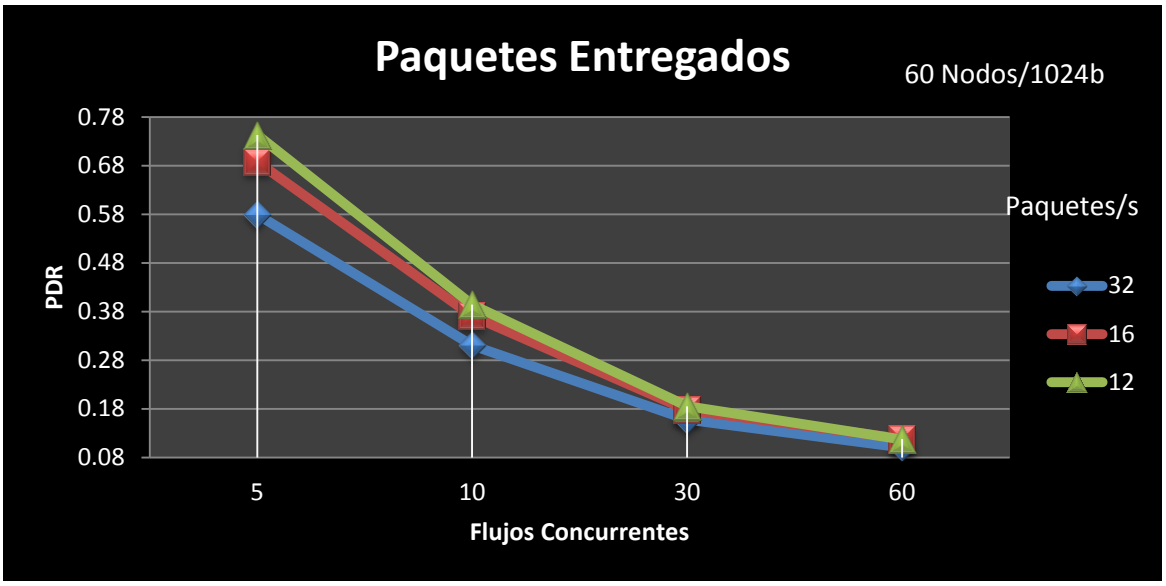


Figura 4.18. Porcentaje de paquetes entregados en una red de 60 nodos con una capacidad de 1024bytes cada paquete, en 5, 10, 30 y 60 flujos concurrentes con una velocidad de envío de 32, 16 y 12 sobre segundo en paquetes CBR.

4.2.8.2 Retardo extremo – extremo

En la Figura 4.19 y la Figura 4.20 se presenta el retardo de entrega de paquetes en una red de 30 y 60 nodos respectivamente con la implementación de AODV con un nodo como servidor de nombres. Comparado con los resultados obtenidos de la propuesta (la Figura 4.3 y la Figura 4.4), se observa que esta es más eficiente (ofrece un menor tiempo de entrega), llegando a un máximo y manteniéndose allí. La posible causa es el retardo de reenvío de los paquetes ya que en la propuesta todo reenvío y envío de paquetes cuenta con un retraso de procesamiento, mientras que AODV sólo lo aplica para algunos casos, el motivo por el que se eligió este comportamiento para la implementación es debido a que es la primera y se consideró sólo el evitar colisiones.

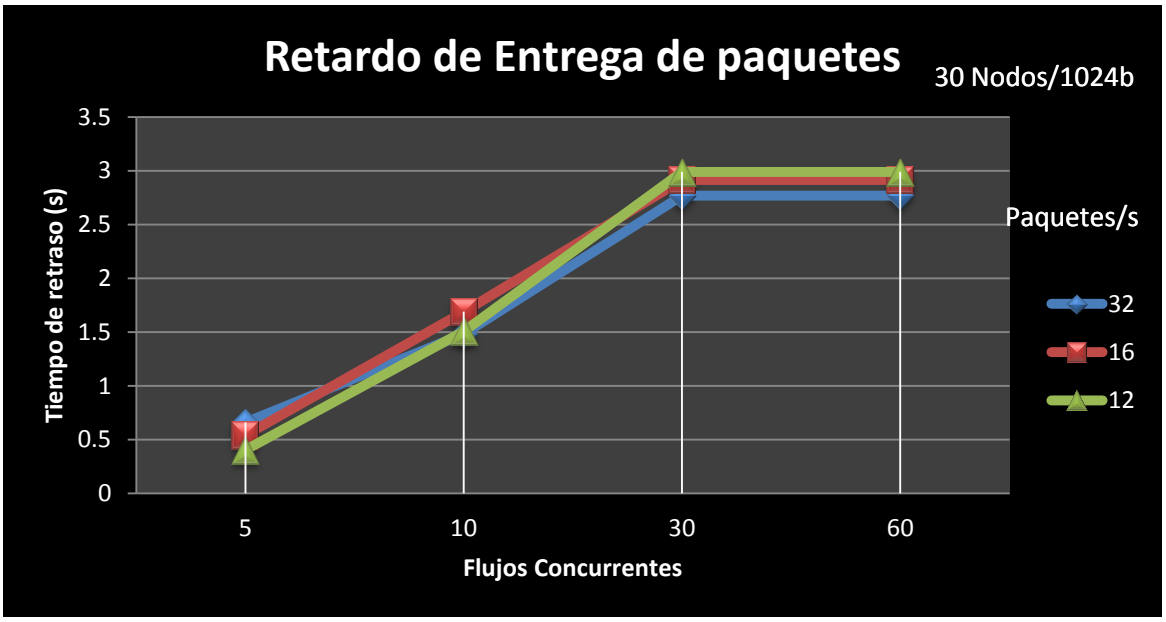


Figura 4.19. Tiempo de retraso en segundos para el retardo de entrega de paquetes en una red con 30 nodos y paquetes CBR de una capacidad de 1024bytes a una velocidad de envío de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

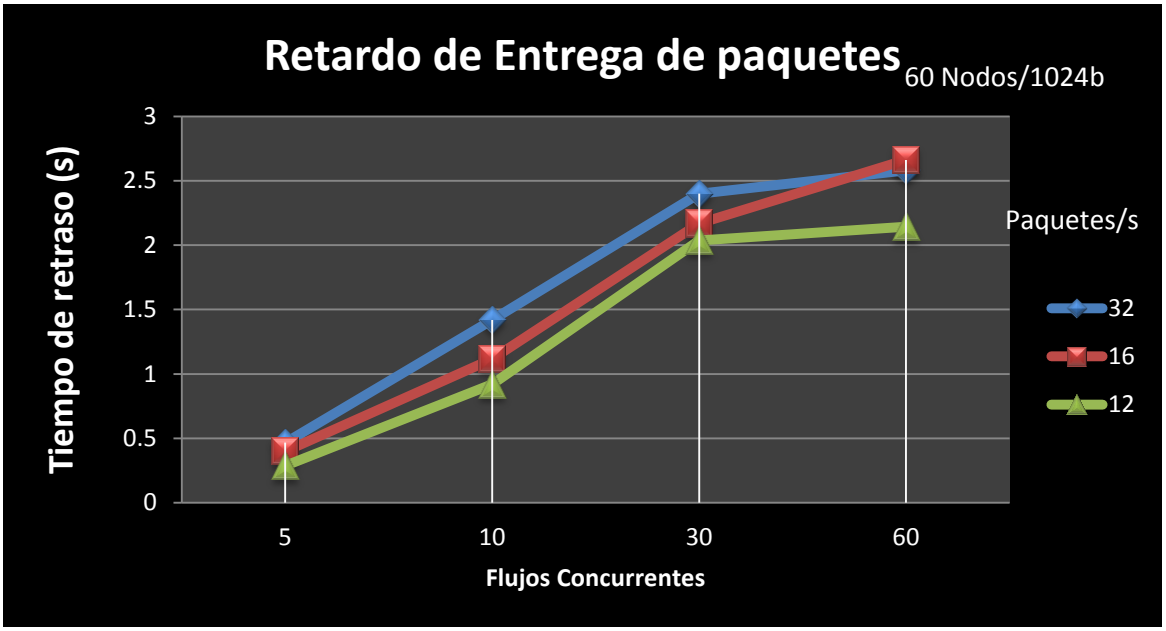


Figura 4.20. Tiempo de retraso en segundos para el retardo de entrega de paquetes en una red con 60 nodos y paquetes CBR de una capacidad de 1024bytes a una velocidad de envío de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60

4.2.8.3 Sobrecarga de paquetes de control

De acuerdo a la Figura 4.21 y a la Figura 4.22, comparando cada una con los resultados de la Figura 4.5 y la Figura 4.6, se puede observar que la modificación de AODV trabaja con un tercio de paquetes de control de acuerdo a como se maneja con la propuesta de esta tesis. Esto se debe a que AODV sólo realiza el conocimiento de la red cuando se necesita (desconoce un nodo que debe de enviar información o la ruta que se conocía a caducado), mientras que la propuesta de la tesis siempre está enviando información (para saludar a los nodos vecinos, para publicarse), lo cual genera estos comportamientos.

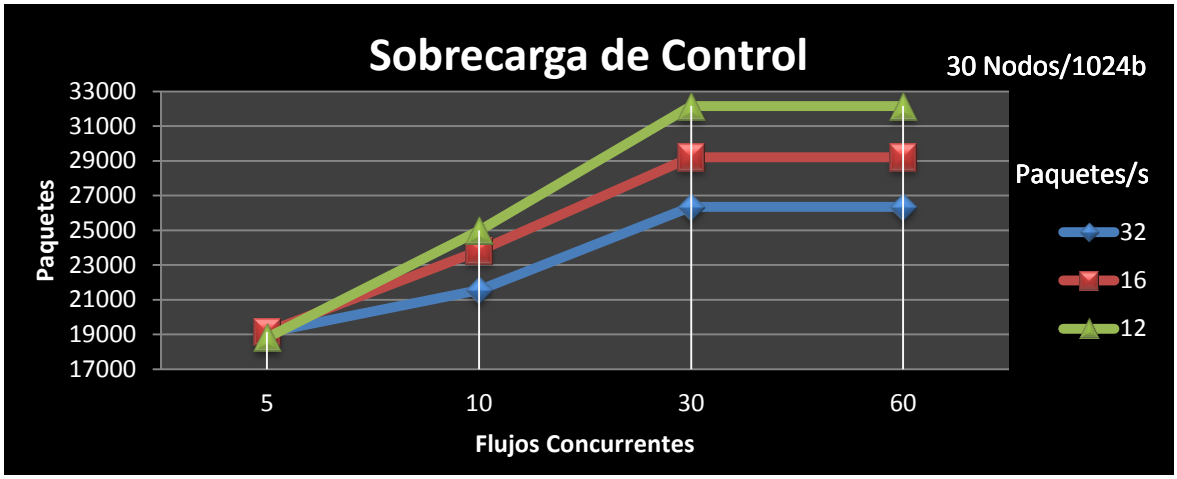


Figura 4.21. Número de paquetes de control enviados en una red de 30 nodos con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

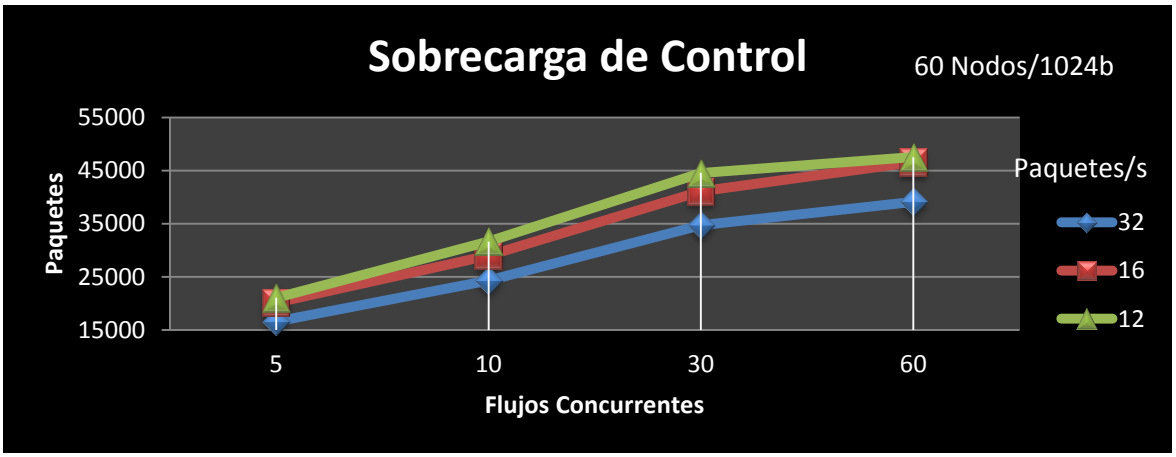


Figura 4.22. Número de paquetes de control enviados en una red de 60 nodos con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.8.4 *Promedio de respuesta de ruta entre la solicitud de rutas*

De acuerdo con la Figura 4.23 y la Figura 4.24 y en comparación con las Figura 4.7 y la Figura 4.8, se puede observar que la propuesta de esta tesis siempre encuentra más rutas que la modificación de AODV, esto se debe a la forma en como está desarrollado el algoritmo propuesto, lo cual indica que se debe de disminuir el número de paquetes de control para poder hacer una mejor entrega de paquetes.

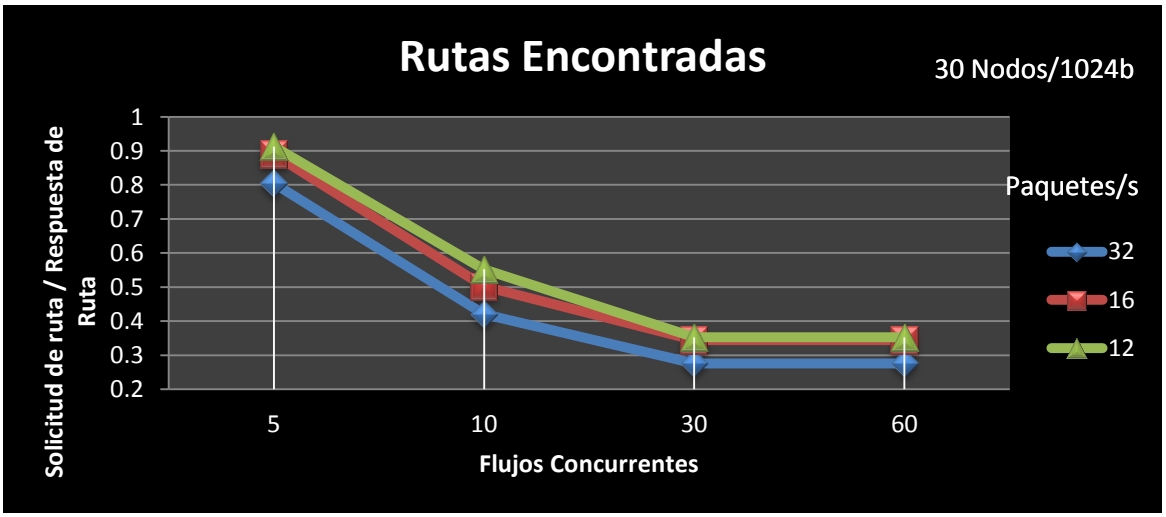


Figura 4.23. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 30 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

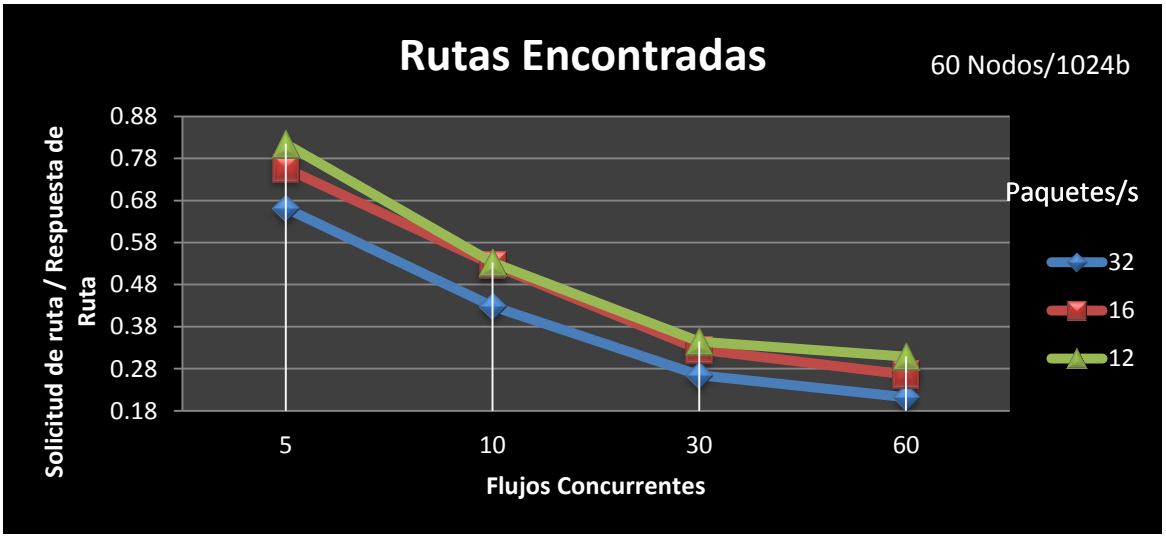


Figura 4.24. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 60 nodos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

4.2.8.5 *Promedio de respuesta de ruta entre la solicitud de rutas (contemplando la perdida de paquetes)*

Si se eliminan de las solicitudes de paquetes aquellas que se perdieron por que estaba saturada la cola de paquetes se obtienen los resultados presentados en la Figura 4.25 y la Figura 4.26, los cuales si se comparan con los obtenidos con la propuesta de esta tesis (Figura 4.9 y Figura 4.10) se puede ver que AODV modificado presenta un menor rendimiento que la propuesta de la tesis. Lo anterior indica y refuerza que se debe de mejorar la sobrecarga de paquetes de control, para mejorar la tasa de paquetes entregados.

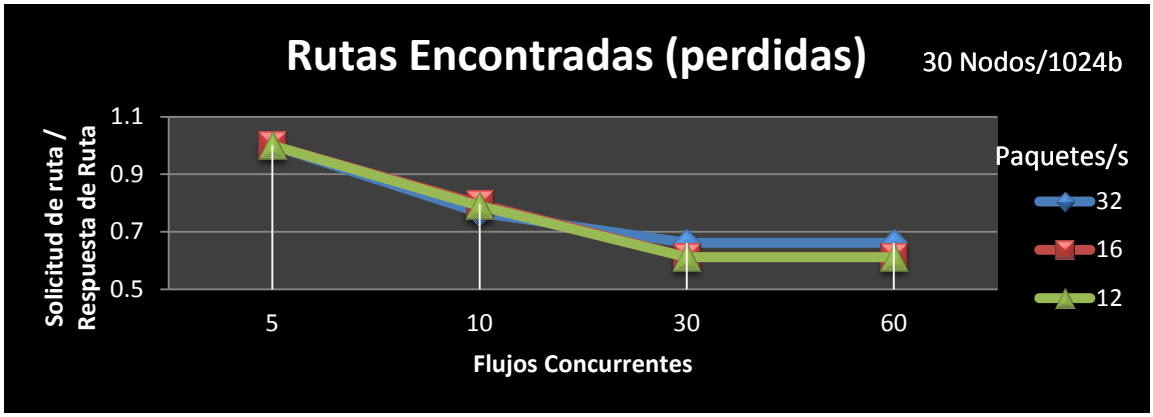


Figura 4.25. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 30 nodos, omitiendo los paquetes perdidos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

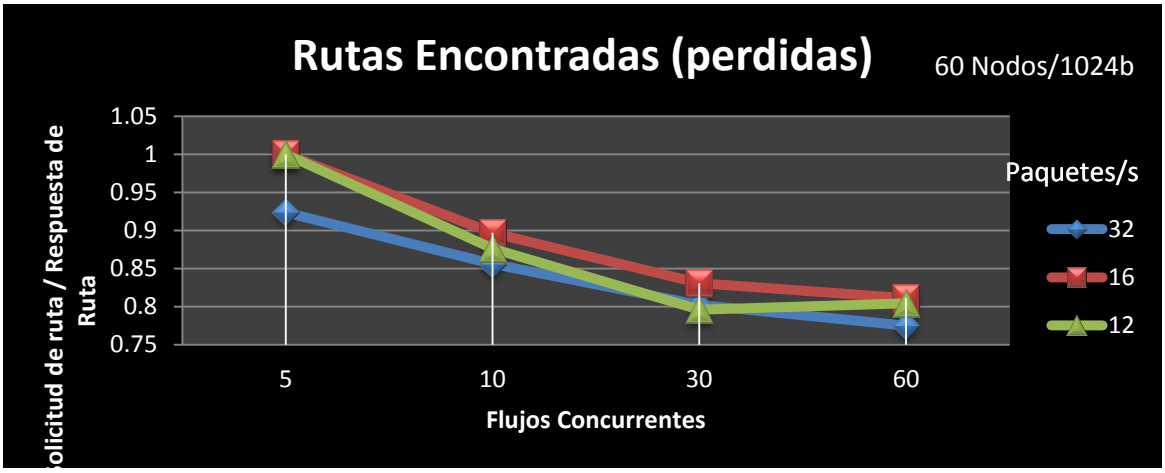


Figura 4.26. Cociente de la respuesta de ruta entre la solicitud de ruta en una red de 60 nodos, omitiendo los paquetes perdidos, con paquetes CBR de una capacidad de 1024bytes enviados a una velocidad de 32, 16 y 12 paquetes sobre segundo en flujos concurrentes de 5, 10, 30 y 60.

Capítulo 5 Conclusiones y trabajo a futuro

5.1 Logros alcanzados y limitaciones

En este documento se ha presentado, implementado y simulado un algoritmo de enrutamiento para redes inalámbricas que está basado en prefijos, la cual es un área muy poco explorada que presenta un gran número de retos y oportunidades de investigación.

El algoritmo presentado, muestra algunas características interesantes como es la elección aleatoria del nodo líder, la cual es eficiente y estable debido a que el criterio usado para la selección es independiente de las condiciones de la red y de la topología. La elección aleatoria se realiza simplemente eligiendo como líder al nodo con el identificador más grande. Este esquema también tiene sus limitaciones ya la si se elige como líder un nodo que se encuentra en la periferia provocará que la longitud de los caminos crezca. Queda como trabajo de investigación futuro proponer y caracterizar nuevos esquemas de elección del líder.

Una vez que se elige al nodo líder, se procede a establecer relaciones padre-hijo entre los nodos que componen la red, formando así un árbol con raíz en el nodo líder. Las relaciones padre-hijo son posteriormente usadas para asignar etiquetas a los nodos que componen la red, tal como se explica en la Sección 2.6 titulada “Redes orientadas a contenido”. Dicho etiquetado es la base para realizar la retransmisión de paquetes hacia el contenido destino y no hacia identificadores fijos de los nodos, es decir, para implementar el enrutamiento orientado a contenido. Así también se demostraron tanto teórica como experimentalmente dos características importantes de esta implementación, las cuales se expusieron en el Teorema 4.1. Las tablas de enrutamiento son del orden $\boldsymbol{O(d)}$ donde *d* es el grado máximo de la red., y el Teorema 4.2. La relación entre los caminos encontrados por el algoritmo propuesto y los caminos más cortos existentes en la red (*stretch*) es $\boldsymbol{O}\ (\mathbf{log}\,\boldsymbol{n})$. Los resultados experimentales muestran que para el primer teorema, el número máximo de vecinos concuerda con el tamaño máximo esperado. Para el segundo teorema, los resultados experimentales mostraron que en todas las simulaciones el tamaño promedio de saltos era de dos unidades, y el salto máximo se llegaba a dar cuando iba del último hijo posible de la red a otro extremo, siendo un salto de 14 unidades el máximo que se llegó a registrar, todo lo anterior realizado en un una red con 60 nodos, tal como se mencionó en la Sección 4.2.7.3.

Por último, con respecto a la serie de experimentos, los resultados obtenidos muestran que en comparación con la implementación de AODV trabajando en conjunto con un servidor de nombres (una de las cosas que se evita con la implementación de la propuesta), la entrega de paquetes con la implementación ofrece un resultado inferior, así como una mayor tasa de paquetes de control. Aún con estas limitantes descritas, se obtiene una mejor tasa de respuesta en cuanto a las solicitudes de ruta, lo cual indica que se pueden desarrollar y/o mejorar las implementaciones de esta propuesta para ofrecer resultados más competitivos, así como la forma en que se etiqueta, ya

que como está desarrollado el algoritmo los nodos se encuentran constantemente enviando esta información. Una de las principales causas de pérdida de paquetes es que las colas de datos tienden a congestionarse y los nuevos paquetes que son formados son simplemente desechados. En comparación, la implementación de AODV no lograba cumplir con las solicitudes de ruta pese a que ellos sí implementan un control dinámico de los paquetes almacenados. Aun así, se considera que la presente tesis presenta un aporte original debido a que en el desarrollo de este trabajo no se encontró alguna implementación en simulador NS2 que realizara lo que se propone, a pesar de que se terminó desarrollando con nodos móviles en lugar de nodos fijos principalmente por problemas de implementación y tiempo.

5.2 Aportaciones

Dentro de las contribuciones, se consiguió desarrollar la base para la futura experimentación con ruteo basado en prefijos, ya que el algoritmo cuenta con módulos y funciones interesantes para poder realizar el etiquetado de una forma fácil y con un consumo de recursos de cómputo moderado.

Así también se desarrollaron funciones que pueden ser fácilmente mejorables o sustituidas que pueden permitir la experimentación, tal es el caso de las funciones hash que se emplea para etiquetar a los nodos.

Todo lo anterior ha sido desarrollado en el ambiente de simulación NS2 que tiene la ventaja de proveer un entorno de simulación realista que considera tanto los efectos de la capa física como la interacción de las diferentes capas que componen a un sistema de comunicaciones real.

Desde el punto de vista teórico, se desarrollaron dos teoremas que caracterizan el comportamiento del esquema de etiquetado propuesto. En particular, se presenta una cota para el stretch máximo y para el tamaño de las tablas de enrutamiento.

5.3 Trabajos Futuros

De este trabajo surgen varias alternativas para reforzarlo y mejorar algunas cualidades, entre ellas se destacan los siguientes puntos:

En primera instancia se puede experimentar con otras alternativas para seleccionar el nodo raíz. Por ejemplo se puede intentar calcular el centro de la red para así seleccionar al nodo que minimice la longitud del camino máximo y por lo tanto el número de saltos que tienen que dar los

paquetes para llegar a su destino. Este esquema, sin embargo, tiene la desventaja de que si la red es dinámica, el centro de la red también lo será lo que provocaría la necesidad de estar cambiando constantemente de raíz y por lo tanto de re-etiquetar la red. Adicionalmente se puede experimentar con el impacto de tener más de un nodo raíz y establecer así un espacio multidimensional basado en los diferentes etiquetados asociados a cada uno de los nodos raíz.

Por otro lado, el esquema basado en prefijos también muestra limitantes en cuanto a la dificultad para reparar los etiquetados ante cambios topológicos. Futuros trabajos tendrán que proponer y evaluar otros esquemas para establecer las etiquetas en los nodos.

Adicionalmente, se puede considerar la implementación de cachés en los nodos para disminuir la sobrecarga inducida en la red, al transportar contenido entre dos nodos ubicados en regiones diferentes de la red.

Capítulo 6 Bibliografía

[1] Andrew S. Tanenbaum, *Computer Networks* , Cuarta ed.: Prentice Hall, 2003.

[2] Vinton Gray Cerf and Robert E. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Trans on Comms*, vol. Com-22, no. 5, pp. 637-648, Mayo 1974.

[3] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425-432, Abril 1980.

[4] Ipoque. (2011, Octubre) Ipoque - Internet Studies. [Online]. http://www.ipoque.com/en/resources/internet-studies

[5] Cisco and/or its affiliates. (2011, Enero) Cisco - Entering the Zettabyte Era. [Online]. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.pdf

[6] Jaeyoung Choi, Jinyoung Han, Eunsang Cho, Ted Kwon, and Yanghee Choi, "A Survey on Content-Oriented Networking for Efficient Content Delivery," *Communications Magazine*, vol. 49, no. 3, pp. 121-127, 2011.

[7] (2009) Ipoque Internet Study 2008-2009. [Online]. http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf

[8] Wikipedia. (2012, Jan.) Wikipedia. [Online]. http://es.wikipedia.org/wiki/Capas_o_niveles_del_TCP/IP

[9] Deepankar Medhi and Karthikeyan Ramasamy, *Network Routing: Algorithms, Protocols, And Architectures*.: Morgan Kaufmann, 2007.

[10] Wikipedia. (2012, Noviembre) Routing table. [Online]. http://en.wikipedia.org/wiki/Routing_table

[11] Ajay D. Kshemkalyani and Mukesh Singhal, *Distributed Computing: Principles, Algorithms, and Systems*, Primera ed.: Cambridge University Press, 2008.

[12] Jie Wu and Li Sheng, "Deadlock-free routing in irregular networks using prefix routing," *Parallel Processing Letters*, vol. 13, no. 4, pp. 705-720, 2003.

[13] John F., Koegel Buford, and Hong Heathe, *P2P Networking and Applications*.: Morgan Kaufmann, 2009.

[14] Wikipedia. (2012, Oct.) Red Superpuesta. [Online]. http://es.wikipedia.org/wiki/Red_superpuesta

[15] Wikipedia. (2012, Nov.) Peer-to-peer. [Online]. http://es.wikipedia.org/wiki/Peer-to-peer

[16] I. Stoica et al., "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *Networking, IEEE/ACM Transactions on*, vol. 11, no. 1, pp. 17-32, 2003.

[17] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," 2001.

[18] Wikipedia. (2012, Nov.) Distributed hash table. [Online]. http://en.wikipedia.org/wiki/Distributed_hash_table

[19] R. L. Rivest, "The MD5 Message-Digest Algorithm," IETF, RFC 1321, 1992.

[20] D. Eastlake, "US Secure Hash Algorithm 1 (sha1)," IETF, RFC 3174, 2001.

[21] Gurmeet Singh Manku, "Dipsea: A Modular Distributed Hash Table," Stanford University, Ph.D. Thesis 2004.

[22] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A Scalable Content-Addressable Network," *SIGCOMM*, pp. 161-172, Agosto 2001.

[23] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," *International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, Noviembre 2001.

[24] Moni Naor and Udi Wieder, "Novel Architectures for P2P Applications: the Continuous-Discrete Approach," SPAA, 2003.

[25] Van Jacobson et al., "Networking Named Content," in *CoNEXT 2009*, Roma, 2009.

[26] Giannis F. Marias, George C. Polyzos Nikos Fotiou, "Information Ranking in Content-Centric Networks," in *Future Network and MobileSummit 2010 Conference Proceedings*, Florencia, 2010.

[27] Yosra Barouni, Marco Bicudo, and Promethee Spathis, "Content Centric Routing for Future Generation Networks," in *INFOCOM Workshops 2009*, 2009.

[28] Dmitriy Kuptsov, Petri Savolainen, Pasi Sarolahti Sasu Tarkoma, "CAT: A Last Mile Protocol for Content-Centric," in *Communications Workshops (ICC)*, 2011.

[29] Xinwen Zhang et al., "Towards Name-based Trust and Security for Content-centric Network," in *Network Protocols (ICNP)*, 2011.

[30] aeyoung Choi et al., "Performance comparison of content-oriented networking alternatives: A tree versus a distributed hash table," in *Local Computer Networks*, Zúrich, 2009.

[31] Jiachen Chen, M. Arumaithurai, Lei Jiao, Xiaoming Fu, and K.K. Ramakrishnan, "COPSS: An Efficient Content Oriented Publish/Subscribe System," in *Architectures for Networking and Communications Systems (ANCS)*, 2011, pp. 99-110.

[32] J.L. Martins and S. Duarte, "Routing Algorithms for Content-Based Publish/Subscribe Systems," *Communications Surveys & Tutorials*, vol. 12, no. 1, pp. 39-58, 2010.

[33] A. Carzaniga, M.J. Rutherford, and A.L. Wolf, "A Routing Scheme for Content-Based Networking ," *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 918-928, 2004.

[34] H. Nishikawa and R. Kurebayashi, "A Networking Oriented Data-Driven Processor: CUE," *Performance Processors and Systems*, pp. 103-111, 2002.

[35] Dmitrij Lagutin, Kari Visala, Andras Zahemszky, Trevor Burbridge, and Giannis F. Marias, "Roles and Security in a Publish/Subscribe Network Architecture," in *Computers and Communications (ISCC)*, 2010, pp. 68-74.

[36] Koponen Teemu et al., "A Data-Oriented (and Beyond) Network Architecture, DONA," in *SIGCOMM'07*, Kioto, 2007, pp. 181-192.

[37] Gritter Mark and R.-Cheriton David, "An Architecture for Content Routing Support in the Internet," in *ACM Computing Surveys 35*, 2003, pp. 114-131.

[38] Visala Kari, Lagutin Dmitrij, and Tarkoma Sasu, "LANES: An Inter-Domain Data-Oriented Routing," in *ReArch'09*, Roma, 2009, pp. 55-60.

[39] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *Internet Computing, IEEE* , vol. 7, no. 6, pp. 68-74, 2003.

[40] J. Apostolopoulos, T. Wong, Wai-tian Tan, and S. Wee, "On multiple description streaming with content delivery networks," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1736-1745,

2002.

[41] C.D. Cranor et al., "Enhanced streaming services in a content distribution network," *Internet Computing, IEEE* , vol. 5, no. 4, pp. 66-75, 2001.

[42] Buyy Rajkumar and Pathan Mukaddim, *Content Delivery Networks*.: Springer, 2008.

[43] D., Lehman, E., Leighton, F., Levine, M., Lewin, D., and Panigrahy, R Karger, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web," *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* , pp. 654-663, Mayo 1997.

[44] D. Lewin, "Consistent hashing and random trees: Algorithms for caching in distributed networks.," *Master's thesis, Department of EECS, MIT*, 1998.

[45] Ben Y. Zhao et al., "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected areas in communications*, vol. 22, no. 1, Enero 2004.

[46] Plaxton C. Greg, Rajaraman Rajmohan, and W. Richa. Andrea, "Accessing nearby copies of replicated objects in a distributed environment," *Proceedings of ACM SPAA*, Junio 1997.

[47] Dhananjay Sampath and J. J. Garcia-Luna-Aceve, "Scalable Integrated Routing Using Prefix Labels and Distributed Hash Tables for MANETs," in *MASS 2009*, 2009, pp. 188-198.

[48] Dhananjay Sampath and J. J. Garcia-Luna-Aceves, "PROSE: Scalable Routing in MANETs Using Prefix Labels and Distributed Hashing," in *SECON 2009*, 2009, pp. 1-9.

[49] Perkins, C.; Belding-Royer, E.; Das, S, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF, RFC 3561, 2003.

[50] Dhananjay Sampath, Suchit Agarwal, and J. J. Garcia-Luna-Aceves, "Ethernet on AIR': Scalable Routing in very Large Ethernet-Based Networks," in *ICDCS 2010*, 2010, pp. 1-9.